ORIGINAL PAPER

Boosting Human Decision-making with AI-Generated Decision Aids

Frederic Becker[1] · Julian Skirzyński[1,2] · Bas van Opheusden[3] · Falk Lieder[1]

Abstract

Human decision-making is plagued by many systematic errors. Many of these errors can be avoided by providing decision aids that guide decision-makers to attend to the important information and integrate it according to a rational decision strategy. Designing such decision aids used to be a tedious manual process. Advances in cognitive science might make it possible to automate this process in the future. We recently introduced machine learning methods for discovering optimal strategies for human decision-making automatically and an automatic method for explaining those strategies to people. Decision aids constructed by this method were able to improve human decision-making. However, following the descriptions generated by this method is very tedious. We hypothesized that this problem can be overcome by conveying the automatically discovered decision strategy as a series of natural language instructions for how to reach a decision. Experiment 1 showed that people do indeed understand such procedural instructions more easily than the decision aids generated by our previous method. Encouraged by this finding, we developed an algorithm for translating the output of our previous method into procedural instructions. We applied the improved method to automatically generate decision aids for a naturalistic planning task (i.e., planning a road trip) and a naturalistic decision task (i.e., choosing a mortgage). Experiment 2 showed that these automatically generated decision aids significantly improved people's performance in planning a road trip and choosing a mortgage. These findings suggest that AI-powered boosting might have potential for improving human decision-making in the real world.

Keywords Improving human decision-making · Boosting · Decision aids · Far-sightedness · Interpretable machine learning · Automatic strategy discovery

Introduction

Many researchers working on judgment and decision-making (e.g., Tversky & Kahneman, 1974; Gilovich et al., 2002) have argued that human decision-making is plagued by many systematic errors known as cognitive biases (but see Gigerenzer, 1991; Gigerenzer et al., 2008). In particular, older people often come to regret the short-sighted decisions they made about their health, education, and finances when they were younger (Kinnier & Metha, 1989). Consistent with this

observation, numerous experiments on intertemporal choice have consistently found that people's decisions depend primarily on the immediate outcomes of potential choices and underweight their more weighty long-term consequences (Milkman et al., 2008; O'Donoghue & Rabin, 2015). Those short-sighted choices can be traced back to the strategies people use to make decisions (Reeck et al., 2017). Therefore, one way to address such problems is to improve people's decision strategies. This is an instance of *boosting* (Hertwig & Grüne-Yanoff, 2017). Boosting human decision-making has many benefits over delegating decisions to algorithms. Firstly, there are multiple areas, such as medicine and the judicial system, in which humans are and will continue to be the ultimate decision-makers for ethical reasons. Secondly, people have to make many subjective decisions that depend on their personal values. Helping people make better decisions in these settings is crucial for increasing our society's well-being.

Boosting can be implemented by training people or by supporting them while they make a decision. The benefits of

Frederic Becker and Julian Skirzyński share joint first authorship.

✉ Frederic Becker
  frederic.becker@tuebingen.mpg.de

1 Max Planck Institute for Intelligent Systems, Tübingen, Germany

2 University of California, San Diego, San Diego, CA 92093, USA

3 Princeton University, Princeton, NJ 08544, USA

training on a simple task rarely transfer to the more complex problems people face in the real world (Sala & Gobet, 2017; Sala et al., 2019; Becker et al., 2021). One way to side-step this problem is to provide people with decision aids that support them directly in those decisions that are to be improved (Fig. 1). Previous research showed that real-world decisions in the domains of finance and medicine can be improved by providing people with decision aids (e.g., a decision tree) that guide them through the application of clever heuristic that direct the decision-maker's attention to the most essential pieces of information (Hafenbrädl et al., 2016). Recent work has developed algorithms for gen-erating and visualizing decision trees automatically (Phillips et al., 2017; Rudin et al., 2022). However, equivalent tools do not yet exist for helping people solve complex planning problems. Designing decision aids for such problems by hand can be very tedious, and coming up with clever heuris-tics can very difficult. As a first step toward addressing this problem, we recently developed a computational method for automatically designing decision aids for sequential decision problems that require planning (Skirzyński et al., 2021a). This AI-powered boosting method leverages Artificial Intelligence (AI) to derive smart decision strategies from a mathematical theory of optimal decision-making with finite time, limited time, and bounded cognitive resources (Lieder & Griffiths, 2020b). The decision aids constructed by this method were able to improve human decision-mak-ing (Skirzyński et al., 2021a). However, using these deci-sion aids is tedious because they do not explicitly specify the decision process directly. Rather, they specify a process for determining whether the next step the decision-maker is considering to take is consistent with the recommended heuristic. The goal of this article is to develop an improved method that can generate decision aids that directly describe the automatically discovered heuristic in natural language.

We hypothesized that step-by-step instructions for how to reach a decision would be significantly easier for people to follow than the decision aids generated by our previous method (Skirzyński et al., 2021a). After confirming that such *procedural instructions* are more interpretable and easier to

follow than the previous version of our decision aids, we developed a new algorithm for transforming the output of the previous method (Skirzyński et al., 2021a) into procedural instructions. The extended method automatically generates step by step, natural language instructions for how to reach a decision. These instructions are generated from a pair of two inputs: (i) a model of the general structure of a particular (sequential) decision problem, and (ii) a dictionary of what the relevant components of the general problem are called in the concrete application. This approach is very general and can be applied to different kinds of decision problems. In particular, the two algorithms we utilized to set up our approach give rise to a new policy-agnostic method for inter-preting reinforcement learning policies, which is an impor-tant part of the problem known as explainable reinforcement learning (Puiutta & Veith, 2020; Dazeley et al., 2021). As a proof of concept, we demonstrate that our method can be used to make more far-sighted choices in two naturalistic decision-tasks: planning a road trip and choosing a mortgage (see Fig. 2a and b). We found that people can understand and follow the automatically generated procedural instruc-tions in both tasks and consequently made better decisions. These findings suggest that AI-powered boosting can be very effective at improving human decision-making in naturalistic tasks.

The article is structured as follows. In the next section we provide further information on approaches striving to improve human decision-making and discuss the connec-tions between our method and other approaches to deci-sion support. In the third section, we show that procedural descriptions of planning strategies are better suited to improve human decision-making than the descriptions gen-erated by the previous method (Skirzyński et al., 2021a). In the following section, we present a new method for auto-matically generating procedural descriptions of planning strategies. In Section 5, we test the extent to which decision aids obtained with our method improve human decision-making in more naturalistic decision problems. Lastly, we summarize and discuss our findings in Section 6.
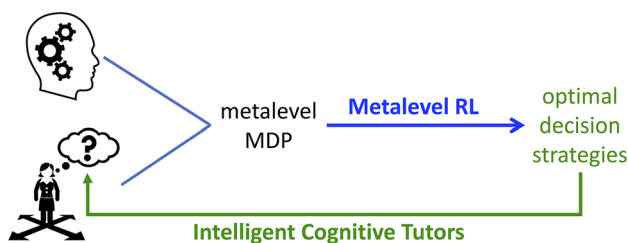
Background

Reinforcement Learning

To understand the underlying principles and mechanisms of AI-powered boosting, we provide a handful of definitions for mathematical constructs that are used by this approach. Those constructs regard the theory of reinforcement learn-ing that enables to compute solutions to sequential decision problems that are provably optimal. Moreover, these meth-ods play an important role in our approach to discovering optimal cognitive strategies for human decision-making.

Fig. 1 AI-powered boosting relies on discovering optimal decision strategies by modeling decision problems as a metalevel Markov Decision Process and solving them with metalevel reinforcement learning (Griffiths et al., 2019)

Fig. 2 Naturalistic decision-tasks used in Experiment 2. **a** In the Road Trip task, the objective is to efficiently find an inexpensive route to a city with an airport. Participants can use a search engine to look up how costly it is to spend the night in different cities. **b** In the Mortgage task the goal is to choose the most affordable mortgage by considering interest rates for different time horizons. Participants can learn about the interest rates by clicking on appropriate boxes
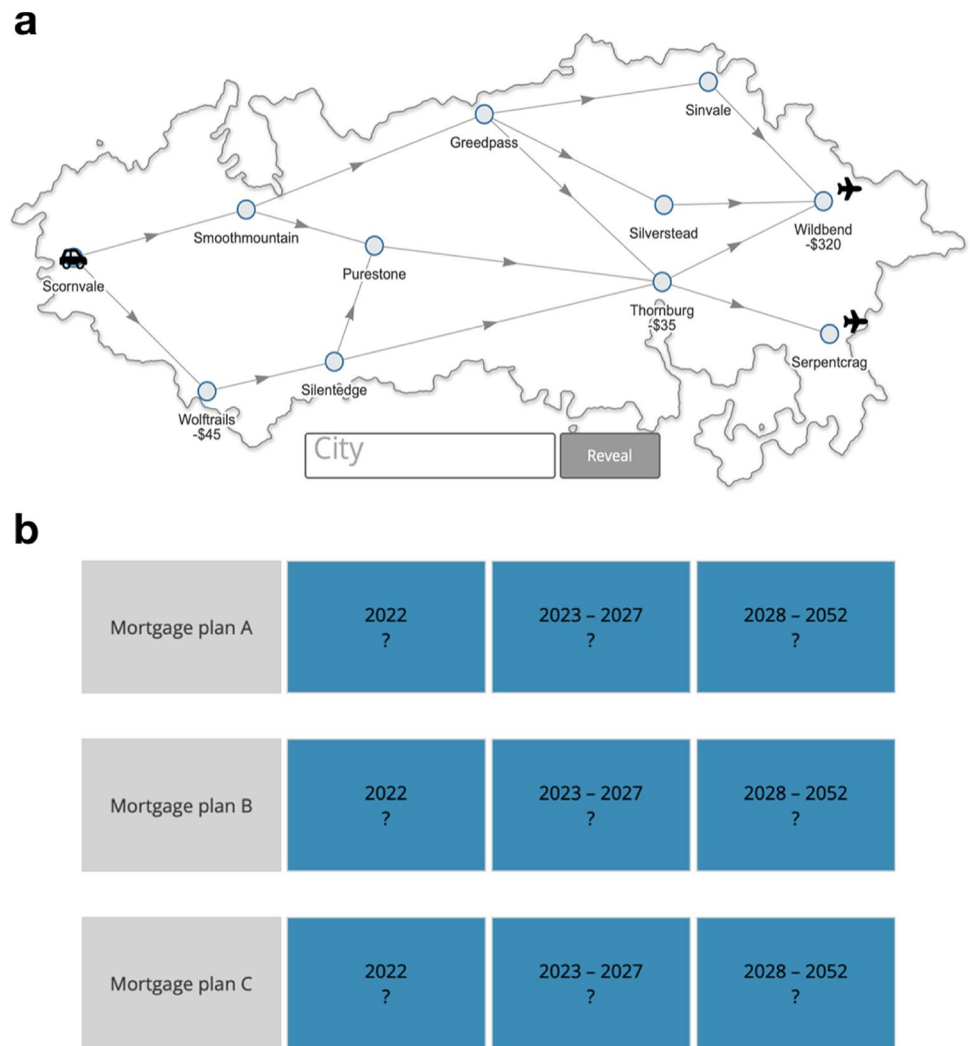
The first definition presents a structure that is a general model of decision-making problems called *Markov Decision Process* (see Definition 1). Representing a decision problem in terms of a Markov Decision Process becomes possible by specifying what states exist in this problem, what actions might be taken, how valuable (rewarding) each of the actions is in the states, and how the actions change the states form one to another.

Definition 1 (**Markov Decision Process**) A Markov decision process (MDP) is a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \gamma)$ where $\mathcal{S}$ is a set of states; $\mathcal{A}$ is a set of actions; $\mathcal{T}(s, a, s') = \mathbb{P}(s_{t+1} = s' \mid s_t = s, a_t = a)$ for $s \neq s' \in \mathcal{S}, a \in \mathcal{A}$ is a state transition function; $\gamma \in (0, 1)$ is a discount factor; $\mathcal{R} : \mathcal{S} \to \mathbb{R}$ is a reward function.

Having modeled a decision problem as an MDP, we may further define a mathematical counterpart of a decision strategy, namely a *policy* (see Definition 2).

Definition 2 (**Policy**) A deterministic policy π is a function $\pi : \mathcal{S} \to \mathcal{A}$ that specifies actions to take in each of the states in the MDP and a non-deterministic policy π is a function $\pi : \mathcal{S} \to Prob(\mathcal{A})$ that defines a probability distribution over the actions for the states in the MDP.

Then, *expected reward* (see Definition 3) serves to quantify the usefulness of policies (i.e. to quantify how valuable or rewarding they are).

Definition 3 (**Expected reward**) The reward r_t represents the quality of performing action a_t in state s_t. The cumulative return of a policy is a sum of its discounted rewards obtained in each step of interacting with the MDP, i.e. $G_t^\pi = \sum_{i=t}^{\infty} \gamma^t r_t$ for $\gamma \in [0, 1]$. The expected reward $J(\pi)$ of policy π is equal to $J(\pi) = \mathbb{E}(G_0^\pi)$.
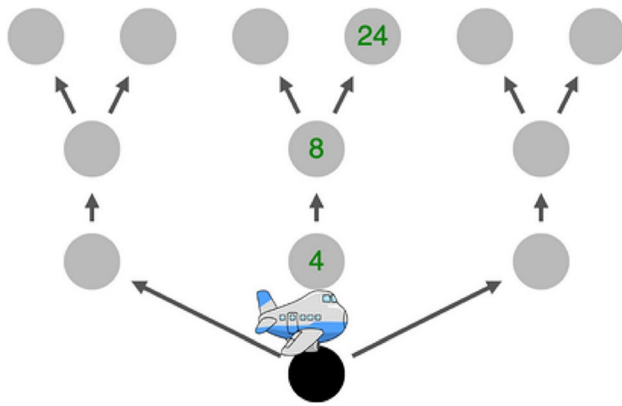
Fig. 3 Mouselab-MDP with increasing variance. Nodes in the Mouselab-MDP represent short-, mid-, and long-term consequences of potential actions where the black node is the starting position, and connections in the Mouselab-MDP symbolize "possible later consequence" relation. Information on the consequences is represented numerically as rewards, and can be acquired by clicking on the nodes. Rewards are drawn from a normal distribution whose variance increases with the distance of the node from the starting node. Thus, rewards that are farthest from the starting position (long-term consequences) show the highest variance. As gathering information is costly, the agent's goal in the Mouselab-MDP is to make as few clicks as possible to uncover the best possible course of actions with the highest total reward

Finally, equipped with these formalizations of decision strategies and their quality in the problem modeled by the MDP, we need an approach for finding strategies whose quality is high. A family of methods which solve that problem and find policies with the highest reward is called *reinforcement learning* (see Definition 4).

Definition 4 (**Reinforcement learning**) Reinforcement learning (RL) is a class of methods that perform iterations over trials and evaluation on a given MDP to find the optimal policy π^* which maximizes the expected reward (Sutton & Barto, 2018).

Modeling Planning as Information Acquisition

In our quest to improve human planning, we used a theoretical framework developed to study and improve human planning, namely the resource-rational theory of optimal human planning (Callaway et al., 2022b, 2018b). According to this theory, planning can be understood as a series of information gathering operations that gradually refine a person's estimates of the short- and long-term consequences of alternative courses of action. This idea gave rise to an empirical paradigm (i.e., the Mouselab-MDP paradigm) that operationalizes human planning by the series of clicks made to collect information about the rewards hidden at different locations of the environment of a path planning task (see Fig. 3). As each click has a cost, finding the best possible

path is a nontrivial problem. According to the theory, by collecting a sequence of clicks people made to decide which path is best, we gain an insight into people's planning process and the strategies they used.

Approaches to Improving Human Decision-making

Interventions designed to tackle decision-making biases include educating people about rational decision-making (Larrick, 2004) and boosting people's decision-making by conveying simple heuristics they can use to arrive at good decisions (Hertwig & Grüne-Yanoff, 2017; Hafenbrädl et al., 2016). In the first approach, harmful biases are tackled through motivating or incentivizing people to make better decisions by showing them what they could gain by adopting the principles of logic, probability theory, and expected utility theory. The key limitation of this approach is that those principles place unrealistically high cognitive demands on people when they are applied to nontrivial real-world problems (Larrick, 2004). In response to the shortcomings of the first approach, the second approach directs people to use adaptive simple heuristics (Hertwig & Grüne-Yanoff, 2017; Hafenbrädl et al., 2016; Gigerenzer & Todd, 1999). These heuristics take advantage of common properties of decision problems, and allow the decision-maker to quickly make advantageous choices while being rather straightforward to understand. Two ways of conveying adaptive heuristics have been explored so far: teaching decision strategies and designing decision aids (Hafenbrädl et al., 2016; Hertwig & Grüne-Yanoff, 2017).

AI-Powered Boosting

The general approach we followed in this study, called AI-powered boosting, is to leverage artificial intelligence (AI) to improve on previous efforts to boost human decision-making. The essence of AI-powered boosting is to employ machine learning to discover decision-making heuristics automatically and then convey them to people (Callaway et al., 2022a; Skirzyński et al., 2021a). This approach rests on defining efficient decision strategies and computing them by solving appropriate optimization problems. According to this approach, optimal strategies for human decision-making do not attempt to maximize the expected outcomes (cf. Definition 3) because that would be intractable for people. Computing the optimal decisions for every possible situation is simply too computationally demanding for both people and computers (Simon, 1997; Van Rooij, 2008). Optimal decision strategies are instead defined as decision procedures that achieve the highest possible level of *resource-rationality* (Lieder & Griffiths, 2020b), which is defined as the expected utility of the choices that a given heuristic will make when a person uses it in a given environment minus the opportunity

cost of the time and mental resources it expends to reach those decisions.

When faced with a new decision, people generally cannot compute the resource-rational decision strategy for that scenario themselves, and doing so would be more demanding than computing the optimal decision (Lieder & Griffiths, 2020b, a; Rich et al., 2020). However, many resource-rational strategies can be executed with minimal effort once they have been learned (Callaway et al., 2022a; He et al., 2021; He & Lieder, 2022; Lieder & Griffiths, 2020b, a). Moreover, people can learn to efficiently detect which of their strategies is best suited for the decision they are facing (Lieder & Griffiths, 2017). This makes deriving resource-rational strategies for specific decisions people frequently face and conveying them to people a promising approach to improving human decision-making (Callaway et al., 2022a). Researchers interested in improving human decision-making can derive resource-rational strategies by leveraging artificial intelligence to compute the optimal policies of metalevel Markov Decision Processes (Lieder et al., 2017; Callaway et al., 2018a; Griffiths et al., 2019). Having solved such a problem, the result is a function for choosing the next planning operation according to information revealed by previous planning operations, called a *metalevel policy*.

Formally, AI-powered boosting (Callaway et al., 2022a) proceeds by modeling the problem of selecting an optimal sequence of cognitive operations as a Markov Decision Process (Sutton & Barto, 2018), leverages reinforcement learning to derive the optimal policy for selecting planning operations from that model, and then conveys this policy to people. There are two different approaches for doing so: via intelligent cognitive tutors (Callaway et al., 2022a; Consul et al., 2022) and via AI-generated decision aids (Skirzyński et al., 2021a) (see Fig. 1). Intelligent cognitive tutors train people to internalize the metalevel policy so that they can subsequently apply the automatically discovered decision strategy independently without further assistance. By contrast, AI-generated decision aids display a description of the automatically discovered decision strategy while people make their decisions.

AI-powered boosting has been shown to be a promising approach for improving human planning skills (Callaway et al., 2022a). The intelligent tutors presented in Callaway et al. (2022a) taught people how to efficiently plan in the environment shown in Fig. 3. After each planning operation performed by participants (i.e., a click uncovering a number), the tutor either praised them for following the optimal planning strategy, or penalized them by forcing to wait an amount of time proportional to the suboptimality of their planning operation. This procedure significantly improved people's planning skills when they interacted with a different, transfer environment. Subsequent work employed intelligent cognitive tutors that teach people by showing
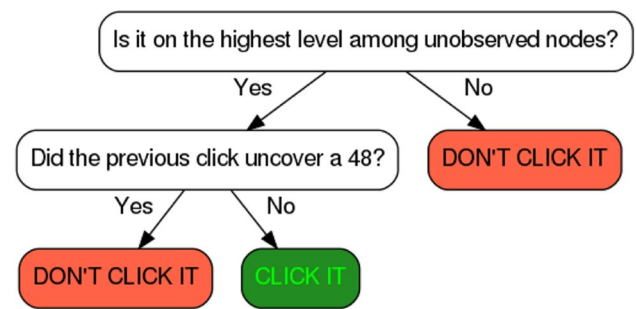

Fig. 4 Sample flowchart obtained via the AI-powered boosting intervention from Skirzyński et al. (2021a). The flowchart expresses a farsighted planning strategy in the Mouselab-MDP task with increasing variance

video demonstrations of the automatically discovered strategy (Consul et al., 2022; Mehta et al., 2022). Consul et al. (2022) developed intelligent cognitive tutors that improved people's planning in very large versions of the environment illustrated in Fig. 3.

Here, we extend and evaluate a recent AI-powered boosting method that conveys automatically discovered strategies through automatically designed decision aids (Skirzyński et al., 2021a). The original version of this method comprises four steps.

The first two steps are the same as in Callaway et al. (2022a): the problem of selecting an optimal sequence of planning operations is modeled as a Markov Decision Process (MDP), and then reinforcement learning algorithms are utilized to find the optimal strategy from this model. This is where the first input to AI-powered boosting mentioned in the Introduction is required: the model of the decision problem. Here, we model decision problems in the Mouselab-MDP framework (Callaway et al., 2017, 2022b). This is a very general framework for modeling problems which involve sequentially processing, integrating, and selecting multiple pieces of information. This makes our approach applicable to a wide range of decision problems.

In the third step, a set of logical primitives is created, and an imitation learning algorithm called AI-Interpret uses these primitives to generate an interpretable description of the optimal strategy as a logical formula. Importantly, the formula generated by AI-Interpret is expressed in conjunctive normal form, that is a disjunction of conjunctions, which can be naturally expressed as a decision tree. Due to that, after the outputted description is automatically translated to natural language via a predefined primitives' dictionary in the fourth step, it is naturally formed into a static flowchart (see Fig. 4). This flowchart evaluates to 1 (perform) or 0 (do not perform) for each action and each possible state people may encounter, and guides people through decision-making. The fourth step also introduces the second input to AI-powered boosting mentioned in the introduction: a dictionary

of the relevant problem components. Our previous work (Skirzyński et al., 2021a) introduced a set of logical predicates capable of representing multiple planning strategies (Skirzyński et al., 2021b) used in the Mouselab-MDP, and hence in multiple planning problems that can be represented in this formalism. By translating these predicates into the elements of the problem (e.g. graph-theoretic is_leaf predicate could mean long-term consequences of actions), our approach generates problem-specific natural language instructions instead of logical formulas. Like previously, successfully fulfilling this step relies on the researcher's knowledge of the problem and understanding of the Mouselab-MDP paradigm. For more complex problems, such as chess playing, it is also possible that new predicates need to be created and then translated.

Still, despite the success of AI-powered boosting in improving human decision-making through static descriptions (Skirzyński et al., 2021a), we hypothesized that more naturalistic tasks may require a more easily interpretable representation of the decision strategy. In the next section, we present an experiment suggesting that procedural instructions are more effective than the decision aids constructed by the original version of boosting presented by Skirzyński et al. (2021a).

Experiment 1: Procedural Descriptions of Planning Strategies Are More Interpretable Than Static Descriptions

A key challenge of AI-powered boosting is to convey an abstract and potentially complex planning strategy to people. To support people effectively, the description of the strategy should be easy to understand. The description presented by the decision aid can either be *static* or *procedural*. We call a family of descriptions that express a (planning) strategy by listing conditions under which certain actions can be taken *static* descriptions. A decision tree is one example of a static description, and a decision set (a set of rules) is another example. In both cases, a sequence of conditions that describe the environment tell the decision-maker when it is allowed to take which actions. A decision tree is the most extreme instance, since the conditions may be manifold, but specific actions have to be always coded inside the tree, whereas the actual actions either perform (1) or do not perform (0) the planning operation under consideration (see Section 2.4). In contrast to static descriptions, one can define *procedural* descriptions that express (planning) strategies in terms of an ordered list of actions that need to be taken sequentially according to the order. A program is a primary example of a procedural description where the subroutines specify the actions, and their positioning in the program (top to bottom) defines the ordering.

The intervention introduced in Skirzyński et al. (2021a) made an important hidden assumption about the preferred way of conveying planning strategies to people, choosing static descriptions as the output. However, a more intuitive assumption is that procedural descriptions facilitate teaching planning strategies to a greater extent. If this were the case, then it might be possible to make AI-powered boosting substantially more effective by generating procedural descriptions rather than static descriptions. To find out if it would be worthwhile to develop algorithms for generating procedural instructions, we first tested whether procedural descriptions of planning strategies are more interpretable than static descriptions.

To do so, Experiment 1 was designed to determine if participants are more successful in applying a planning strategy when this strategy is expressed in terms of procedural instructions or the static descriptions used by Skirzyński et al. (2021a). To answer this question, participants in Experiment 1 received decision aid on how to solve a sequential decision problem. One half of the participants received the static descriptions generated by the original version of the method (Skirzyński et al., 2021a) and the other half received hand-crafted procedural instructions. The main goal of Experiment 1 was to compare people's ability to follow procedural versus static instructions. We therefore asked participants to execute the described strategy as accurately as possible. As a result, group differences in the task performance should result primarily from differences in the comprehensibility of static versus procedural descriptions.

Methods

Task

The template for the experimental task was the Mouselab-MDP paradigm (Callaway et al., 2017, 2022b). In this task, participants have to choose between six possible paths, each of which involves a series of 3 steps. People can gather information about how much reward they would receive for visiting a location by clicking on it for a small fee. The objective is to balance expenses while finding a rewarding path. Skirzyński et al. (2021a) demonstrated that automatically generated static descriptions of planning strategies can boost participants' performance across multiple versions of that task. We adopted the experiment with a static description for the most demanding strategy for our purposes.

To create an unbiased comparison of static descriptions and procedural instructions, we created an experiment where the participants' only task is to execute the described strategy. Participants were instructed to select planning operations (i.e., clicks) according to the described strategy. To minimize the risk that participants misunderstood the task as collecting rewards, participants were not informed that
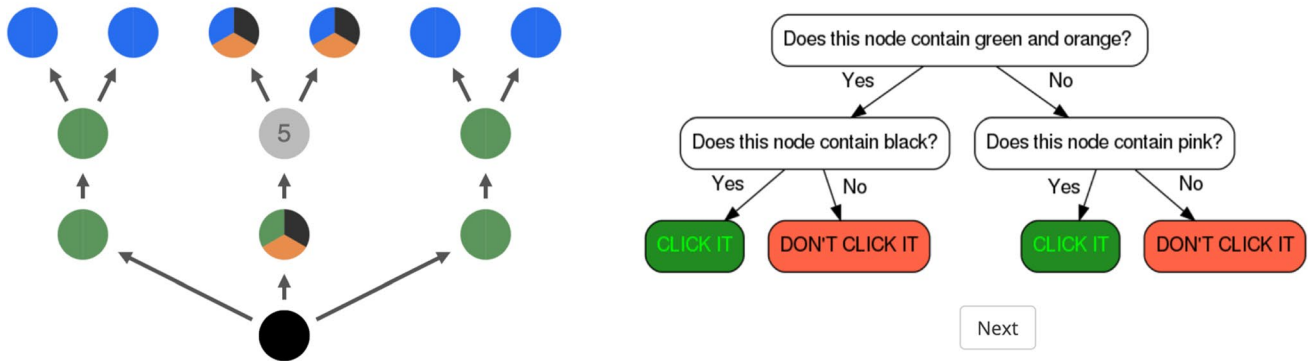
Fig. 5 Experiment 1: The experimental screen as shown to participants in the flowchart condition. On the right, the Mouselab-MDP task with color-coded nodes. Each color represents a node property, which can change depending on the uncovered values. On the right, the color-coded flowchart depicting a specific clicking procedure. In the alternative condition the flowchart is replaced with procedural instructions

the numbers correspond to rewards and could neither pick a path nor collect any rewards. This resulted in a sequential clicking task in which participants' only objective was to click nodes according to presented instructions. The original static descriptions described the locations to click based on attributes derived from graph theory, for example whether a location is on the current most rewarding path and whether a location can be found at a certain depth of the graph. These kinds of descriptions can be difficult to understand for laymen and usually require training. To simplify the procedure, each attribute was visually presented as a color. Based on the uncovered rewards, we computed if an attribute applied to a location and, if so, the location was marked with the corresponding color (see Fig. 5). One location could be tagged with multiple colors. The attributes used in the static descriptions and in the procedural instructions were replaced by the corresponding color names. The manipulations described had two main advantages. First, participants were not tempted to neglect the clicking instructions and apply their own clicking strategy, as we concealed the original planning task. Second, the instructions required only knowing the colors, giving very little room for misinterpretations compared to more complex attributes in the original task. In sum, this manipulation eliminated possible limitations of the original task and let us compare the degree of interpretability of both static descriptions and procedural instructions more directly. The environment participants interacted with was the color-coded Mouselab-MDP and was displayed on the left of the experiment's screen, whereas the instructions appeared on the right of the screen (see Fig. 5). The static descriptions were the same as the ones used in Skirzyński et al. (2021a) except that the attributes were

now color-coded. The procedural instructions were hand-crafted and described the same clicking strategy as the static descriptions. The instructions were as follows: "Click the nodes that contain green and orange until you find a +10. Then click the nodes that contain blue and pink.".

Dependent Variables

To measure whether a participant understood the instructions, we counted the number of clicks that were consistent versus inconsistent with the strategy they were instructed to follow. Both measures were combined into one metric called click agreement, which we defined as the proportion of consistent clicks out of all performed clicks, that is

$$\text{agreement} = \frac{n_{\text{consistent}}}{n_{\text{consistent}} + n_{\text{inconsistent}}}. \tag{1}$$

In the event that a participant finished the trial before having clicked all the instructed locations, the expected number of missed clicks counted toward the inconsistent clicks. We calculated the number of missed click as the average number that the instructed strategy performed in 1000 simulations minus the number of clicks performed by the participant. Click agreement is reported in percent. In addition, we measured the expected value of the score participants would have received if they had performed their clicks in the Mouselab-MDP task. By definition, the expected score is the sum of rewards along the best path identified by the participant's clicks minus the cost of those clicks. Because the expected value of unrevealed rewards was zero, the expected reward of a path connecting the start node was equal to the sum of

Fig. 6 Experiment 1: **a** Procedural instructions are more interpretable than static descriptions (*p*=.018). The plot shows the mean click agreement of participants of the experimental conditions (blue and red) and a control condition (gray). **b** Participants assisted by procedural instructions had higher expected scores than participants assisted by static descriptions (*p*=.09). The expected score that participants would have received if they had performed their clicks in the Mouselab-MDP task
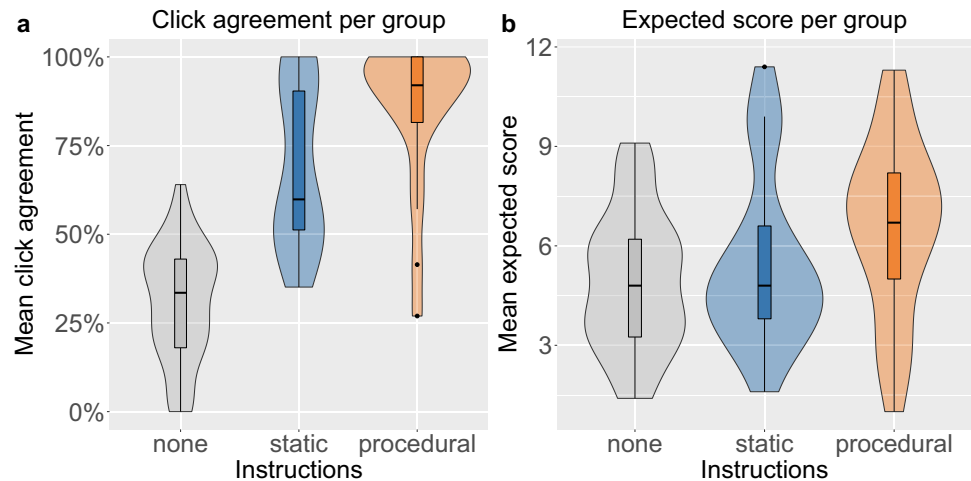
the rewards revealed on that path. The cost of the participant's clicks was 1 point per click.

Procedure

Participants were randomly assigned to a static descriptions or a procedural instructions condition. Both conditions started with an instructions block, in which the task was introduced and motivated as a pass-code test required to enter an extraterrestrial planet that can only be accessed by following specific instructions. This was followed by an attention check consisting of 3 multiple-choice questions, a block explaining the procedure and the instructions (see Appendix 1) and a final attention check consisting of 3 multiple-choice questions. After this, participants engaged in a single practice trial in which they were instructed to click locations marked with orange. They received feedback on the correctness of their clicks and when to end the trial. Lastly, there was a block of 10 test trials and a small demographic survey. The minimum time required to spend on a trial was ten seconds. Participants were informed that they would start with a bonus of $2 and lose 20 cents for each trial in which they made an inconsistent click or quit early. Neither their current amount of bonus nor the score was displayed in the task. In addition, all participants received a base payment of $1.

Participants

We recruited 21 people for the static descriptions condition and 21 people for the procedural instructions condition on Amazon Mechanical Turk (average age: 35.9 years, range: 18–65 years; 21 female). The experiment lasted 11.9 min on average. Our predefined exclusion criterion, which excludes individuals who do not perform any click in half of the test trials, did not apply to any participant.

Results

The mean click agreement was 68.8% (Median: 59.8%, *SD*=22.9%) in the static descriptions condition and 85.2% (Median: 92.0%, *SD*=20.5%) in the procedural instructions condition, as illustrated in Fig. 6a. The variable was not normally distributed; we thus employed a one-sided Mann-Whitney-U test. We found that click agreement was significantly higher in the procedural instructions condition (*A* = .69;[1] *U*=137, *p*=.018). Click agreement and expected score were positively correlated (*r*(40)=.45 *p*=.003). Accordingly, the mean expected score was higher in the procedural instructions condition (*M*= 6.6, *SD*=2.7; see Fig. 6b) than in the static descriptions condition (*M*= 5.5, *SD*=2.6), although this difference was not statistically significant (*d*=0.41; *t*(40)=-1.3, *p*=.09). The learning curves for click agreement and expected score over the ten trials can be found in Appendix 2 (see Fig. 10). Furthermore, participants assisted by procedural instructions were able to reach their decisions significantly faster than participants assisted by static descriptions (21.0 s vs. 31.6 s; *d*=0.95, *t*(40)=3.1, *p*=.002). Moreover, we compared the performance of the two experimental groups to a control group (*N* = 60) that was instructed to maximize their score in the Mouselab-MDP task without being assisted by a decision aid.[2] The click agreement in both, the static descriptions condition (*A* = .93; *U*=93, *p*=<.001) and the procedural instructions condition (*A* = .95; *U*=59, *p*=<.001) was

---

[1] For non-parametric tests, we report the common language effect size *A*, which describes the probability that a randomly chosen member of group 1 scores higher on the dependent variable than a randomly chosen member of group 2 (Ruscio 2008)

[2] The control condition was part of Experiment 3 of our previous work (Skirzyński et al. 2021a, b)

significantly larger than in the control condition ($M=$ 31%, $SD=$16%). This also applied to the expected score which was significantly larger in the static descriptions condition ($d=0.49$, $t(79)=1.9$, $p=.027$) and the procedural instructions condition ($d=0.9$, $t(79)=3.6$, $p=.<001$) than in the control condition ($M=$ 4.2, $SD=$2.6).

Discussion

The experiment was designed to compare people's ability to follow procedural versus static instructions on how to make a plan. We deliberately reduced the potentially confounding effect of participants' general decision-making competency, domain knowledge, insights into the specific decision problem, and opinions about which strategy might be best. To achieve this, we chose a highly abstract task and instructed all participants to follow the instructions as accurately as possible. We can therefore interpret the observed differences in task performance primarily in terms of the comprehensibility of the instructions. In accordance with our hypothesis, we found that procedural instructions were more helpful for people than static descriptions. Even though the final expected score obtained in the experiment did not differ significantly between the conditions, the group which utilized the procedural instructions was both faster and numerically more accurate in applying the described strategy. The main reasons why the noticeable numerical difference in accuracy was not statistically significant might be that our sample size was rather small relative to the high variance of the rewards in this particular task. The comparison to a control group showed that both decision aids significantly improved participants' adherence to the instructed strategy and the corresponding gain in expected score. One possible explanation for the results we observed is that static descriptions underutilize people's capacity to comprehend and execute structured, abstract procedures (Miller et al., 1960). This interpretation suggests that the static descriptions were unnecessarily detailed, long, and repetitive.

We acknowledge that the optimal strategies for some difficult problems could be so complex that they cannot be approximated well by any verbal description. Apart from those cases, the results of Experiment 1 should also hold for more complex tasks because the difference in the amount of effort required to follow static versus procedural instructions would be even larger for larger tasks. People's increased compliance with the near-optimal strategy and their faster decisions found in Experiment 1 thus suggest that it is worthwhile to extend our AI-based boosting method to procedural instructions because people appear to be much more willing and able to follow them.

AI-Powered Boosting with Procedural Descriptions of Optimal Decision Strategies

As mentioned in Section 2.4, the original version of AI-powered boosting (Skirzyński et al., 2021a) generates static descriptions in form of a flowchart (decision tree) for verifying whether a candidate planning operation is consistent with an optimal planning strategy. A key limitation of this approach is that it does not explicitly tell people which planning operations to perform but requires them to come up with a good planning operation themselves and then verify their hypothesis. Motivated by the results of Experiment 1, we now present an algorithm for transforming the output of AI-Interpret into a procedural description of how to plan that explicitly states which planning operation should be performed first, second, third, and so on. By coupling AI-Interpret with this new algorithm, we obtain a general method for describing any RL policy through procedural instructions. Our method is very general because AI-Interpret is a policy-agnostic method that only utilizes demonstrations of the policy to describe it. It therefore contributes not only to the field of boosting human decision-making but also to the field of explainable reinforcement learning (Puiutta & Veith, 2020; Dazeley et al., 2021). Notably, both AI-Interpret (Skirzyński et al., 2021a) and our new algorithm compute approximations to the optimal policy and the flowchart, respectively. AI-powered boosting hence generates lossy simplifications of the policies computed by dynamic programming or reinforcement learning. However, those simplifications are not only easier to grasp than static descriptions (see Experiment 1), but they also achieve their objective to improve human decision-making, as will we show in the following sections.

Later in this article, we will focus on the application of our new method to AI-powered boosting rather than standard RL tasks. This section hence presents our two major technical contributions. In the first subsection, we detail DNF2LTL — our algorithm for transforming disjunctive normal form logical formulas into procedural descriptions. In the second subsection, we present AI-powered boosting extended with this algorithm.

Generating Procedural Descriptions of Planning Strategies

The original output of AI-Interpret that is utilized by AI-powered boosting is a Disjunctive Normal Form formula (DNF; see Definition 9). Our algorithm, which we call DNF2LTL, transforms such formulas into the procedural format of Linear Temporal Logic (LTL; see Definition 5).

DNF2LTL operates in two phases. In the first phase, it modifies the input disjunctive normal form formula into an entity called a *procedural formula* (see Definition 8), that is an expression in a specific from of linear temporal logic. In the second phase, the procedural formula is pruned to remove as many unnecessary predicates as possible to obtain the possibly simplest procedural description of the automatically discovered strategy. We formally define the output of our algorithm in the next section.

Procedural Formulas

Procedural formulas generated by our algorithm are special cases of Linear Temporal Logic (LTL) formulas extended with two additional operators.

First, LTL itself is a type of propositional logic that allows expressing processes that change in time (see Definition 5).

Definition 5 (**Linear Temporal Logic**) Let $\mathcal{P}$ be the set of propositional variables p (variables that can be either true or false), let $\neg, \wedge, \vee$ be standard logical operators for negation, AND, and OR, respectively, and let $\mathbf{X}, \mathbf{U}, \mathbf{W}$ be modal operators for NEXT, UNTIL, and UNLESS, respectively. Linear temporal logic (LTL) is a logic defined on (potentially infinite) sequences of truth-assignments of propositional variables. LTL formulas are expressions that state which of the variables are true, and when they are true in the sequence. Whenever this agrees with the actual truth-assignment in an input sequence, then we say that a formula is true.

Formally, for α and β being LTL formulas, we define a formula to be expressed in LTL inductively: ψ is an LTL formula if $\psi \in \mathcal{P}$ (ψ states that one of the variables is true in the first truth-assignment in the sequence), $\psi = \neg\alpha$ (ψ is a negation of an LTL formula), $\psi = \alpha \vee \beta$ (ψ is a disjunction of two LTL formulas), $\psi = \alpha \wedge \beta$ (ψ is a conjunction of two LTL formulas), $\psi = \mathbf{X}\alpha$ (ψ states that LTL formula α is true starting from the next truth-assignment in the sequence) or $\psi = \alpha\mathbf{U}\beta$ (ψ states that LTL formula α is true until some truth-assignment in the sequence where LTL formula β becomes true).

We extend the standard definition of LTL to allow more natural transition from decision trees to procedural instructions that, as we found in Experiment 1, are easier for people to follow. To do so, we add a Hold operator that allows introducing a default stopping condition (see Definition 6), and a Loop operator that defines which part of the procedure to repeat (see Definition 7).

Definition 6 (**Hold modal operator**) The HOLD operator $\mathbf{H}$ is a unary operator in the linear temporal logic. LTL formula $\mathbf{H}\psi$ states that ψ is true at least for the first truth-assignment in the sequence of truth-assignments, and then eventually

becomes false. HOLD operator is the UNTIL operator with a default until condition ("until it is no longer satisfied").

Definition 7 (**Loop modal operator**) The LOOP operator $\mathbf{L}$ is a binary operator in the linear temporal logic. LTL formula $\phi\mathbf{L}\psi$ states that(i) part of ϕ is an LTL formula ψ, (ii) if ψ is replaced with some number of NEXT ψ operators $\mathbf{X}\psi \wedge \cdots \wedge \mathbf{X}\psi$, then the new ϕ is true across the whole sequence of truth-assignments. In other words, the LOOP operator states that in order for LTL formula ϕ to be true and satisfy all truth-assignments in the sequence, LTL formula ψ inside of ϕ needs to be repeated the appropriate number of times (form a "loop"). This formula corresponds to truth-assignments in the sequence that disagree with ϕ.

Finally, by adding the introduced operators to the LTL formalism, we obtain procedural formulas (see Definition 8).

Definition 8 (**Procedural formula**) We say that f is a procedural formula if and only if f is an expression written in linear temporal logic where the propositional variables are predicates $h : \mathcal{S} \times \mathcal{A} \to \{0, 1\}$ for some set of states $\mathcal{S}$ and some set of actions $\mathcal{A}$, and where the modal operators are $\mathbf{X}$ (NEXT), $\mathbf{U}$ (UNTIL), $\mathbf{W}$ (UNLESS), $\mathbf{H}$ (HOLD), and $\mathbf{L}$ (LOOP).

Transforming Disjunctive Normal Form Formulas into Procedural Formulas

In the first phase, DNF2LTL generates a procedural formula (see the previous section) out of a DNF formula (see Definition 9).

Definition 9 (**Disjunctive Normal Form**) Let $f_i, h : \mathcal{X} \to \{0, 1\}$ for $i \in \mathbb{N}$ be binary-valued functions (predicates) on domain $\mathcal{X}$. We call $f_1(\boldsymbol{x}) \vee f_2(\boldsymbol{x}) \vee \cdots \vee f_n(\boldsymbol{x})$ a disjunction of predicates and $f_1(\boldsymbol{x}) \wedge f_2(\boldsymbol{x}) \wedge \cdots \wedge f_n(\boldsymbol{x})$ a conjunction of predicates. We say that h is in disjunctive normal form (DNF) if h is a conjunction of disjunctions of predicates f_i.

To do so, our algorithm accepts four main inputs: the set of trajectories that led to the creation of the DNF formula, a set of predicates that could serve as the until or unless conditions, a set of predicates which are unwanted in the procedural formula, and, naturally, the DNF formula itself.

Definition 10 (**Trajectory**) A trajectory $\tau = [\phi_0, \ldots, \phi_{N-1}]$ is a sequence of N state-action pairs $\phi_i = (s_i, a_i)$ with $s_i \in \mathcal{S}, a_i \in \mathcal{A}, i = 0, \ldots, N-1$.

The trajectories (see Definition 10) play the role of the sequences of truth-assignments from Definition 5, whereas

the set of predicates for until/unless conditions and the DNF formula define the building blocks out of which the procedural formula would be constructed. The other remaining parameter is optional, and in case of a failure in producing an output, the algorithm is ran again without removing the redundant predicates. On a high level, our algorithm exploits the idea that a DNF formula is satisfied when at least one of its conjunctions is satisfied. It iterates over the trajectories to discover the dynamics of changes in truth values of the conjunctions, and uses the conjunctions, the found dynamics, and the candidate until/unless conditions to generate procedural formulas.

During the first phase, DNF2LTL generates an initial procedural description in four steps. In the first step, the algorithm extracts potential subroutines from the inputted DNF formula. In the second step, the algorithm determines the order in which those subroutines should be performed. In the third step, the algorithm computes the logical conditions for transitioning from each step to the next. Finally, in the fourth step, our method connects the subroutines with the appropriate conditions into a complete procedural description and outputs the result. Algorithm 1 presents a pseudocode that implements the first phase of DNF2LTL and the following paragraph provides a technical description of each of these four steps in greater detail. We relate this description to the pseudocode by listing its relevant line numbers in brackets. Readers who are primarily interested in the big picture and the application to boosting human planning can skip these technical details.

Step 1: DNF2LTL starts by dividing the DNF formula into a set of conjunctions and removing all the unwanted predicates [Lines 3–4].

Step 2: Then, it iterates over the trajectories and for each trajectory records the sequence of conjunctions that were true for that trajectory so that the whole DNF formula could be true across all the state-action pairs within it. Our algorithm then creates a transition graph where conjunction c_i is connected with conjunction c_j if there is at least one trajectory τ where the value of c_i changed from true to false at the same moment when the value of c_j changed from false to true [Line 9]. The transition graph is used to generate maximum length sequences of conjunctions $c_{i_1} c_{i_2} \ldots c_{i_n}$ to capture the possibly fullest transition evidenced in the data [Line 10]. The last predicate in this sequence (i.e., c_{i_n}) either has no outgoing connections in the transition graph or connects to one of the c_{i_j}s in which case the sequence ends with a special loop symbol

that indicates which i_j that is. The resulting maximum length sequences are used to define equivalence classes for the trajectories. These equivalence classes represent potential dynamics of how the conjunctions change their truth values so that the full DNF formula was satisfied. Each trajectory, treated as a sequence of conjunctions of the DNF formula, is then assigned to a number of equivalence classes. Namely, trajectory τ represented by sequence s is assigned to all equivalence classes e for which s is a subsequence of e. For instance if τ is represented by sequence $c_1 c_3$, it could be assigned to equivalence class $c_1 c_2 c_3 c_4$ LOOP c_2 [Line 11]. This whole process in Step 2 is performed to generate candidates for procedural descriptions.

Step 3: Then, the algorithm transforms unempty equivalence classes into procedural formulas. It does so by using the trajectories in the class [Line 13] to iteratively find UNTIL operators (UNTIL conditions) that could separate each of the elements in the sequence representing the class. During one iteration, DNF2LTL searches for the UNTIL condition separating a pair of subsequent conjunctions. Possible candidates for UNTIL conditions are the allowed predicates provided as an input to the method and 2-element disjunctions of those predicates [Input P], i.e. we hand-engineer possible operators a priori. For a pair of subsequent conjunctions $c_i c_{i+1}$, the matching conditions are such whose truth value changes from constantly false, while c_i is true, to true, when c_{i+1} true. We select the UNTIL condition among matching conditions as the u_i that maximizes the likelihood of the trajectories in the equivalence class under c_1 UNTIL u_1 AND NEXT $\ldots c_i$ UNTIL u_i AND NEXT TRUE, i.e. the formula generated so far [Lines 21–24]. This process allows us to select a condition that we know is appropriate (belongs to input P), and that is the most likely under the data. If there is no matching condition, the algorithm adds the default UNTIL condition — the HOLD operator or, if some predicates were removed from the formula, tries again with the original formula [Lines 25–29]. If some trajectories have a conjunction representation shorter than the representation of the class, the algorithm also adds an UNLESS operator after the UNTIL operator, and searches for the UNLESS condition in a similar way. If there are no matching conditions in input P, *FALSE* is selected as the condition so that to allow excessive planning and prevent UNLESS to be met. This process models

Input: DNF formula f;
        Set of N trajectories $D = \{((s_{1j}, a_{1j}))_{j=1}^{l_1}, \ldots, ((s_{Nj}, a_{Nj}))_{j=1}^{l_N}]\}$;
        Unless/until predicates P;
        Redundant predicates R;
Output: Procedural formula ψ;
1: Transform the trajectories in D into a matrix of predicate truth-assignments M_D
2: $L \leftarrow$ lengths of trajectories in D
3: Transform f to a list of m conjunctions $C = [p_{11} \wedge \ldots \wedge p_{1n_1}, \ldots, p_{m1} \wedge \ldots \wedge p_{mn_m}]$.
4: Remove redundant predicates R from C.
5: **if** Nothing was removed and $len(C) == 1$ **then**
6:     $\xi = C[0]$
7:     **return** $\psi = \mathbf{H}\ \xi$
8: **end if**
9: $G \leftarrow$ graph of connections between $c \in C$ based on D
10: $S \leftarrow$ max length sequences from G
11: $E \leftarrow$ equivalence classes from S, D
12: **for** $e \in E$ **do**
13:     $e_{trajs} \leftarrow$ Trajectories from e.
14:     $\beta_{prev} = \emptyset$
15:     **for** β in e **do**
16:         **if** *loop* in β **then**
17:             Use the special *loop* symbol in β to determine the conjunction to return to γ.
18:             $\phi = \phi\ \mathbf{L}\ \gamma$
19:             Go to step 29
20:         **end if**
21:         $\Theta \leftarrow$ possible until conditions found with $e_{trajs}, M_D, \beta, \beta_{prev}$.
22:         **if** success **then**
23:             $\theta^* \leftarrow \arg\max_{\theta \in \Theta} \text{likelihood}(e_{trajs}, \phi \wedge \mathbf{N}\ \beta\ \mathbf{U}\ \theta)$
24:             $\phi = \beta\ \mathbf{U}\ \theta$
25:         **else if** nothing was removed from C **then**
26:             $\phi = \mathbf{H}\ \beta$
27:         **else**
28:             Run the algorithm with $R = \emptyset$
29:         **end if**
30:         Determine if $\exists \tau \in e_{trajs} : len(\tau) < len(e)$ and the unless operator is necessary.
31:         **if** necessary **then**
32:             $\Theta \leftarrow$ possible unless conditions found with e_{trajs}, M_D, β.
33:             **if** success **then**
34:                 $\theta^* \leftarrow \arg\max_{\theta \in \Theta} \text{likelihood}(e_{trajs}, \phi\ \mathbf{W}\ \theta)$
35:                 $\phi = \phi\ \mathbf{W}\ \theta$
36:             **else if** nothing was removed from C **then**
37:                 $\phi = \phi\ \mathbf{W}\ \textbf{false}$
38:             **else**
39:                 Run the algorithm with $R = \emptyset$
40:             **end if**
41:         **end if**
42:         **if** β is not last in e **then**
43:             $\phi = \phi \wedge \mathbf{N}$
44:         **end if**
45:     **end for**
46:     **if** β is the first element in e **then**
47:         $\psi = \phi$
48:     **else**
49:         $\psi = \psi \vee \phi$
50:     **end if**
51:     $\beta_{prev} = \beta$;
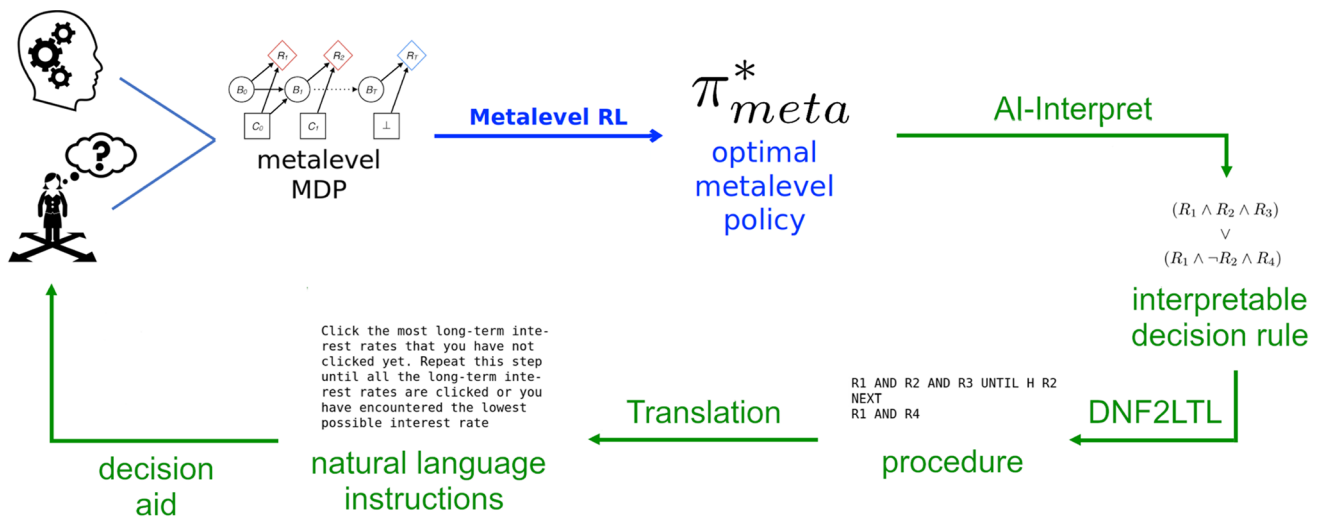52: **end for**
53: **return** ψ

Fig. 7 AI-powered boosting with decision aids conveys automatically found planning strategies through automatically generated procedural instructions. Automation in the former area is achieved by modeling the problem as a metalevel MDP and solving it with metalevel reinforcement learning. Automation in the latter area rests on (i) utilizing an imitation learning algorithm AI-Interpret that constructs a DNF formula of predefined predicates to describe the strategy, (ii) applying the DNF2LTL method that transforms the DNF into a formula in LTL, (iii) translating the output into natural language using a predefined predicate-to-expression dictionary

a situation when the formula allows early stopping [Lines 29–40]. Having set the condition(s), the generated LTL subformula is attached to the formula built so far via the NEXT operator [Lines 42–44]. The algorithm then iterates [Line 15]. If the last pair of conjunctions from the sequence representing the equivalence class contains a conjunction and a loop symbol, this symbol is transformed into the LOOP operator (see Definition 7) and the conjunction and the loop operator are joined through the NEXT operator [Lines 15–20]. If there are demonstrations that end before the loop, the UNLESS operator is added in the same way as before [Line 19].

Step 4: After generating the procedural formulas for each of the equivalence classes, the final procedural formula is returned as a disjunction of these formulas [Line 49]. Note, however, that only one of the elements in the disjunction is returned by DNF2LTL after it performs pruning (see below).

Our algorithm captures a special type of procedural formulas. For a DNF formula with only one conjunction, the structure of the output can be described by the regular expression

$$[\mathbf{H}\ \Phi \wedge \mathbf{N}\ |\ \Phi\ \mathbf{U}\ P\ (\mathbf{W}\ P) \wedge \mathbf{N}]^{+}[\mathbf{H}\ \Phi\ |\ \Phi\ \mathbf{U}\ P\ |\ \mathbf{L}\ P](\mathbf{W}\ P)$$

(2)

where P may be substituted with either of the input allowed predicates or their 2-element disjunction, and Φ may be substituted with an arbitrary conjunction of those predicates. The expression given in Eq. 2 thus generates procedural formulas in the form of a sequence of NEXT operators, where subsequent conjunctions are separated with UNTIL conditions (and/or UNLESS conditions) or accompanied by the HOLD operator. The formula ends with the last NEXT operator or with a LOOP operator.

Pruning

After our algorithm generates a procedural formula Ψ in the first phase, it enters the second phase. During the second phase, DNF2LTL prunes the predicates appearing in the conjunctions of Ψ. Recall, however, Ψ is a disjunction of procedural formulas. Because of this reason, pruning occurs for each element of that disjunction separately. To do so, DNF2LTL maps each procedural formula ψ_i of the disjunction Ψ onto a distinct binary vector b_i. Each element of b_i is the truth value of one of the predicates appearing in the conjunctions making up psi_i. Our algorithm iterates over b_is and in each step performs a greedy optimization. Concretely, for each consecutive predicate of ψ_i the corresponding entry of b_i is set to zero if and only if removing that predicate increases the likelihood of the trajectories under the pruned description relative to the unpruned description. Some predicates increase the likelihood and are consequently pruned. After performing this optimization for each b_i (and ψ_i), the algorithm outputs the pruned ψ_i for which the likelihood was the highest as the final procedural description.

Extending AI-Powered Boosting to Procedural Descriptions

Following the result on human preference toward procedural descriptions of planning strategies, we extended the vanilla AI-powered boosting method by adding our DNF2LTL algorithm (see Fig. 7). Instead of generating static, flowchart descriptions of optimal decision strategies, our extended version performs AI-powered boosting with procedural descriptions of optimal decision strategies. Originally, AI-powered boosting models the decision problem theoretically, finds the optimal policy for that model and relies on the AI-Interpret algorithm to find a static description of this policy. Lastly, it translates this description to natural language (see Section 2.4). In the new version of AI-powered boosting, we transform the description generated by AI-interpret into a program-like procedural description of logical primitives using DNF2LTL. We translate these descriptions into natural language instructions only after this transformation. In total, our new method comprises five steps: (1) modeling the planning problem, (2) finding the optimal strategy through that model, (3) creating a description of that strategy in form of a logical formula, (4) changing the formula to a procedural formula, (5) translating the procedural formula to natural language instructions.

To evaluate our extension of the general AI-powered boosting method, we applied it to discover and teach the optimal planning strategy for the three-step planning task illustrated in Fig. 3. This planning task presents a version of the Mouselab-MDP paradigm (see Section 2.2) created by Lieder et al. (2019) where the variance of the possible rewards is small in the first step (immediate consequences), becomes slightly larger in the second step (short-term consequences), and much larger in the third step (long-term consequences). In this scenario, people often neglect inspecting the long-term consequences of their actions (Jain et al., 2021; Callaway et al., 2022b; Lieder et al., 2019). By contrast, the optimal planning strategy for this environment takes a far-sighted approach to decision-making that would be beneficial in the real world. We hypothesized that our extended AI-powered boosting approach may represent this strategy in a number of more naturalistic decision problems, and help people improve their decision-making therein.

In applying the extended AI-boosting method to new problems, we completed steps 1 to 4 using the same methodology and the same parameters as in Skirzyński et al. (2021a), but focused specifically on the Mouselab-MDP with increasing variance structure. In step 4 we selected candidate predicates for the until and unless conditions using the domain-specific language introduced in Skirzyński et al. (2021a). In step 5 we adapted the predicate-dictionary introduced in Skirzyński et al. (2021a) to work on procedural descriptions and be domain-specific, depending on the

problem that AI-powered boosting is to provide decision support on. Concretely, the procedural description is created by separating the formula into steps, where each step is a part of the formula between two NEXT operators (or the part between the beginning of the formula and the first NEXT operator or the part between the last NEXT operator and the end of the formula). The steps are then separately translated according to the alignment of LTL operators and predicates in the step, using the mentioned domain-specific dictionary. The whole description is returned as an enumeration of those translations.

Concretely, each step is translated according to the following logic: if there is at least one non-negated predicate in a step (other than *TRUE*), the translation always begins with the template "ACT*pred*(OBJ, REW)", where ACT is substituted with a domain-specific action word, and *pred*(OBJ, REW) is substituted with a domain-specific translation of predicate *pred* that contains OBJ as the object word represented by nodes in the Mouselab-MDP, and REW as the reward word represented by numbers hidden underneath the nodes. For instance, the action word could be "*Look up*", the object word could be "*hotels*", the reward word could be "*the prices*" and the predicate could capture the property of being positioned in the last level from the start node of the Mouselab-MDP. The translation would then read "*look up the prices of the most distant hotels*".

If there are any negated predicates in a step, they are translated afterwards. The translations of those predicates themselves are listed in bulletpoints following the template "*Do not* ACT:". Special predicates include the always *TRUE* predicate, which is translated to "*Stop planning right away or* ACT *some random* OBJ *and then stop planning*" (because the available planning operations always include both clicking the nodes and terminating), and always *FALSE* predicate, which is translated using the template "*Do not* ACT *anything*" (because the always *FALSE* predicate appears in the description only on its own — for the "no-action" strategy).

For the translation of the temporal operators, when a step includes an UNTIL operator with condition *cond*(OBJ, REW), the translation template becomes "*Repeat this step until cond*(OBJ, REW)". If there is an UNLESS operator with condition *cond*(OBJ, REW) in a step, the translation of the step starts with the text "*Unless cond*(OBJ, REW), *in which case stop at the previous step*". If the there is a HOLD operator in a step, the translation of the step is added the following text "*Repeat this step as long as possible*". Finally, if there is a LOOP operator with expression EXPR in a step (which can only occur in the last step), the algorithm matches the number of the step in which expression EXPR appeared the latest, say NUM, and adds the template "*GOTO step* NUM" to the translation. The exact translation for the predicates in our DSL and the rules governing how

the steps are translated can be found in our project's repository in the `translation.py` file.

For our purposes, the final procedural formula we obtained was

$$among(not(is\_observed, has\_largest\_depth) \textbf{ U}$$
$$(are\_leaves\_observed \lor is\_previous\_observed\_max) \quad (3)$$

Although the above algorithm is carefully crafted and depends on the character of the predicates included in the step, it also contains a number of placeholders. Having filled those placeholders with predicate or operator translations from the domain-specific dictionary, it is possible to obtain strategy descriptions that accommodate for other problems. We treat Experiment 2 as a proof of concept that shows this for one class of problems, and adapt the dictionary such a way, as to have our method output instructions people understand. In more detail, the specific dictionary we used to transform this formula into task-dependent instructions was informed by pilot studies in which we tested multiple options and selected wording that resulted in the highest overall compliance. These natural language instructions are detailed in the next section, in which we assess the benefits of conveying the found strategies in behavioral experiments.

Experiment 2: Boosting Human Performance in Naturalistic Decision-making and Planning Tasks with AI-Generated Decision Aids

Our previous work (Skirzyński et al., 2021a) showed that the static descriptions generated by AI-Interpret improve the performance of individuals in the Mouselab-MDP task. However, those improvements were lower in environments which required more complex planning strategies. This was partly due to the fact that static descriptions of complex strategies are more difficult to understand. Experiment 1, which in fact tested the interpretability of the most complex strategy of Skirzyński et al. (2021a), showed that procedural instructions are easier to understand than static descriptions. In addition, the results of Experiment 1 suggest that our updated decision aids come with additional benefits over static descriptions. First, it takes less time to follow procedural instructions than to use decision aids requiring you to evaluate each planning step individually. Second, in contrast to static descriptions, procedural instructions require no introduction on how to apply them.

Equipped with these improvements, we test in this section whether decision aids generated by our extended AI-powered boosting method can enhance human performance in tasks that are more naturalistic than those used by Skirzyński et al.

(2021a). Concretely, we evaluate our approach on two naturalistic tasks. In the Road Trip task (Callaway et al., 2022b), participants are asked to plan an inexpensive trip by looking up hotel prices across cities visited during the trip (see Fig. 2a). In the Mortgage task, participants are asked to choose a mortgage based on the interest rates of the available options in the first year, the following five years, and the following 15 years, respectively (see Fig. 2b). Similar to the Mouselab-MDP task with increasing variance, the rewards in the Road Trip task vary the most at the potential final destinations. This reward structure favors far-sighted planning. This task allows us to test if AI-powered boosting can help people become more far-sighted because previous work found that human planning in this task is more short-sighted than the optimal planning strategy (Callaway et al., 2022b). The same is also true of the Mortgage task. We designed this task so that the most long-term financial consequences of choosing a mortgage are the most crucial for the total cost. It can therefore be seen as a more naturalistic measure of people's shortsightedness in intertemporal choice (O'Donoghue & Rabin, 2015; Meier & Sprenger, 2010).

To test the benefits of using our extended AI-powered boosting approach, we conducted a large-scale online experiment in which participants were presented with the Road Trip task and the Mortgage task. To measure the benefits conferred by our AI-generated decision aids, we compare the performance of people being assisted by the automatically generated decision aids that conveyed a far-sighted strategy described in the previous section against the performance of a control group making decisions without a decision aid. We did not include a condition with static instructions because Experiment 1 conclusively showed that static instructions are less effective than procedural instructions. In Experiment 2, we also overcome the main limitation of Experiment 1 by studying how beneficial AI-powered boosting might be in real-world settings where people can choose to ignore the decision aid. Thus, instead of requiring participants to comply with the recommended strategy, Experiment 2 provides people with a decision aid and allows participants to freely decide how they want to make their decisions.

Methods

Participants

We recruited 111 people on Prolific. The mean duration of the experiment was 18.1 min in the control condition and 17.9 in the experimental condition. We excluded 2 participants (1.8%; both in the experimental condition) who needed more than 3 quiz attempts in one of the two quizzes. This yielded 54 participants for the control condition (average age: 38.0 years, range: 19–74 years; 39 female) and 55

participants for the experimental condition (average age: 35.6 years, range: 18–70 years; 37 female).

Tasks

Participants were engaged in two different planning tasks. The Road Trip task (see Fig. 2a) asked people to plan a route from a starting location to one of multiple airport cities. Each city on the route requires the traveler to rent a hotel for the night. The participant's task was to efficiently find a route with low accommodation costs. To do so, the participant could look up the price of the cheapest hotel in a city for a $10 fee by typing the city name into a search engine. Hotel prices were drawn from a uniform distribution over the values {$30, $35, $40, $45} and airport hotel prices were drawn from a uniform distribution over the values {$260, $290, $320, $350, $380}. In addition, one randomly selected airport city offered a price of only $20. A route could be submitted after selecting a sequence of roads connecting the start city with an airport city.

The Mortgage task (see Fig. 2b) asked people to choose the cheapest mortgage out of three options presented in a table. Each mortgage was defined by three different interest rates: the interest rate for the fist year (2022), the interest rate for the following 5 years (2023–2027) and the interest rate for the 25 years after that (2028–2052). Three different mortgages were presented: 1. A mortgage whose interest rates increased over time. The interest rates were drawn from normal distributions with means 0.5%, 1.5% and 2.5% for the three time periods, respectively. 2. A mortgage whose interest rates stayed constant over time. The interest rates were drawn from a normal distribution with mean 1.5% for all three time periods. 3. A mortgage whose interest rates decreased over time. The interest rates were drawn from normal distributions with means 2.5%, 1.5% and 0.5% and for the three time periods, respectively. All distributions had a standard deviation of 0.44%. The minimum value of an interest rate was set to 0. The mortgage with decreasing interest rates offered the lowest overall interest rate payments and thus represented the best choice. Participants could reveal up to three different interest rates for no fee by clicking on the corresponding table cell. At each point in the task, the participant could decide for one mortgage plan and proceed to the next trial.

Outcome Measures

We quantified far-sightedness by examining which information participants gathered in what order. We defined far-sighted planning as gathering information about the most long-term consequences first (i.e., the prices in airport cities in the Road Trip task and the interest rate for the last 25 years in the Mortgage task). We therefore measured

far-sightedness by the proportion of such far-sighted planning operations among the first k planning operations, where k denotes the number of pieces of information that are available about the most long-term consequences. We call this measure the far-sightedness quotient (FSQ). For example, consider a Road Trip trial that includes two possible final destinations: Choosing these two final destinations in the first 2 clicks results in an FSQ of 1. Choosing one final destination and one stopover in the first two clicks results in an FSQ of 0.5. If a person performed fewer planning operations than there were far-sighted planning options, k was reduced to the number of performed planning operations. The values are reported as percentages.

We utilized the click agreement metric from Experiment 1 (see Section 3.1.2) to capture how well participants applied the far-sighted planning strategy recommended by the AI-generated decision aid. Whether a performed planning operation is consistent with the far-sighted planning strategy depends on the strategy's stopping rule. The far-sighted planning strategy stops planning if it encounters the best possible long-term outcome, which is given by a hotel price of $20 in the Road Trip task and by an interest of 0% in the Mortgage task.

In addition, we quantified participants' planning success. In the Road Trip task was measured the score per trial which was defined as the sum of lookup fees and route costs subtracted from the initial budget of $500. In the Mortgage task, we measured participants' performance by whether they selected the mortgage plan with decreasing interest rates because its total cost was always the lowest.

Procedure

Participants were randomly divided into an experimental group and a control group. Each condition consisted of a Mortgage task block and a Road Trip task block, and a final demographic survey. The order in which the two task blocks were presented was randomized across participants. Both task blocks opened with instructions on the task (see Appendix 3), followed by a multiple-choice quiz on the task, an additional instructions page displayed only in the experimental condition (see Appendix 3), and eight trials of the actual task. The instructions of the Road Trip task explicitly stated that there is always a hotel in one of the airport cities with a rate of only $20 per night. Participants received a bonus of 2 pence for every 100 points scored in the Road Trip task, or 5 pence per point scored in the Mortgage task. Additionally, everyone received a base payment of £1.5. In the experimental condition, participants were assisted by our AI-generated decision aids. These decision aids were introduced as "Advice for scoring a high bonus" on the additional instructions page. Participants were told that the decision

Fig. 8 Experiment 2: The Road Trip task (**a**) and the Mortgage task (**b**) with corresponding AI-generated decision aids as presented to the experimental condition. Only participants in the experimental condition were assisted with decision aids
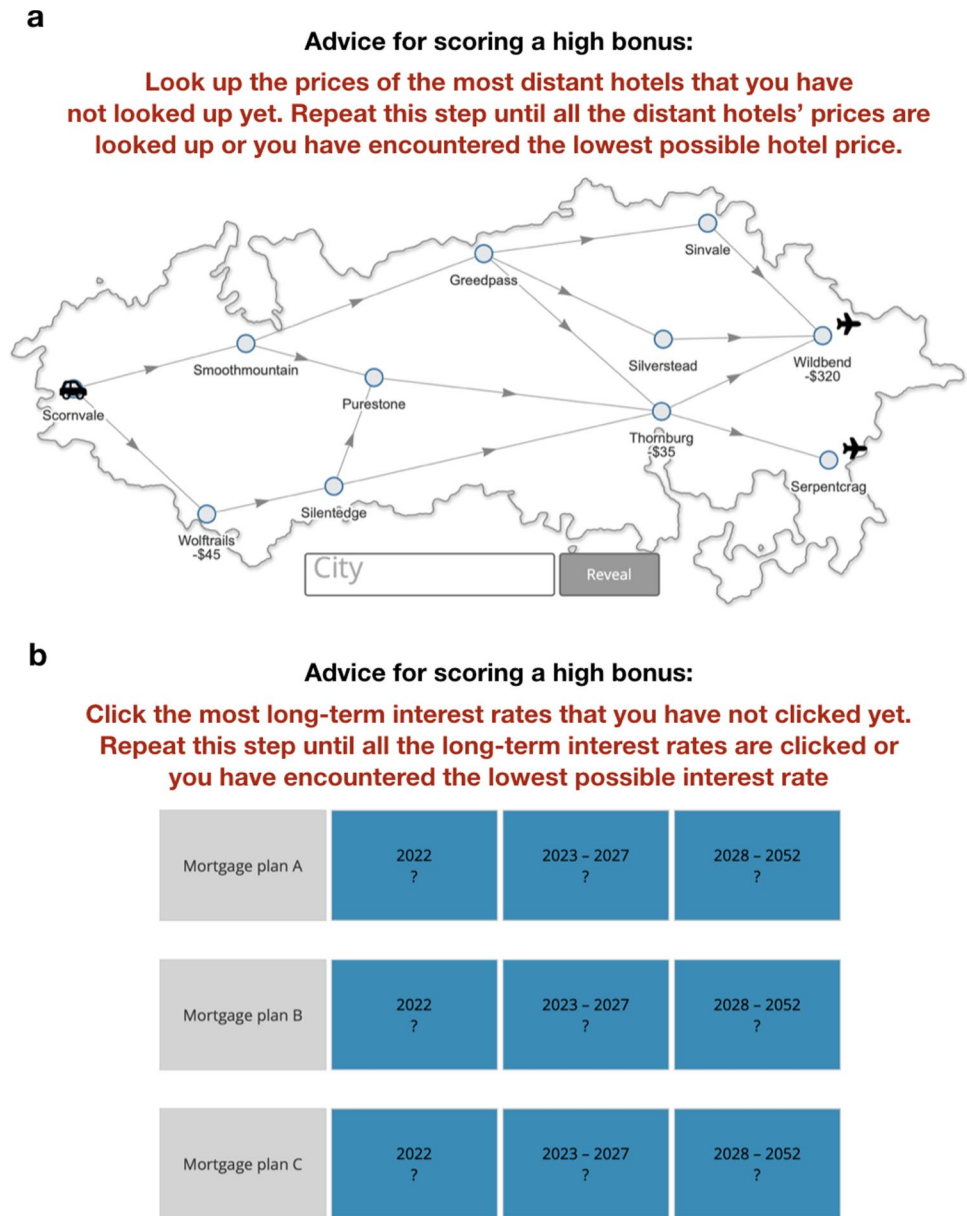
aids convey a near-optimal strategy for gathering information to arrive at a good decision. Participants were asked to try to understand the advice of the decision aid and how it can be applied to the task at hand. In addition, the AI-generated decision aids were displayed in the task trials at the top in red font with the note: "Advice to achieve a high bonus" (see Fig. 8a and b).

As described above, the AI-generated decision aids comprised procedural instructions for how to reach a decision. The procedural instructions for the Mortgage task were "*Click the most long-term interest rates that you have not clicked yet. Repeat this step until all the long-term interest rates are clicked or you have encountered the lowest possible interest rate.*" The procedural instructions for the Road Trip task were "*Look up the prices of the most distant hotels that you have not looked up yet. Repeat this step until all the distant hotels' prices are looked up or you have encountered the lowest possible hotel price.*"

Analysis

We performed one-sided Mann-Whitney-U tests for group level comparisons according to our hypotheses that the experimental group would plan more far-sightedly and score higher than the control group. The study was pre-registered.[3]

---

[3] https://aspredicted.org/JRD_D7Z

Fig. 9 Results of Experiment 2. The decision aids generated by our AI method boosted participants far-sightedness and performance in the Road Trip and Mortgage task. The experimental group was aided by the procedural instructions generated by our computational method. All differences are statistically significant (all $p \leq .01$). Panels **a** and **b**: The experimental group planned more far-sighted than the control group. Panels **c** and **d**: The experimental group achieved higher click agreements than the control group. Panels **e** and **f**: Performance in both tasks was measured according to each task's objective. Higher scores mean better performance
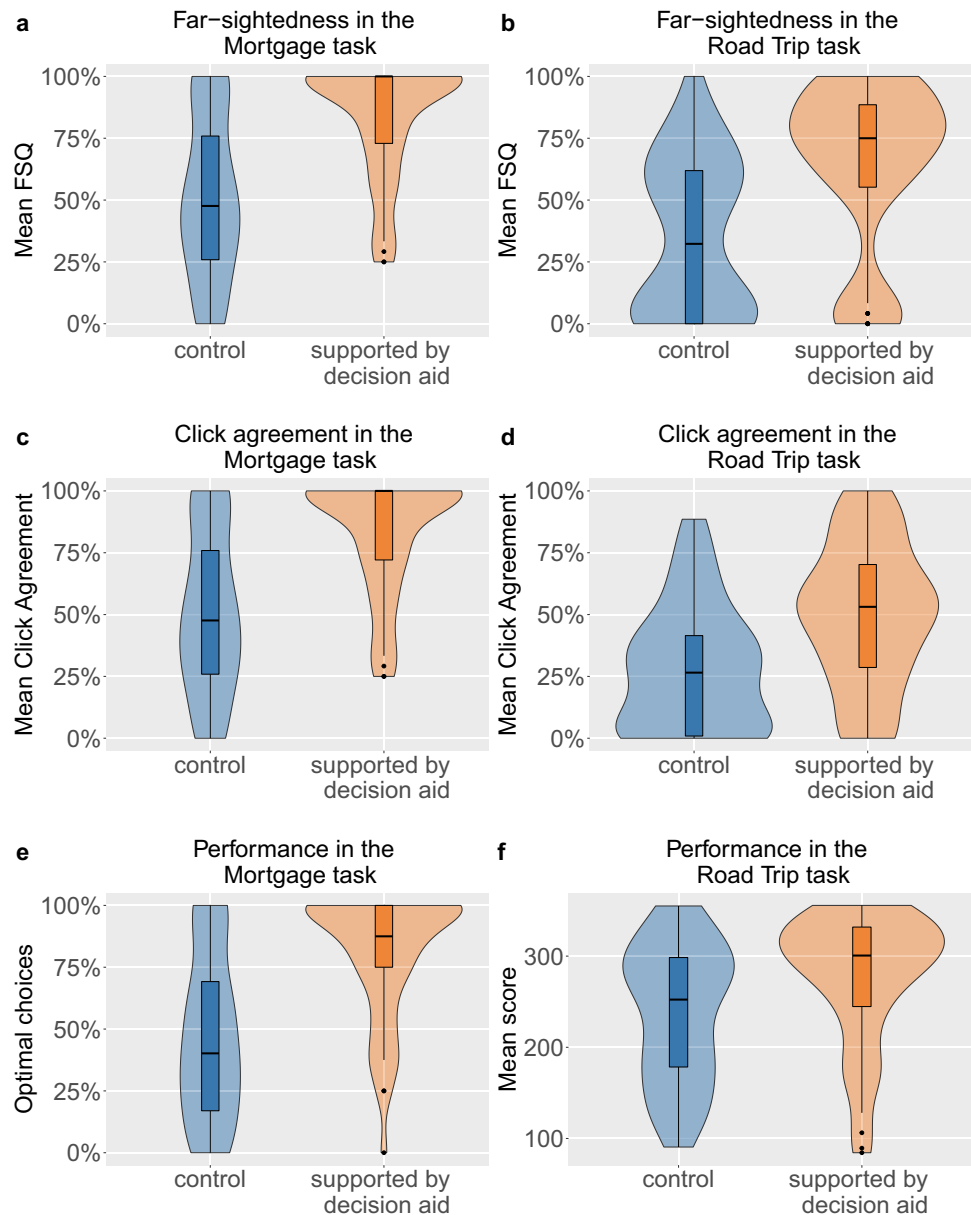
Results

As illustrated in Fig. 9a and b, the experimental group planned significantly more far-sightedly than the control group in both tasks (Road Trip task: $A = .75$; $U=741$, $p=<.001$, Mortgage task: $A = .78$; $U=658$, $p=<.001$). In the Mortgage task, the automatically generated decision aid increased participants' average FSQ from 50.8% in the control condition to 83.3% in the experimental condition (medians: 47.6% vs. 100%). In the Road Trip task, the AI-generated decision aid increased the average FSQ from 35.1% in the control condition to 63.8% in the experimental condition (medians: 32.3% vs. 75%). In a follow-up analysis, we found that participants who had encountered the minimum airport price in one trial of the Road Trip task planned significantly

more far-sightedly in the following trial compared to trials in which participants had not encountered the minimum airport price in the previous trial (FSQ: 74.7% vs. 23.2%; $A = .68$; $U=3203$, $p=<.001$).

The positive effect of AI-generated decision aids was also evident in participants' click agreement with the instructed strategy. Participants in the experimental group showed significantly higher click agreement than the control group in both tasks (Road Trip task: $A = .72$; $U=838$, $p=<.001$, Mortgage task: $A = .78$; $U=659$, $p=<.001$). In the Road Trip task, the average click agreement was 27.9% in the control condition and 49.8% in the experimental condition (medians: 26.5% vs. 53.1%; see Fig. 9d). In the Mortgage task, the stopping rule of the recommended strategy was never met; thus the mean click agreement only differed from the

FSQ in the sense that it penalized not clicking all long-term options available. This was rarely the case; hence the click agreement was similar to the FSQ (control condition: 50.4%; experimental condition: 82.9% see Fig. 9c).

Furthermore, we found that the AI-generated decision aid significantly boosted participants' performance in both tasks (Road Trip task: $A = .66$; $U=994$, $p=.001$, Mortgage task: $A = .78$; $U=654$, $p=<.001$). On average a participant in the control condition selected the cheapest mortgage in only 46.8% of trials (median: 40.2%); by contrast participants in the experimental condition selected the cheapest mortgage 80.7% of the time (median: 87.5%; see Fig. 9e). In the Road Trip task, the AI-generated decision aid increased the participants' score from 237.4 points in the control condition (median: 252.3) to 276.3 points in the experimental condition (median: 300.6; see Fig. 9f).

Finally, we inspected how participants reached their decisions when they deviated from the recommended strategy. To do so, we averaged the frequency of the most common ways in which participants deviated from the optimal strategy across the two tasks. Out of every trial in which participants did not follow the recommended strategy, 21.1% of the trials involved no clicks at all, in 35% of trials participants started by inspecting an immediate outcome, in 12.5% of trials participants started by clicking on an intermediate outcome, and in 31.4% of trials participants started by inspecting a final outcome but then deviated from the optimal strategy in a later step. Two thirds of the time, the latter deviation already occurred in the second click.

Discussion and Conclusion

In this work, we developed a new approach to boosting human decision-making. While most previous work designed decision aids by hand, we extended a computational method that uses AI to generate decision aids automatically. Our method discovers and describes near-optimal strategies for human decision-making. The main technical contribution of this article was to develop an algorithm for transforming disjunctive normal form descriptions of planning strategies into procedural instructions for good decision-making that are easy to understand.

The results of Experiment 1 suggested that people can understand the kind of procedural instructions generated by our new method faster and follow the strategy more accurately than when it was conveyed by the decision aids generated by our previous method (Skirzyński et al., 2021a). Moreover, we demonstrated that the decision aids generated by our new method can improve human decision-making in two naturalistic tasks: planning a road trip and choosing a mortgage. Our AI-generated decision aids improved the process and outcomes of people's decision-making in both tasks. This happened despite the fact people had the freedom to use their own decision strategy for those tasks. Presumably, people used the strategies hinted by our decision aid since they believed that it would be beneficial for them. To the best of our knowledge, this is the first demonstration that human decision-making in naturalistic tasks can be improved in this way by AI-generated natural language descriptions of near-optimal decision strategies.

Together, the two major components of our AI-powered boosting method, that is AI-Interpret and DNF2LTL, give rise to a new algorithm for explainable reinforcement learning that can create descriptions of virtually any learned policy (Puiutta & Veith, 2020; Dazeley et al., 2021). Our experiments indicated that this new algorithm generates syntactically interpretable descriptions, that when translated into natural language, facilitate people's trust. People followed the policy and planned far-sightedly in Experiment 2 after only being presented with the output of AI-powered boosting as a decision aid they could use or ignore. Future work on explainable reinforcement learning should aim to test this new algorithm on a wider range of tasks and environments.

The goal of this article was to develop a method for improving human decision-making. We therefore evaluated our overall approach by whether and to which extent the resulting decision aids leads to better decisions. This metric depends on how much better the resource-rational strategy for a given decision problem is than people's intuitive strategy, the method used to discover the resource-rational strategy, and how accurately the automatically generated natural language description captures the essence of the discovered strategy. Each of those components can be assessed separately. For an in-depth evaluation of how accurately the natural language instructions generated by our method describe the different decision strategies, please see Skirzyński et al. (2021b).

Our experiments demonstrate that the decision aid boosted people's performance in the assisted decisions by guiding them through the process of executing a resource-rational decision strategy. This strategy ensured that participants considered the most important consequences of the actions they were choosing between. It thereby guided them to utilize crucial information that they might not have seen otherwise. In principle, this improvement might have occurred solely because the participant blindly followed the instructions of the decision aid. However, it is also conceivable that practicing good decision-making with a decision aid improves people's decision-making competency. Concretely, people's decision-making competence might improve because they internalize the described decision strategy through repetition or because they gain insights into the logic of the conveyed strategy and why it is adaptive. If that were the case, then they might also use the conveyed

strategy in future decisions they make without the decision aid. In that case, our decision aid would have boosted not just their performance in the task, but also their decision-making competence. Given that people can learn and transfer adaptive decision strategies through practice (Callaway et al., 2022a; He et al., 2021; He & Lieder, 2022), repeatedly using our decision aids might indeed have led to learning-induced improvements in their decision-making competency. Testing whether such learning occurs is an important direction for future research.

Whether our decision aid boosted people's decision-making competence is closely related to the question of why and how its provision improved people's decisions. There are at least two possible mechanisms: insight versus compliance. According to the first hypothesis (i.e., insight), the decision aid helped people gain insights into the logic of good decision-making that they then autonomously applied to improve their performance. Concretely, people might try out the recommended strategy because they have some level of trust in such recommendations. Because the recommended strategy is highly adaptive, the experienced outcomes will likely convince the participants that using the strategy is beneficial for them. Moreover, they might realize that the far-sighted strategy is adaptive because the final outcomes are more variable than earlier outcomes (insight). Based on that, they might then choose to continue using the strategy because they conclude that it works well and makes sense. According to the second hypothesis, participants interpreted the decision aid as a series of orders that they felt obliged to follow. Moreover, participants might have followed those instructions without understanding why they make any sense. We think that the most extreme version of this interpretation is unlikely because Experiment 2 emphasized the participants' autonomy by framing the decision aid as advice (see Fig. 8). However, participants often comply with what they perceive to be the experimenter's implicit expectations, regardless of what the experimenter's actual intent is Orne (1996). Moreover, our experiments were not designed to distinguish between the two interpretations. Therefore, the mechanism through which the observed improvements were attained remains unclear. To address this important question, future studies should measure why participants followed the recommended strategy and test participants' understanding of the logic behind the recommended strategy and why it is effective. Based on existing measures of autonomous motivation (Sheldon et al., 2004), future studies could measure why participants followed the recommended strategy by asking them to rate to which extent they followed the strategy because they felt that it was required or expected of them and to separately rate to which extent they followed it because it made sense to them and because they thought it was a good strategy.

Since we designed our algorithm for generating natural language descriptions of decisions strategies with a particular class of problems in mind and only tested it on those problems, it remains unclear how well it would work for other kinds of decision problems. The principles are general enough that they are applicable to a wide range of decision problems. However, the dictionary is application-specific, and the implementation uses heuristics that probably will not work well for all possible applications. Therefore, depending on how different future applications will be from the ones we tackled in this article, our method will require some amount of adaptation. Nevertheless, our work provides a proof of concept that it is possible to discover and describe rational decision strategies automatically. Developing a more principled and more general translation algorithm is an important direction for future work.

Based on our positive findings on improving people's decision-making, we believe that future research should focus on even more realistic tasks and decision support in the real world. In this work, we found that the near-optimal decision strategy that the automatic strategy discovery method (Callaway et al., 2022a; Skirzyński et al., 2021a) discovered for a simple Mouselab-MDP task (see Fig. 3) could be automatically translated into effective decision aids for two more complex and more naturalistic tasks (see Fig. 2). This worked not only for the Road Trip task that is structurally similar to the Mouselab-MDP task, but also for the problem of choosing a mortgage that is analogous to the Mouselab-MDP task at a more abstract level. This suggests that our method is, in principle, applicable to a wide range of decision problems people face in the real world as long as the essential structure of those problems can be modeled within our general metalevel MDP framework (Griffiths et al., 2019; Callaway et al., 2022b; a). We have previously argued that this is the case for a wide range of common real-world decisions, such as purchasing decisions, hiring choices, investment decisions, deciding which charity to donate to, medical diagnosis, treatment planning, and credit approval decisions (Mehta et al., 2022; Skirzyński et al., 2021a; Consul et al., 2022; Callaway et al., 2022a).

Applying our approach to such real-world problems requires building models of real-world decisions. Developing such models generally requires making evidence-based assumptions about the structure of the real world. Our knowledge of the real-world problems in which people have to make decisions are inevitably uncertain, sometimes inaccurate, and usually incomplete (Hertwig et al., 2019). However, our approach does not require that all of those assumptions are correct. To the contrary, the methods described here can be combined with recent advances that have made strategy discovery methods robust to errors in the model of the decision problems to be solved (Mehta et al., 2022). Moreover, our strategy discovery methods can also be extended to environments where the

Fig. 10 Experiment 1: **a** Procedural instructions are more interpretable than flowcharts. The plot shows the mean click agreement of participants per condition as a function of trial. Click agreement with the instructed clicking strategy was measured as the proportion of consistent clicks out of all performed clicks. **b** The mean expected score that per condition as a function of trial. The shaded areas mark the 95% confidence intervals
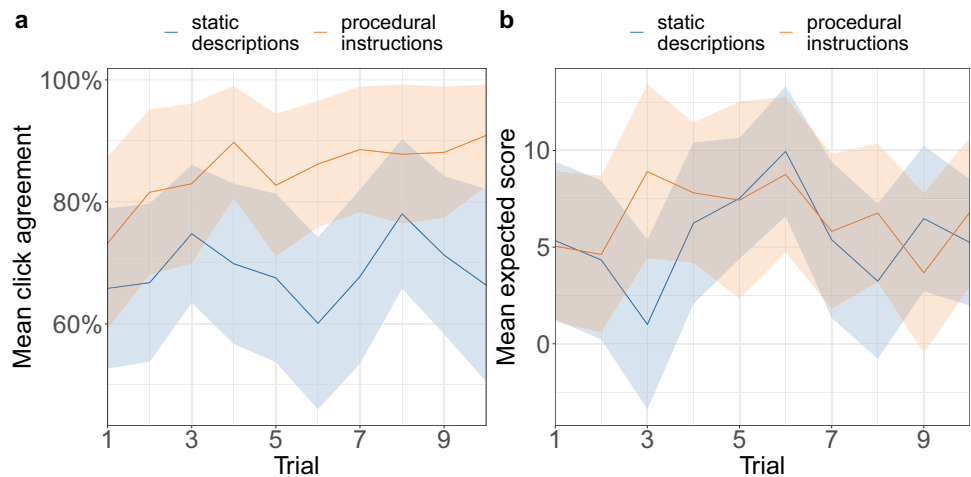
true state of affairs is unknown because it cannot be observed directly (Heindrich et al., 2022). In practice, the applicability of our method also depends on the size of the model. However, technical advances in machine learning methods for automatic strategy discovery are making our approach increasingly more scalable (Consul et al., 2022). Regardless thereof, making AI-powered boosting work in the real world remains a difficult challenge. Whether future work will be able to overcome the remaining difficulties remains to be seen, but the results presented in this article make us cautiously optimistic.

Appendix 1. Experiment 1: Task instructions

Experiment 1 consisted of a condition that was assisted by static descriptions (flowchart) and a condition that was assisted by procedural instructions. To inform participants how to use the static descriptions in the task at hand, we used the following instructions:

"Flowchart: The description of the strategy takes the form of a flowchart. It walks you through a list of one or more questions that you need to answer to by looking at the tree, and describes which nodes to click. Look at the image below to see how a flowchart can look like. Task procedure: 1. Read the flowchart carefully. 2. Think of a node you would like to click 3. Go through the flowchart and answer questions about that node. 4. Click that node if the flowchart landed you in a "Click it" decision. Otherwise, think of a different node. 5. Once you are sure that you clicked all the nodes the flowchart allows clicking — that is it would evaluate to "Don't click it" for every node — click Next to advance to the next trial."

To inform participants how to use the procedural instructions in the task at hand, we used the following instructions:

"Instructions: The description of the strategy is conveyed as a sequence of instructions. The instructions tell you what to click step by step. Look at the image below to see how a sequence of instructions can look like. Task procedure To enact the strategy conveyed by the instructions, do the

following: 1. Read the instructions carefully. 2. Click the nodes by following the procedure described in the instructions. 3. Once there are no more nodes the instructions allow clicking, click Next to advance to the next trial."

Appendix 2. Experiment 1: Development of click agreement over time

The development of click agreement over time is illustrated in Fig. 10a. A linear regression model found that the click agreement in the static descriptions condition kept stable over time ($\beta = 0.002, p = .66$), whereas it significantly increased in the procedural instructions condition ($\beta = 0.013, p = .002$), suggesting that the procedural instructions condition learned to apply the instructed strategy more precisely over time. Further, we found that the expected score (see Fig. 10b) did not systematically change over time in the static descriptions condition ($\beta = 0.152, p = .487$) and in procedural descriptions condition ($\beta = -0.038, p = .87$). Lastly, we found that the fitted intercepts did not significantly differ between conditions (click agreement: $\beta = 0.1, p = .198$; expected score: $\beta = 2.1, p = .279$), indicating that there were no differences in the initial understanding of the task between conditions.

Appendix 3. Experiment 2: Task instructions

These were the instructions on the Road Trip Task: "In the Travel Planner game, you pretend to be a travel planner. You start by seeing a map as shown below. Your client needs to travel from the city with the car (Ruby Ridge) to one of the cities with an airport. Getting from city A to city B is only possible when there is an arrow from city A to city B. Your client can travel only one city per day. During the night, he or she stays in a hotel, which costs money. Your client wants a morning flight, so they must

Fig. 11 Experiment 2: The experimental group (red) was supported by our decision aid whereas the control group (blue) was not. **(a)** The plot shows the average far-sightedness quotient as a function of trial per condition for the Mortgage task. **b** The plot shows the average far-sightedness quotient as a function of trial per condition for the Road Trip task. **c** The plot shows the proportion of optimal choices as a function of trial per condition for the Mortgage task. **d** The plot shows the average score as a function of trial per condition for the Road Trip task. The shaded areas mark the 95% confidence intervals
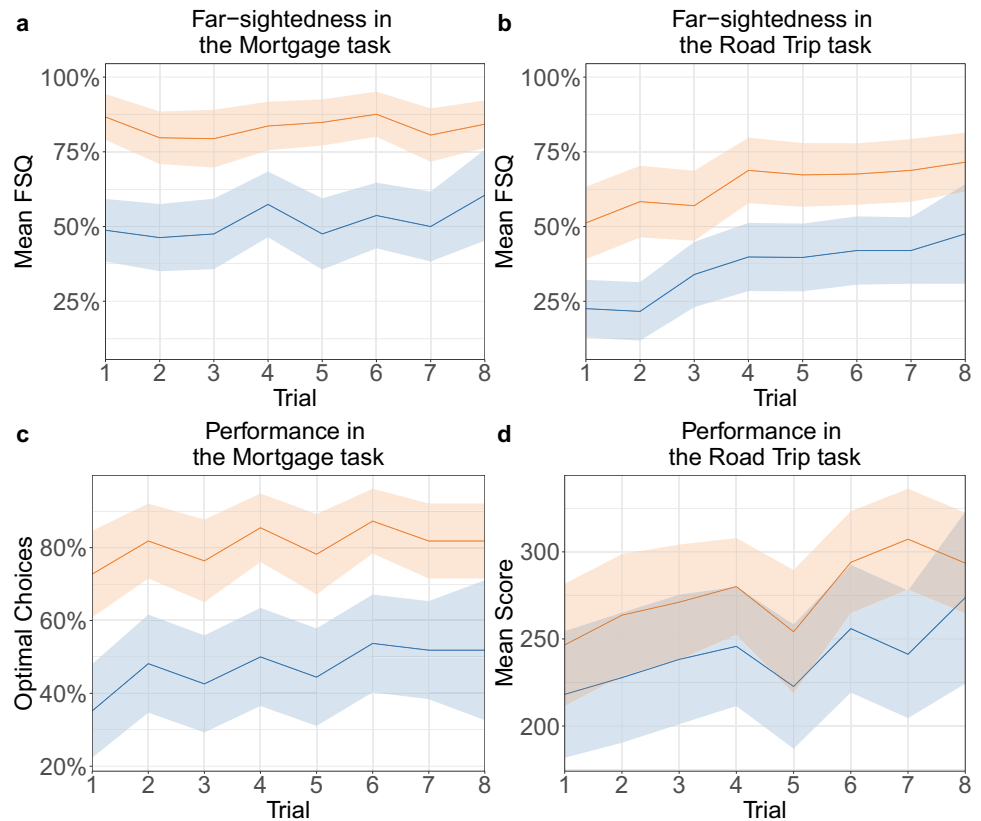
pay for a hotel in the airport city as well. The price of the hotel varies between the different cities. Airport hotels start at $20. Your client is on a tight budget of $500 and wishes to take the cheapest route. Your goal is to choose which cities to traverse so that the price of the trip was as cheap as possible. You can look up the price of the cheapest hotel in a city by typing the city name in a text box and clicking Reveal. The prices are negative to convey the cost you will incur by staying in the city. When you look up a city, its price is revealed on the map. Revealing the price costs $10. At any time, you can select parts of the client's route by clicking on the arrows. If you change your mind, you can unselect arrows by clicking them again. When you have finalized your route, click Submit. You do not need to check the prices of every city on the route before submitting."

In addition, the experimental group received this information: "Advice for scoring a high bonus: To help you score higher in the roadtrip planner game, we will show you its near-optimal strategy. This strategy describes in what order to explore the hotel prices. Please take a moment to understand this advice and how you could apply it in the game. Look up the prices of the most distant hotels that you have not looked up yet. Repeat this step until all the distant hotels' prices are looked up or you have encountered the lowest possible hotel price."

These were the instructions on the Mortgage task: "In the Mortgage game, you have found your dream property and want to ask the bank for a loan. The bank presents you with three different mortgage plans. Each mortgage plan has three different interest rates: One for the 1st year (2022), one for the 2nd until 5th year (2023–2027) and one for the 6th until 30th year (2028–2052). Unfortunately, the bank clerk forgot to tell you about the interest rates. In the example below, you can see three plans (Mortgage plan A, Mortgage plan B, Mortgage plan C) but their corresponding interest rates are hidden underneath the blue fields. You decide to call the bank to ask about the interest rates. However, the bank clerk only has time to tell you up to three interest rates. Each time a bank clerk tells you about the interest rate corresponds to one click. That means you can only click up to 3 times. Below you will see an example with one field revealed. In the example, the interest rate from 2023 to 2027 for mortgage plan B was revealed. In the example, you would have to pay 1.61% interest rate in each of the 4 years when you select mortgage plan B. You can click up to 3 times, after which you have to make a decision which mortgage plan to choose. You can select a mortgage plan at any time by clicking on the grey mortgage plan button (A, B or C)."

In addition, the experimental group received this information: "Advice for scoring a high bonus: To help you score higher in the mortgage game, we will show you its near-optimal strategy. This strategy describes in what order to

explore the interest rates. Please take a moment to understand this advice and how you could apply it in the game: Click the most long-term interest rates that you have not clicked yet. Repeat this step until all the long-term interest rates are clicked, or you have encountered the lowest possible interest rate."

Appendix 4. Experiment 2: Development of FSQ over time

In an exploratory analysis, we regressed the participant's FSQ in each task on the predictors *decision aid* and *trial number* and their interaction. We found that the intercept was significantly larger in the experimental condition than in the control condition in both tasks (Mortgage task: $\beta = 0.35, p < .001$; Road Trip task: $\beta = 0.31, p < .001$), suggesting that the provision of the decision aid led to an immediate improvement in far-sightedness. As illustrated in Fig. 11, we found that in the Mortgage task the FSQ kept stable over time (trial number: $\beta = 0.01, p = .075$), whereas we found that the FSQ increased over time in the Road Trip task (trial number: $\beta = 0.03, p < .001$). The interaction of trial number and decision aid was insignificant for both tasks (Mortgage task: $\beta = 0.00, p = .266$; Road Trip task: $\beta = 0.00, p = .357$), indicating that the presence of the decision aid did not limit the learning.

Appendix 5. Experiment 2: Development of performance over time

In addition, we regressed the participants' performance in each task on the predictors *decision aid* and *trial number* and their interaction. We found that the intercept was significantly larger in the decision aid condition than in the no aid condition in the Mortgage task, but not in the Road Trip task (Mortgage task: $\beta = 2.5, p < .001$; Road Trip task: $\beta = 29.4, p = .1$). As illustrated in Fig. 11c, we found that in the Mortgage task, the number of optimal choices increased over time (trial number: $\beta = 0.12, p = .034$). As illustrated in Fig. 11d, we found that in the Road Trip task the score increased over time, however not significantly (trial number: $\beta = 4.7, p = .068$). The interaction of trial number and decision aid was insignificant for both tasks (Mortgage task: $\beta = -0.02, p = .785$; Road Trip task: $\beta = 2.3, p = .509$).

Declarations

References

Becker, F., Skirzyński, J., van Opheusden, B., & Lieder, F. (2021). Encouraging far-sightedness with automatically generated descriptions of optimal planning strategies: Potentials and limitations. In: *Proceedings of the annual meeting of the cognitive science society* (vol. 43)

Callaway, F., Lieder, F., Krueger, P.M., & Griffiths, T.L. (2017). Mouselab-MDP: A new paradigm for tracing how people plan. In: *The 3rd multidisciplinary conference on reinforcement learning and decision making*. Ann Arbor, MI

Callaway, F., Gul, S., Krueger, P., Griffiths, T.L., & Lieder, F. (2018a). Learning to select computations. In: *Uncertainty in artificial intelligence: Proceedings of the thirty-fourth conference*

Callaway, F., Lieder, F., Das, P., Gul, S., Krueger, P.M., Griffiths, T. (2018b). A resource-rational analysis of human planning. In: *CogSci*

Callaway, F., Jain, Y. R., van Opheusden, B., Das, P., Iwama, G., Gul, S., et al. (2022). Leveraging artificial intelligence to improve people's planning strategies. *Proceedings of the National Academy of Sciences, 119*(12), e2117432119.

Callaway, F., van Opheusden, B., Gul, S., Das, P., Krueger, P., Lieder, F., & Griffiths, T. (2022b). Rational use of cognitive resources in human planning. *Nature Human Behavior*

Consul, S., Heindrich, L., Stojcheski, J., & Lieder, F. (2022). Improving human decision-making by discovering efficient strategies for hierarchical planning. *Computational Brain & Behavior, 5*(2), 185–216.

Dazeley, R., Vamplew, P., & Cruz, F. (2021). Explainable reinforcement learning for broad-xai: a conceptual framework and survey. arXiv:2108.09003

Gigerenzer, G. (1991). How to make cognitive illusions disappear: Beyond "heuristics and biases''. *European Review of Social Psychology, 2*(1), 83–115.

Gigerenzer, G., & Todd, P. M. (1999). *Simple heuristics that make us smart*. USA: Oxford University Press.

Gigerenzer, G., Hertwig, R., Hoffrage, U., & Sedlmeier, P. (2008). Cognitive illusions reconsidered. *Handbook of Experimental Economics Results, 1*, 1018–1034.

Gilovich, T., Griffin, D., Kahneman, D., et al. (2002). *Heuristics and biases: The psychology of intuitive judgment*. Cambridge: Cambridge University Press.

Griffiths, T. L., Callaway, F., Chang, M. B., Grant, E., Krueger, P. M., & Lieder, F. (2019). Doing more with less: meta-reasoning and meta-learning in humans and machines. *Current Opinion in Behavioral Sciences, 29*, 24–30.

Hafenbrädl, S., Waeger, D., Marewski, J. N., & Gigerenzer, G. (2016). Applied decision making with fast-and-frugal heuristics. *Journal of Applied Research in Memory and Cognition, 5*(2), 215–231.

He, R., & Lieder, F. (2022). Where do adaptive planning strategies come from? https://doi.org/10.13140/RG.2.2.28966.60487, manuscript submitted for publication.

He, R., Jain, Y.R., & Lieder, F. (2021). Measuring and modelling how people learn how to plan and how people adapt their planning strategies the to structure of the environment. In: *International conference on cognitive modeling*. Retrieved from https://re.is.mpg.de/uploads_file/attachment/attachment/671/20210720_ICCM_submission_final.pdf. Accessed 5 Sep 2022

Heindrich, L., Consul, S., & Lieder, F. (2022). Leveraging ai to improve human planning in large partially observable environments. Manuscript submitted for publication

Hertwig, R., & Grüne-Yanoff, T. (2017). Nudging and boosting: Steering or empowering good decisions. *Perspectives on Psychological Science, 12*(6), 973–986.

Hertwig, R., Pleskac, T. J., & Pachur, T. (2019). *Taming uncertainty*. Cambridge: MIT Press.

Jain, Y.R., Callaway, F., Griffiths, T.L., Dayan, P., Krueger, P.M., & Lieder, F. (2021). A computational process-tracing method for measuring people's planning strategies and how they change over time. Manuscript in revision

Kinnier, R. T., & Metha, A. T. (1989). Regrets and priorities at three stages of life. *Counseling and Values, 33*(3), 182–193.

Larrick, R.P. (2004). Debiasing. Blackwell handbook of judgment and decision making pp 316–338

Lieder, F., & Griffiths, T. L. (2017). Strategy selection as rational metareasoning. *Psychological Review, 124*(6), 762.

Lieder, F., & Griffiths, T.L. (2020a). Advancing rational analysis to the algorithmic level. *Behavioral and Brain Sciences, 43*

Lieder, F., & Griffiths, T. L. (2020). Resource-rational analysis: understanding human cognition as the optimal use of limited computational resources. *Behavioral and Brain Sciences, 3*, 1–85.

Lieder, F., Krueger, P.M., & Griffiths, T. (2017). An automatic method for discovering rational heuristics for risky choice. In: *CogSci*

Lieder, F., Callaway, F., Jain, Y.R., Krueger, P.M., Das, P., Gul, S., & Griffiths, T. (2019). A cognitive tutor for helping people overcome present bias. In: *The fourth multidisciplinary conference on reinforcement learning and decision making.* * These authors contributed equally

Mehta, A., Jain, Y.R., Kemtur, A., Stojcheski, J., Consul, S., Tošic, M., & Lieder, F. (2022). Leveraging machine learning to automatically derive robust decision strategies from imperfect knowledge of the real world. *Computational Brain & Behavior*

Meier, S., & Sprenger, C. (2010). Present-biased preferences and credit card borrowing. *American Economic Journal: Applied Economics, 2*(1), 193–210.

Milkman, K. L., Rogers, T., & Bazerman, M. H. (2008). Harnessing our inner angels and demons: What we have learned about want/should conflicts and how that knowledge can help us reduce short-sighted decision making. *Perspectives on Psychological Science, 3*(4), 324–338.

Miller, G. A., Galanter, E., & Pribram, K. H. (1960). Plans and the structure of behavior. Henry Holt and Co. https://doi.org/10.1037/10039-000

O'Donoghue, T., & Rabin, M. (2015). Present bias: Lessons learned and to be learned. *American Economic Review, 105*(5), 273–79.

Orne, M.T. (1996). Demand characteristics. In: *Introducing psychological research* (pp. 395–401). Springer

Phillips, N. D., Neth, H., Woike, J. K., & Gaissmaier, W. (2017). FFTrees: A toolbox to create, visualize, and evaluate fast-and-frugal decision trees. *Judgment and Decision making, 12*(4), 344–368.

Puiutta, E., & Veith, E. (2020). Explainable reinforcement learning: A survey. In: *International cross-domain conference for machine learning and knowledge extraction* (pp. 77–95). Springer

Reeck, C., Wall, D., & Johnson, E. J. (2017). Search predicts and changes patience in intertemporal choice. *Proceedings of the National Academy of Sciences, 114*(45), 11890–11895.

Rich, P., Blokpoel, M., de Haan, R., & van Rooij, I. (2020). How intractability spans the cognitive and evolutionary levels of explanation. *Topics in cognitive science, 12*(4), 1382–1402.

Rudin, C., Chen, C., Chen, Z., Huang, H., Semenova, L., & Zhong, C. (2022). Interpretable machine learning: Fundamental principles and 10 grand challenges. *Statistics Surveys, 16*(none), 1–85. https://doi.org/10.1214/21-SS133.

Ruscio, J. (2008). A probability-based measure of effect size: robustness to base rates and other factors. *Psychological methods, 13*(1), 19.

Sala, G., & Gobet, F. (2017). Does far transfer exist? negative evidence from chess, music, and working memory training. *Current Directions in Psychological Science, 26*(6), 515–520.

Sala, G., Aksayli, N.D., Tatlidil, K.S., Tatsumi, T., Gondo, Y., Gobet, F., Zwaan, R., & Verkoeijen, P. (2019). Near and far transfer in cognitive training: A second-order meta-analysis. *Collabra: Psychology, 5*(1)

Sheldon, K. M., Elliot, A. J., Ryan, R. M., Chirkov, V., Kim, Y., Wu, C., et al. (2004). Self-concordance and subjective well-being in four cultures. *Journal of Cross-Cultural Psychology, 35*(2), 209–223.

Simon, H. A. (1997). *Models of bounded rationality: Empirically grounded economic reason* (Vol. 3). Cambridge: MIT Press.

Skirzyński, J., Becker, F., & Lieder, F. (2021a). Automatic discovery of interpretable planning strategies. *Machine Learning*, 1–43.

Skirzyński, J., Jain, Y.R., & Lieder, F. (2021b). Automatic discovery and description of human planning strategies. arXiv:2109.14493

Sutton, R. S., & Barto, A. G. (2018). *Reinforcement learning: An introduction*. Cambridge: MIT Press.

Tversky, A., & Kahneman, D. (1974). Judgment under uncertainty: Heuristics and biases. *Science, 185*(4157), 1124–1131. https://doi.org/10.1126/science.185.4157.1124

Van Rooij, I. (2008). The tractable cognition Thesis. *Cognitive Science, 32*(6), 939–984.