A Scalable Machine Learning Approach to Improving Human Decision Making

Dissertation

der Mathematisch-Naturwissenschaftlichen Fakultät

der Eberhard Karls Universität Tübingen

zur Erlangung des Grades eines

Doktors der Naturwissenschaften

(Dr. rer. nat.)

vorgelegt von

Lovis Heindrich

aus Wiesbaden

Tübingen

2025

Gedruckt mit Genehmigung der Mathematisch-Naturwissenschaftlichen Fakultät der Eberhard Karls Universität Tübingen.

Tag der mündlichen Qualifikation:	19.05.2025
Dekan:	Prof. Dr. Thilo Stehle
1. Berichterstatter/-in:	Prof. Dr. Falk Lieder
2. Berichterstatter/-in:	Prof. Dr. Peter Dayan

Thesis advisor: Professor Falk Lieder Lovis Heindrich

A scalable machine learning approach to improving human decision making

Abstract

Human decision-making is often suboptimal due to biases and cognitive limitations. Making good decisions requires efficient metareasoning — weighing the costs and benefits of investing additional (mental) resources into refining a decision. This has been formalized in the theory of resource-rationality, which describes the quality of a decision-strategy as a combination of its expected reward and the expended cognitive resources. While the extent of optimal decision-making in humans is still a debated topic, it is generally understood that people's decision-making is not fully rational even when taking their computational constraints into account. While humans have developed useful heuristics that circumvent some of these issues, in many situations they can also lead to highly suboptimal outcomes.

Improving human decision-making by teaching them better decision-strategies is a promising research direction with a large potential of improving people's lives. One promising approach towards this goal is research on strategy discovery, which aims to develop computational methods that automatically discover efficient decision strategies for a given planning problem, which are then taught to people using intelligent tutoring systems. While current research on strategy discovery and intelligent cognitive tutors has demonstrated first successes in improving human decision-making, current approaches are constrained to small and simplified planning problems, which are far from the complexity of the problems people face in the real world. In this thesis, I identify key limitations of current strategy discovery methods and cognitive tutors, and present novel contributions that move the field closer towards improving human decision-making in the real world.

The first limitation I address is the scalability of existing methods, which are currently limited to relatively small planning problems due to their computational complexity. To overcome this limitation, I introduce hierarchical-BMPS, a strategy discovery method that can discover planning strategies for significantly larger planning problems than existing methods by hierarchically decomposing the planning problem. I demonstrate the effectiveness of hierarchical-BMPS in a computational benchmark, where it performed competitively with existing methods at a fraction of their computational cost, and two human training experiments, where hierarchical-BMPS discovered decision strategies that are more resource-rational than the strategies people intuitively use. Teaching the strategies discovered by hierarchical-BMPS using cognitive tutors improved human decision-making in larger environments than previously possible. Additionally, I introduce a novel cognitive tutoring system that overcomes limitations in the complexity and scalability of existing approaches. The new tutoring system is adapted to teaching complex planning problems by creating a series of practice

problems with increasing difficulty that incrementally build up to the full planning problem. This is achieved by limiting the number and type of actions learners choose between, while also providing feedback on the quality of the learner's choices. The cognitive tutor was evaluated in multiple human training experiments, where it led to significant improvements in human planning.

The second limitation of existing strategy discovery methods is that they were developed for fully observable environments, while most aspects of the world are only partially observable and planning usually features a high degree of uncertainty. To address this limitation, I developed a metareasoning model for planning in partially observable environments, and a novel strategy discovery method (MGPO), that is adapted to partial observability by taking the uncertainty of planning actions into account. MGPO was evaluated in a computational simulation, where it outperformed baseline methods in both the quality of their discovered decision strategies and their computational efficiency. A human training experiment showed that teaching the discovered strategies to people can significantly improve their decision-making in environments featuring uncertainty.

The third limitation I address is that metareasoning has not yet been applied to discovering optimal algorithms for complex real-world problems. First, I introduce a metareasoning model for a first real-world task: project selection. The model parameters are estimated from real-world data. To discover efficient planning strategies, I developed an extended version of MGPO adapted to the unique requirements of the project selection task, and evaluated the method in simulation experiments and a human training experiment, in which it successfully improved human decision-making. Second, I present a metareasoning model for a second class of real-world problems, social dilemmas, which can be used to model a wide range of real-world dilemmas.

The strategy discovery methods and cognitive tutors presented in this thesis overcome multiple critical limitations of previous methods to enable discovering and teaching highly complex planning strategies in more realistic decision environments than previously possible. This constitutes an important step towards applying automatic strategy discovery methods to discover optimal algorithms for solving real-world problems. The cognitive tutors I developed contribute towards translating these advances in algorithm discovery to the real world, and offer a promising path towards teaching people decision strategies they can use in their daily lives.

German abstract Menschliche Entscheidungsfindung ist aufgrund von Bias und kognitiven Einschränkungen oft suboptimal. Gute Entscheidungen zu treffen erfordert effiziente Metakognition — das Abwägen von Kosten und Nutzen im Falle der Verwendung zusätzlicher (mentaler) Ressourcen, um eine Entscheidung zu verbessern. Dies wurde in der Theorie der Ressourcenrationalität formalisiert, welche die Qualität einer Entscheidungsstrategie als Kombination aus dem zu erwartenden Nutzen und den aufgewendeten kognitiven Ressourcen beschreibt. Während das Maß der Optimalität menschlicher Entscheidungen nach wie vor ein umstrittenes Thema ist, wird allgemein angenommen, dass die menschliche Entscheidungsfindung selbst unter Berücksichtigung ihrer eingeschränkten kognitiven Ressourcen nicht vollständig rational ist. Während Menschen nützliche Heuristiken entwickelt haben, um einige dieser Probleme zu umgehen, können diese in vielen Situationen auch zu höchst suboptimalen Ergebnissen führen.

Das Verbessern der menschlichen Entscheidungsfindung durch die Vermittlung besserer Entscheidungsstrategien ist eine vielversprechende Forschungsrichtung mit einem großen Potenzial, das Leben von Menschen zu verbessern. Ein aussichtsreicher Ansatz zur Erreichung dieses Ziels ist die Forschung zur Strategieentdeckung. Diese zielt darauf ab, algorithmische Methoden zu entwickeln, die automatisch effiziente Entscheidungsstrategien für ein gegebenes Planungsproblem entdecken. Diese Strategien werden daraufhin mithilfe intelligenter Tutorensystemen Menschen beigebracht. Wenngleich die aktuelle Forschung auf dem Gebiet der Strategieentdeckung und der intelligenten Tutorensysteme erste Erfolge beim Verbessern menschlicher Entscheidungsfindung gezeigt hat, sind aktuelle Ansätze auf kleine und vereinfachte Planungsprobleme beschränkt, welche von der Komplexität der Probleme, denen sich Menschen in der realen Welt gegenübersehen, noch weit entfernt sind. In dieser Dissertation identifiziere ich die zentralen Einschränkungen von aktuellen Strategieentdeckungsmethoden und intelligenten Tutorensystemen, und präsentiere neue Beiträge, die das Forschungsfeld näher an die Verbesserung der menschlichen Entscheidungsfindung in der realen Welt heranrücken.

Die erste Einschränkung, die ich angehe, liegt in der Skalierbarkeit bestehender Methoden, die momentan aufgrund ihrer Berechnungskomplexität auf relativ kleine Planungsprobleme beschränkt sind. Um diese Einschränkung zu überwinden, führe ich mit hierarchischem-BMPS eine Methode zur Strategieentdeckung ein, die Planungsstrategien durch hierarchische Aufteilung des Planungsproblems für signifikant größere Planungsprobleme finden kann als bestehende Methoden. Ich demonstriere die Effektivität von hierarchichem-BMPS in einem rechnergestützten Benchmark und zwei menschlichen Trainingsexperimenten. In dem Benchmark hat hierarchisches-BMPS mit einem Bruchteil der Rechenleistung vergleichbare Ergebnisse zu bestehenden Methoden erzielt. In den Trainingsexperimenten hat hierarchisches-BMPS Entscheidungsstrategien gefunden, die ressourcenrationaler sind als die von Menschen intuitiv verwendeten Strategien. Die Vermittlung der von hierarchischem-BMPS gefundenen Entscheidungsstrategien mittels intelligenter Tutorensysteme verbesserte die menschliche Entscheidungsfindung in größeren Umgebungen als bisher möglich. Zusätzlich stelle ich ein neues intelligentes Tutorensystem vor, das die Limitierungen in der Komplexität und Skalierbarkeit bestehender Ansätze überwindet. Das neue Tutorensystem ist darauf ausgelegt, komplexe Entscheidungsstrategien zu vermitteln, indem es eine Reihe von Übungsproblemen mit steigendem Schwierigkeitsgrad erzeugt, die sich schrittweise zum vollständigen Planungsproblem aufbauen. Dies wird erreicht, indem die Anzahl und die Art der Aktionen, zwischen denen Lernende auswählen, begrenzt werden, während gleichzeitig Feedback zur Qualität ihrer Entscheidungen gegeben wird. Der intelligente Tutor ist in mehreren menschlichen Trainingsexperimenten evaluiert worden, in denen er zu signifikanten Verbesserungen in der menschlichen Planung führte.

Die zweite Einschränkung existierender Strategieentdeckungsmethoden besteht darin, dass sie für vollständig beobachtbare Umgebungen entwickelt wurden, während die meisten Aspekte der Welt nur teilweise beobachtbar sind und Planung oft ein hohes Maß an Unsicherheit aufweist. Um diese Einschränkung zu beheben, entwickelte ich ein Metakognitions-Modell für die Planung in teilweise beobachtbaren Umgebungen und eine neue Strategieentdeckungsmethode (MGPO), die an teilweise Beobachtbarkeit angepasst ist, indem sie die Unsicherheit von Planungsaktionen

in Betracht zieht. MGPO wurde in einer computergestützten Simulation evaluirt, in der es Referenzmethoden sowohl in der Qualität der gefundenen Entscheidungsstrategien als auch in ihrer Recheneffizienz übertroffen hat. Ein menschliches Trainingsexperiment hat gezeigt, dass das Vermitteln der gefundenen Strategien an Menschen ihre Entscheidungsfindung in Umgebungen, die von Unsicherheit gekennzeichnet sind, signifikant verbessern kann.

Die dritte von mir angegangene Einschränkung liegt darin, dass Metakognition bisher nicht angewandt wurde, um optimale Entscheidungsstrategien für komplexe Probleme der realen Welt zu entdecken. Zunächst stelle ich ein Metakognitions-Modell für ein erstes real existierendes Entscheidungsproblem vor: Projektauswahl. Die Modellparameter wurden anhand realer Daten geschätzt. Um effiziente Planungsstrategien zu entdecken, habe ich eine erweiterte Version von MGPO entwickelt, die an die einzigartigen Anforderungen des Projektauswahl-Problems angepasst ist. Ich evaluire die Methode in Simulationsexperimenten und einem menschlichen Trainingsexperiment, in dem sie die menschliche Entscheidungsfindung erfolgreich verbessern konnte. Hierauf stelle ich ein Metakognitions-Modell für eine zweite Klasse realer Probleme vor, soziale Dilemmata, welches zur Modellierung einer Vielzahl realer Konfliktsituationen genutzt werden kann.

Die in dieser Dissertation vorgestellten Strategieentdeckungsmethoden und intelligenten Tutorensysteme überwinden mehrere kritische Limitierungen früherer Methoden und ermöglichen die Entdeckung und Vermittlung von höchst komplexen Entscheidungsstrategien in realistischeren Umgebungen als bisher möglich. Dies stellt einen wichtigen Schritt dar, um automatisierte Strategieentdeckungsmethoden zur Entdeckung von optimalen Algorithmen für die Lösung von Problemen in der realen Welt anzuwenden. Die von mir entwickelten intelligenten Tutorensysteme tragen dazu bei, diese Fortschritte in der Algorithmusentdeckung in die reale Welt zu übertragen, und bieten einen vielversprechenden Weg, Menschen Entscheidungsstrategien beizubringen, die sie in ihrem täglichen Leben verwenden können.

Contents

Listing of figures

Acknowledgments

The individual chapters of this dissertation are based on my prior publications Consul et al. (2022), Heindrich et al. (2023), Heindrich and Lieder (2024). Chapter 2 is based on my published article Consul et al. (2022), in which I share first authorship with Saksham Consul. We closely collaborated on developing the main method and running the experiments. In addition to these shared contributions, I led the development of the metacontroller, goal switching, and tree contraction methods while Saksham led the development of the simulation experiments and complexity analysis (partially excluded from this thesis). Jugoslav Stojcheski made minor contributions to the development of the main method, and Falk Lieder supervised the project and contributed to designing the method and experiments. Both Falk Lieder and Saksham Consul collaborated with me on writing the article. Chapter 3 is based on a publicly available preprint Heindrich et al. (2023). Saksham Consul contributed to designing early versions of the main method. Falk Lieder supervised the project, contributed to designing the method and experiments, and collaborated with me on writing the article. Chapter 4 is based on a publicly available preprint Heindrich and Lieder (2024). Falk Lieder supervised the project, contributed to designing the method and experiments, and collaborated with me on writing the article. The mentioned chapters heavily reuse text from the mentioned articles. In addition to the chapters directly based on prior publications, I have reused some elements of the writing from these articles in my Introduction, Background, and Conclusion.

I am deeply grateful to have been given the opportunity to pursue this research. This dissertation would not have been possible without the invaluable support and guidance of many people along the way. First, I want to thank my advisor, Falk Lieder, for his unwavering belief and continuous support of my work. I am also immensely grateful to the additional members of my Thesis Advisory Committee, Peter Dayan and Charley Wu, for their insightful guidance and always ensuring that my work stays on the right track. I am further thankful to the IMPRS-IS for creating an excellent research environment with many opportunities for personal growth. I want to thank my collaborators, Saksham Consul and Jugoslav Stojcheski, whose contributions made the work presented in Chapter 2 possible. Additionally, I want to thank my lab members Julian Skirzyński, Frederic, Becker, Ruiqi He, and Valkyrie Felso for the engaging discussions and support. Lastly, I am deeply thankful to Victoria Eshelby and my family for their encouragement and support throughout my journey.

0

Introduction

Humans often make suboptimal and irrational decisions. Errors in decision-making can be caused by human biases and heuristics that fail to generalize to specific situations (Tversky and Kahneman, 1974), as well as the biological limitations on cognitive resources that make perfectly rational decision-making infeasible (Simon, 1990). While heuristics are generally useful as they allow us to make fast and sufficiently good decisions in many situations (Gigerenzer and Todd, 1999), they can also lead to systematic errors (Kahneman et al., 1991). In many situations, making irrational deci-

sions can have large negative real-world consequences (Tetlock and Mellers, 2002, O'Donoghue and Rabin, 2015), and being able to make better decisions could have a large possible impact on people's lives. This motivates research concerned with improving human decision-making, either by teaching people more efficient decision-making strategies, by creating choice environments in which people instinctively make better decisions, or by augmenting human cognition with the use of computers.

Historically, human rationality is a highly debated topic, where on one side research in cognitive science led to numerous insights in human limitations and suboptimal decision-making caused by biases, heuristics (Kahneman et al., 1991, Tversky and Kahneman, 1974), and cognitive limitations (Simon, 1990), while on the other side economic sciences usually assumed rational decision-makers who act in a way that maximizes their expected utility (Simon, 1956, Tetlock and Mellers, 2002). Today, most people agree that people make suboptimal choices in certain situations. However, there is persistent disagreement about the causes of this suboptimality. Those disagreements matter because the causes of suboptimal decision-making determine what, if anything, should be done to improve it (Stanovich, 2010). One crucial point of disagreement is whether people's decisions are suboptimal because of cognitive limitations versus suboptimal heuristics (Stanovich and West, 2000, Stanovich, 2011). If people used suboptimal heuristics, then it should be sufficient to teach them better decision-making strategies (Larrick et al., 1990, Gigerenzer and Gaissmaier, 2011, Hertwig and Grüne-Yanoff, 2017, Hafenbrädl et al., 2016). But if people's cognitive constraints made good decision-making impossible for them, then one could argue that crucial decisions should be partly outsourced to computers (Marakas, 2003, Lieder et al., 2019b) or policymakers (Thaler and Sunstein, 2021).

One common way to improve human decision-making is nudging: designing choice environments in a way that subconsciously nudges people towards making the better choice (Hertwig and Grüne-Yanoff, 2017). For example, improving the location of salad bars in school cafeterias can substantially increase the amount of fruit and vegetables students consume (Adams et al., 2016). While

nudging has been effective in improving people's choices (Hummel and Maedche, 2019) and widely used across a large range of areas like saving energy (Abrahamse et al., 2005) or healthy eating choices (Bucher et al., 2016, Vecchio and Cavallo, 2019), it is not without controversy. Common criticisms of nudging are that it infringes on peoples' autonomy and can potentially be misused by bad actors to manipulate people's choices (Schmidt and Engelen, 2020). Additionally, since nudging is commonly applied subconsciously and relies on the specifically designed choice environment, it is unclear how well nudges scale to new situations or lead to a sustained long-term behavior change (Beshears and Kosowsky, 2020, Santos Silva, 2022).

An alternative approach is to outsource parts of the decision-making process to computers. Decision support systems aim to augment human decisions by providing additional information, for example through data analysis or estimating the effects of actions (Burstein et al., 2008). Applications of decision support systems include helping doctors make more accurate patient assessments (Payne, 2000) or improving visual object recognition (Fendley and Narayanan, 2012). Completely outsourcing human decisions might seem particularly appealing, given that the application of reinforcement learning algorithms has led to intelligent systems that can outperform people in complex planning problems, such as playing Atari games (Mnih et al., 2013), abstract strategy games like chess and Go (Silver et al., 2016, 2018), and even complex real-time strategy games like StarCraft (Vinyals et al., 2019). While often studied in the context of games, artificial intelligence also has a large potential to aid humans in real-life decisions, for example in the medical domain (Liu et al., 2020). Generally, the public perception of automating decisions in this way has been mostly skeptical due to widespread concerns about the algorithm's fairness and usefulness (Araujo et al., 2020). Additionally, the black-box nature of deep learning raises general safety concerns about the goals internalized by large-scale AI systems (Russell, 2019), which makes outsourcing important decisions to AI problematic.

Improving people's decision competencies by teaching them better decision strategies is known

as *boosting* (Hertwig and Grüne-Yanoff, 2017). In contrast to the first two approaches, boosting protects and enhances people's freedom to make their own decisions based on their own goals, views, and preferences. This is especially important for personal choices that people do not want to give up. Recent work has shown that teaching people clever decision strategies is a promising way to improve their decision-making in the real world (Hafenbrädl et al., 2016). This suggests that it might be possible to enable people to make their own far-sighted decisions in a wide range of complex real-life situations by teaching them effective strategies for long-term planning.

Intelligent tutoring systems (ITS) are a promising approach to facilitate student learning by combining deliberate practice, personalized feedback, and cognitive models of the student's learning progress (Corbett et al., 1997, Graesser et al., 2012). Compared to human tutors, ITS have been shown to be similarly effective (VanLehn, 2011, Anderson et al., 1985), and offer an efficient way to scale personalized tutoring to large numbers of learners (Koedinger et al., 1997). While intelligent tutoring systems in the past have mainly been applied to teach concrete problem-solving skills in subjects like computer science or mathematics (Mousavinasab et al., 2021). More recently, intelligent tutors that aim to improve the metacognitive abilities of learners, for example by teaching them when to seek help, have been proposed as well (Aleven et al., 2006, Guerra and Mellado, 2017, Roll et al., 2005, Callaway et al., 2022a, Chi and VanLehn, 2010). This demonstrates the usefulness of intelligent tutors for boosting human decision-making. Because these tutoring systems usually involve cognitive models of the student's learning progress, they are also called cognitive tutors. Utilizing cognitive models of learners allows the tutoring systems to provide feedback and select practice problems adapted to the student's unique needs (Koedinger and Corbett, 2001).

A key bottleneck to improving human decision-making is that it is usually unknown which cognitive strategies are optimal for a given problem. However, for heuristics to be effective, they have to be well-adapted to the structure of the particular decision environment in which they are used (Simon, 1956, Gigerenzer and Selten, 2002, Todd and Gigerenzer, 2012). The lack of knowledge

about optimal decision strategies has led to past research often relying on hand-crafted strategies (Aleven et al., 2006, Chi and VanLehn, 2010, Hafenbrädl et al., 2016) and limited applications of boosting on more complex sequential decision-problems. On the other hand, early work on boosting taught people the normative principles of probability theory, logic, and expected utility theory (Larrick, 2004), giving learners the theoretical tools to make optimal decisions. Although these educational interventions succeeded to improve people's performance on simple textbook problems, they failed to improve people's performance in the real world (Larrick, 2004). The likely reason is that the amount of mental computation that would be required to apply those normative principles to complex real-world problems far exceeds people's cognitive capacities (Lieder and Griffiths, 2020a).

Resource-rational analysis is a cognitive modeling paradigm that addresses the mismatch between expected utility theory and human limitations in cognitive resources (Lieder and Griffiths, 2020a). By taking not only the quality of decisions, but also the computational costs of reaching them into account, optimal decision-making according to expected utility theory can potentially be reconciled with the previously observed irrationalities in human decision-making. According to this paradigm, good decision-making consists of making efficient use of the available computational resources by balancing the competing objectives of maximizing the expected decision quality and minimizing computational costs. Resource-rationality can be used to describe the quality of a decision strategy by subtracting the cost of the cognitive operations from the expected reward of the resulting decision. Using a model of the decision task, resource-rational analysis also opens up the possibility of comparing the adaptiveness of different decision strategies and heuristics in a principled way.

People are usually not fully resource-rational (Krueger et al., 2024), and figuring out what decision strategies would enable people to perform as well as possible is an important open problem (Callaway et al., 2022a, Mehta et al., 2022). Utilizing resource-rational models of planning tasks makes it possible to develop computational methods that derive optimal planning strategies. This

research area is called *automatic strategy discovery*, and leverages methods from reinforcement learning to automatically discover efficient planning strategies that achieve an optimal trade-off between the decision quality and our limited time and cognitive resources (Callaway et al., 2018b,a).

Being able to algorithmically discover resource-rational planning strategies that perform better than people's intuitive planning strategies opens up the possibility of teaching these strategies to people. This idea has been successfully applied in previous work, where human planning was improved in a fully automated way through training sessions with a cognitive intelligent tutoring system (Callaway et al., 2022a, Mehta et al., 2022) that taught people more adaptive strategies that made better trade-offs between utilizing the information they already possess and acquiring additional (costly) information. Training with the tutor led people to plan in a more resource-rational way than participants who tried to discover good strategies by themselves.

The motivation of this thesis is to extend existing strategy discovery research towards applications closer to the real world. By enabling automatic strategy discovery to be applied to more naturalistic problems, we aim to use the developed methods to discover meta-reasoning strategies that are more effective than the strategies people use by themselves and develop cognitive intelligent tutors to improve people's decision-making. To make progress towards this goal, I have identified the following key limitations in the existing state-of-the-art.

The first limitation of current strategy discovery methods is their scalability. Due to their complexity, current methods are limited to planning problems that only plan a few steps into the future (Callaway et al., 2022a, Mehta et al., 2022, Callaway et al., 2018a). This is because existing methods are highly computationally expensive since they require comparing the expected results from all possible mental planning operations at each step in the planning process, creating a search tree with exponentially increasing complexity. Similar limitations also apply to the cognitive tutors used to teach metareasoning strategies. To provide accurate feedback, existing tutors rely on computationally expensive methods from reinforcement learning that can't be applied to larger planning prob-

lems (Callaway et al., 2022a). These limitations are a key obstacle for applying strategy discovery to problems closer to the real world. More realistic planning problems, like planning one's future career, often require planning over many sequential steps and over large timespans.

A second limitation of existing methods is that they don't capture a core feature of the real world: uncertainty. All prior research was conducted in simplistic planning tasks where the environment is fully observable. However, in real-world applications, planning requires managing high levels of uncertainty. In the real world, thinking about possible outcomes of a future action is unlikely to lead to a completely reliable prediction of its consequences. For example, when planning which bus to take, the precise time of arrival cannot be guaranteed since the bus can be delayed by a wide range of random events, such as traffic jams. Therefore, one needs to make, refine, and plan with uncertain estimates of the bus timings. While general computer algorithms for planning in partially observable environments have been studied for a long time (Monahan, 1982), there is no prior work on discovering metacognitive strategies for human planning in partially observable environments.

Lastly, an anticipated limitation of applying strategy discovery to real-world problems is that real-world scenarios are difficult to model accurately in the metareasoning framework. Improving human decision-making in a real-world task is theoretically possible in two ways: either by modeling the real-world task accurately and teaching people effective metareasoning strategies that they can directly apply when encountering the task again in the real world, or by teaching people generally useful strategies in an abstract setting which they can transfer to real-world problems with a similar structure. While prior work has shown that strategy transfer is possible between similar planning problems (Callaway et al., 2022a), this transfer has not been tested for real-world problems. Generally, no real-world tasks have been modeled as metareasoning problems so far due to the difficulty of estimating the underlying environment parameters, and all existing research has been conducted on highly simplified toy-environments.

The remainder of the introduction will outline how this thesis addresses each of the three key

limitations. Chapter 2 addresses the limitation of limited scalability by introducing hierarchical Bayesian metalevel policy search (hierarchical-BMPS), a novel strategy discovery method building upon BMPS, the existing state-of-the-art method (Callaway et al., 2018a). Hierarchical-BMPS enables strategy discovery to be applied to larger metareasoning problems than previously possible by increasing its computational efficiency. The method achieves this improved efficiency by hierarchically decomposing the planning task into two separate steps: planning which goal to pursue and planning how to achieve the selected goal. The decomposition reduces the computational complexity of solving large planning problems by limiting the number of planning actions that have to be evaluated at once and draws inspiration from the hierarchical structure of human behavior (Botvinick, 2008, Miller et al., 1960, Carver and Scheier, 2001, Tomov et al., 2020). The method is tested in both a computational simulation and three human experiments, where teaching the discovered strategy significantly improved people's efficiency compared to the strategies people used by themselves.

To overcome the limited scalability of intelligent tutoring systems, I present multiple improvements to the cognitive intelligent tutors used in prior work which enable their use on increasingly complex metareasoning environments. In Chapter 3, I propose a computational simplification to the feedback tutor presented by Callaway et al. (2022a) that makes it possible to compute approximate metacognitive feedback in large planning problems where the exact solution is infeasible to compute. Additionally, Chapters 3 and 4 introduce new learning curriculums inspired by shaping (Skinner, 1953) that simplify the learning process by starting with a simplified version of the planning task and incrementally add complexity along multiple dimensions: the number and type of planning operations learners choose between, and the overall size of the planning task.

Chapter 3 extends strategy discovery to partially observable environments and overcomes the second identified limitation: prior strategy discovery can't be applied to environments in which the outcomes of planning operations are uncertain. The chapter presents a metareasoning model

of partially observable planning problems that represents uncertainty in planning operations by only revealing uncertain observations of an action's potential outcomes. Additionally, I introduce the meta-greedy policy for partially observable environments (MGPO), a novel strategy discovery algorithm that discovers efficient strategies in the partially observable setting. MGPO is evaluated twofold: in a simulation experiment where I compare its performance to a computational baseline, and a human experiments where human decision-making is improved by practicing with a cognitive tutor that teaches them the strategy MGPO discovered.

Chapter 4 addresses the third limitation by formalizing project selection, a real-world decision task, as a metareasoning problem. In project selection tasks, a decision-maker or corporation chooses between multiple candidate projects that are usually evaluated by multiple experts across a number of relevant criteria (Coldrick et al., 2002, Khalili-Damghani and Sadi-Nezhad, 2013, Liu et al., 2017). I frame this task as a metareasoning problem in which the decision-maker has to balance which and how much information they request before deciding between projects. To address the unique challenges the project selection task presented for strategy discovery methods, I further present an adapted version of MGPO, the meta-greedy policy for project selection (MGPS). MGPS introduces multiple novel capabilities relevant to the requirements of the real-world setting: integrating information from multiple information sources across multiple relevant criteria. I evaluate MGPS in a simulation experiment and a human training experiment, and demonstrate that the method is capable of significantly improving human decision-making on the project selection task.

Lastly, Chapter 5 describes how an additional class of real-world tasks can be modelled as a metareasoning problems: social dilemmas. The metareasoning task builds upon common game theoretic descriptions of two player games like the prisoners' dilemma (Rapoport and Chammah, 1965). In these games, players usually decide between cooperation and defection in a way that mutual cooperation often leads to the best outcome, but individually, each player has an incentive to defect. The metareasoning model of the task involves evaluating one's options by taking into account how the

other player is likely to act, as well as how the outcome would affect oneself, the other player, and additional parties over long time horizons.

1

Background

Resource-rational analysis (Lieder and Griffiths, 2020a) is a cognitive modeling paradigm that reconciles rational decision-making as described by expected utility theory and the fact that people's cognitive resources and time are limited. To evaluate competing decision strategies, we can score them by their resource-rationality, which takes both the expected utility and the utilized cognitive resources to reach the decision into account. In the resource-rational framework, the decision operations people can perform to arrive at a decision are modeled explicitly and assigned a cost. The

overall efficiency of a decision strategy h in an environment e can then be computed by subtracting the expected costs λ of the N used decision operations from the expected utility R_{total} of the resulting decision (see Equation 1.1). This measure is called resource-rationality score (*RR-score*).

$$\mathrm{RR}(h, e) = \mathbb{E}[R_{total}|h, e] - \lambda\mathbb{E}[N|h, e] \tag{1.1}$$

Equation 1.1 specifies a criterion that optimal heuristics must meet. But it does not directly tell us what those optimal strategies are. Finding out what those optimal strategies are is known as *strategy discovery*. Callaway et al. (2018b) developed a method to automatically derive resource-rational planning strategies by modeling the optimal planning strategy as a solution to a meta-level Markov Decision Process (meta-MDP), an extension of Markov Decision Processes (MDP). MDPs are a general framework used to simulate and evaluate decision-making and described by a tuple $(\mathcal{S}, \mathcal{A}, P, r)$, in which an agent interacts with a (stochastic) environment in state $s \in \mathcal{S}$ by performing actions $a \in \mathcal{A}$ and, in return, receiving a reward $r(s, a)$ and a new environment state s' sampled from the transition function $P(s, a)$.

A meta-level MDP $M = \langle \mathcal{B}, \mathcal{C}, T, r \rangle$ is defined as an undiscounted MDP where $b \in \mathcal{B}$ represents the belief state, $T(b, c, b')$ is the probability of transitioning from belief state b to belief state b' by performing computation $c \in \mathcal{C}$, and $r(b, c)$ is a reward function that describes the costs and benefits of computation (Hay et al., 2014). It is important to note that the actions in a meta-MDP are computations that are different from object-level actions – the former are planning operations and the latter are physical actions that move the agent through the environment.

Previous methods for discovering near-optimal decision strategies (Lieder et al., 2017, Callaway et al., 2018b,a) have been developed for and evaluated in a planning task known as the Mouselab-MDP paradigm (Callaway et al., 2017, 2020). The Mouselab-MDP paradigm was developed to make people's elementary planning operations observable. This is achieved by externalizing the pro-

cess of planning as information seeking. Concretely, the Mouselab-MDP paradigm illustrated in Figure 1.1 shows the participant a map of an environment where each location harbors an occluded positive or negative reward. To find out which path to take, the participant has to click on the locations they consider visiting to uncover their rewards. Each of these clicks is recorded and interpreted as the reflection of one elementary planning operation. The cost of planning is externalized by the fee that people have to pay for each click. People can stop planning and start navigating through the environment at any time. The participant has to follow one of the paths along the arrows to one of the outermost nodes.

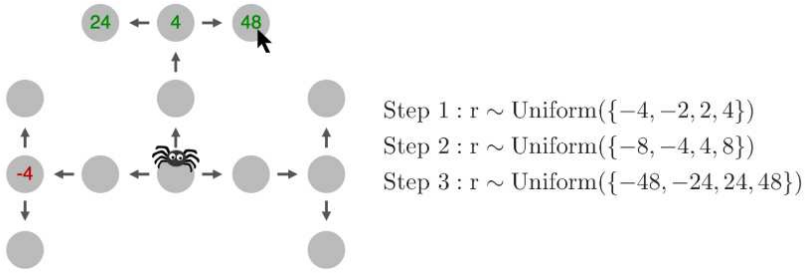


Figure 1.1: Illustration of the Mouselab-MDP paradigm. Rewards are revealed by clicking with the mouse, prior to selecting a path using the keyboard. This figure shows one concrete task that can be created using this paradigm. Many other tasks can be created by varying the size and layout of the environment, the distributions that the rewards are drawn from, and the cost of clicking.

Discovering efficient decision strategies requires solving the meta-level MDP. Different specialized approaches to solving meta-level MDPs have been proposed, such as Hay et al. (2014), Callaway et al. (2018a), Svegliato and Zilberstein (2018), Griffiths et al. (2019). While general methods for solving MDPs such as dynamic programming can also be applied to meta-MDPs, they are often limited to very small planning problems due to the exponential nature of possible belief states. For example, while the environment in Figure 1.1 consists of only 12 nodes that each can take one of 5 possible values, there are $5^{12} = 244140625$ possible belief states. To plan optimally, a strategy discovery algorithm needs to be able to compute which planning operation is optimal in each possible

belief state.

In their seminal paper, Russell and Wefald (1991b) introduced the theory of rational metareasoning, building the theoretical foundation of later metareasoning algorithms. In Russell and Wefald (1991a), they define the value of computation $\mathrm{VOC}(c, b)$ to be the expected improvement in decision quality achieved by performing computation c in belief state b and continuing optimally, minus the cost of computation c. Using this formalization, the optimal planning strategy π^*_{meta} is a selection of computations that maximizes the value of computation (VOC, see Equation 1.2). When the VOC is non-positive for all *available* computations, the policy terminates ($c = \perp$) and executes the best object-level action according to the current belief state. Hence, $\mathrm{VOC}(\perp, b) = 0$. In general, the VOC is computationally intractable, but it can be approximated (Callaway et al., 2018a). Lin et al. (2015) utilize the myopic approximation (VOI_1) to the VOC proposed by Russell and Wefald (1991a), which is the expected improvement in decision quality that would be attained by terminating deliberation immediately after performing the computation. Hay et al. (2014) approximated rational metareasoning by solving multiple smaller meta-MDPs that each define the problem of deciding between one object-level action and its best alternative.

$$\pi^*_{\text{meta}}(b) = \arg\max_c \mathrm{VOC}(c, b) \tag{1.2}$$

Inspired by research on how people learn how to plan (Krueger et al., 2017), Callaway et al. (2018a) developed a reinforcement learning method for learning when to select which computation. This method uses Bayesian optimization to find a policy that maximizes the expected return of a meta-MDP. The policy space is parameterized by weights that determine to which extent computations are selected based on the myopic VOC versus less short-sighted approximations of the value of computation. It thereby improves upon approximating the value of computation by the myopic VOC by considering the possibility that the optimal meta-level policy might perform ad-

ditional computations afterward. Concretely, BMPS approximates the value of computation by interpolating between the myopic value of information and the value of perfect information (see Equation 1.3).

$$\widehat{\mathrm{VOC}}(c, b; \mathbf{w}) = w_1 \cdot \mathrm{VOI}_1(c, b) + w_2 \cdot \mathrm{VPI}(b) + w_3 \cdot \mathrm{VPI}_{\mathrm{sub}}(c, b) - w_4 \cdot \mathrm{cost}(c) \tag{1.3}$$

Equation 1.3 describes how the value of computation is approximated by BMPS, where VOI_1 denotes the myopic value of information and $\mathrm{VPI}(b)$ denotes the value of perfect information. $\mathrm{VPI}(b)$ assumes that all computations possible at a given belief state would take place. Furthermore, $\mathrm{VPI}_{\mathrm{sub}}(c, b)$ measures the benefit of having full information about the subset of parameters that the computation reasons about (e.g., the values of all paths that pass through the node evaluated by the computation), $\mathrm{cost}(c)$ is the cost of the computation c, and $\mathbf{w} = (w_1, w_2, w_3, w_4)$ is a vector of weights. Since the VOC and $\mathrm{VPI}_{\mathrm{sub}}$ are bounded by the VOI_1 from below and by the VPI from above, the approximation of the VOC (i.e. $\widehat{\mathrm{VOC}}$) is a convex combination of these features, and the weights associated with these features are constrained to a probability simplex set. Finally, the weight associated with the cost function $w_4 \in [1, h]$, where h is the maximum number of available computations to be performed. The value of these weights are computed using Bayesian Optimization (Mockus, 2012). Discovery of the optimized weights, is analogous to discovering the optimal policy in the environment. This algorithm has been further extended by Mehta et al. (2022), who applied Bayesian Inference to account for uncertainty in the environment model.

Alternative methods to solve meta-MDPs include works by Sezener and Dayan (2020) and Svegliato and Zilberstein (2018). Sezener and Dayan (2020) solves a multi-arm bandit problem using a Monte Carlo Tree Search based on the static and dynamic value of computations. In a bandit problem, unlike most models of planning, transitions depend purely on the chosen action and not on

the current state. Svegliato and Zilberstein (2018) devised an approximate metareasoning algorithm using temporal difference (TD) learning to decide when to terminate the planning process.

A related task to strategy discovery is program induction, which aims to automatically discover algorithms for programmatic tasks. Similar to the strategy discovery task, the problem of discovering an algorithm that solves a given task can be formulated as an MDP, in which program induction is modeled as a search problem over the elemental operations of the programming language (Ellis et al., 2019). Recent work on program induction has utilized deep reinforcement learning to discover algorithms that outperformed their human designed complements, for example faster matrix multiplication algorithms (Fawzi et al., 2022) or sorting algorithms (Mankowitz et al., 2023).

Externalizing human planning with the Mouselab-MDP paradigm and discovering optimal planning strategies using strategy discovery algorithms opens up the possibility of teaching the optimal planning strategies to people. Prior work has successfully applied this idea by developing cognitive tutors, a type of intelligent tutoring system that teaches planning strategies by giving learners metacognitive feedback (Callaway et al., 2022a). The tutor let people practice planning in the Mouselab-MDP paradigm and provided them immediate feedback on each chosen planning operation. The feedback relies on an exact solution to the meta-MDP and is given in two ways: (1) information about what the optimal planning strategy would have done; and (2) an affective element given as positive feedback (e.g., "Good job!") or negative feedback. The negative feedback included a slightly frustrating time-out penalty, during which participants were forced to wait idly for a duration that was proportional to how suboptimal their planning operation had been. Using the optimal solution to the meta-MDP, Skirzyński et al. (2021) developed a tutor that teaches strategies using automatically generated human-interpretable descriptions, which further improved learning compared to metacognitive feedback.

Summarizing the current state of the art, prior work has shown that (1) it is possible to model human metareasoning as a meta-MDP, which allows measuring the resource-rationality of internal

planning strategies (Callaway et al., 2017), (2) strategy discovery methods are capable of automatically discovering near-optimal planning strategies (Callaway et al., 2018a), and (3) teaching people how to use algorithmically discovered metareasoning strategies with intelligent cognitive tutors can significantly improve the resource-rationality of human planning (Callaway et al., 2022a). The remainder of this thesis will address how I overcome the key limitations that prevent current work in automated strategy discovery to be applied to more complex and realistic planning problems: scaling strategy discovery and intelligent cognitive tutors to larger and partially observable planning problems, and modeling realistic planning problems as meta-MDPs.

2

Hierarchical strategy discovery

This chapter introduces hierarchical-BMPS, a scalable strategy discovery method, and demonstrates its usefulness in computer simulations and human training experiments. The chapter is based off my published article "Improving Human Decision-making by Discovering Efficient Strategies for Hierarchical Planning" Consul et al. (2022), which I co-authored with my collaborators Saksham Consul, Jugoslav Stojcheski, and Falk Lieder.

2.1 Introduction

To compute optimal decision strategies, strategy discovery methods would have to compare the utilities of all possible sequences of computations in each step. As such, these algorithms have to explore the entire meta-MDP's state space, which grows exponentially with the number of nodes. As a consequence, exact solution methods like dynamic programming are extremely limited in their ability to scale to problems with larger state spaces or long planning horizons. Existing strategy discovery methods (Callaway et al., 2018a, Russell and Wefald, 1991a, Hay et al., 2014, Lin et al., 2015) instead approximate optimal strategy discovery, usually by only planning ahead for a single time step. The current state-of-the-art strategy discovery method Bayesian Metalevel Policy Search (BMPS) (Callaway et al., 2018a) evaluates groups of computations at once to approximate planning multiple steps ahead, but still computes this approximation for every possible computation at every step. While this approximation is already much more scalable than dynamic programming, it is still highly limited in the size of environment the method can be applied to. Estimating the utility of all available computations is computationally expensive and also not representative of how people plan for complex real-world tasks. In contrast to the exhaustive enumeration of all possible planning operations performed by those methods, people would not even consider making detailed low-level motor plans for navigating to a specific distant location (e.g., Terminal C of San Juan Airport) until they arrive at a high-level plan that leads them to or through that location (Tomov et al., 2020). In this chapter, I build on insights about human planning to develop a more scalable method for discovering efficient planning strategies.

To overcome the computational bottleneck of existing strategy discovery methods, my collaborators and I developed a scalable method for discovering planning strategies. The strategies our method discovers are more adaptive at deciding which and how much information to acquire than humans on some of the planning problems that are too large for existing strategy discovery meth-

ods. The approach draws inspiration from the hierarchical structure of human behavior (Botvinick, 2008, Miller et al., 1960, Carver and Scheier, 2001, Tomov et al., 2020). Research in cognitive science and neuroscience suggests that the brain decomposes long-term planning into goal-setting and planning at multiple hierarchically nested timescales (Carver and Scheier, 2001, Botvinick, 2008). Furthermore, Solway et al. (2014) found that human learners spontaneously discover optimal ways to divide tasks into smaller subtasks. Inspired by these findings, my collaborators and I extended the near-optimal strategy discovery method proposed in Callaway et al. (2018a) by incorporating hierarchical structure into the space of possible planning strategies. Concretely, the planning task is decomposed into first selecting one of the possible final destinations as a goal solely based on its own value and then planning the path to this selected goal.

My collaborators and I found that imposing hierarchical structure makes automatic strategy discovery methods significantly less computationally expensive without compromising the resource-rationality score of the discovered strategies on the test environments. The hierarchical decomposition led to a substantial reduction in the computational complexity of the strategy discovery problem, which made it possible to scale up automatic strategy discovery to many planning problems that were prohibitively large for previous strategy discovery methods. This allowed my method to discover planning strategies that achieve a super-human level of computational efficiency on non-trivial planning problems, in the sense that the discovered strategies had a higher resource-rationality score than the strategies used by people.

The remainder of this chapter presents the method and results of the hierarchical strategy discovery method. First, I present hierarchical-BMPS, the new reinforcement learning method for discovering hierarchical planning strategies. Next, I describe how my collaborators and I demonstrated that the method can discover near-optimal planning strategies for larger environments than previous methods were able to handle and characterize these optimal strategies. Lastly, I show that the resulting advances are sufficient to improve human decision-making in complex planning problems by

teaching people the discovered strategies using video demonstrations.

2.2 Hierarchical problem decomposition

To efficiently plan over long time horizons, people (Botvinick, 2008, Carver and Scheier, 2001, Tomov et al., 2020) and hierarchical planning algorithms (Kaelbling and Lozano-Pérez, 2010, Sacerdoti, 1974, Marthi et al., 2007, Wolfe et al., 2010) decompose the problem into first setting goals and then planning how to achieve them. This two-stage process breaks large planning problems down into smaller problems that are easier to solve. To discover hierarchical planning strategies automatically, our proposed strategy discovery algorithm decomposes the problem of discovering planning strategies into the sub-problems of discovering a strategy for selecting a goal and discovering a strategy for planning the path to the chosen goal. A pictorial representation is given in Figure 2.1.

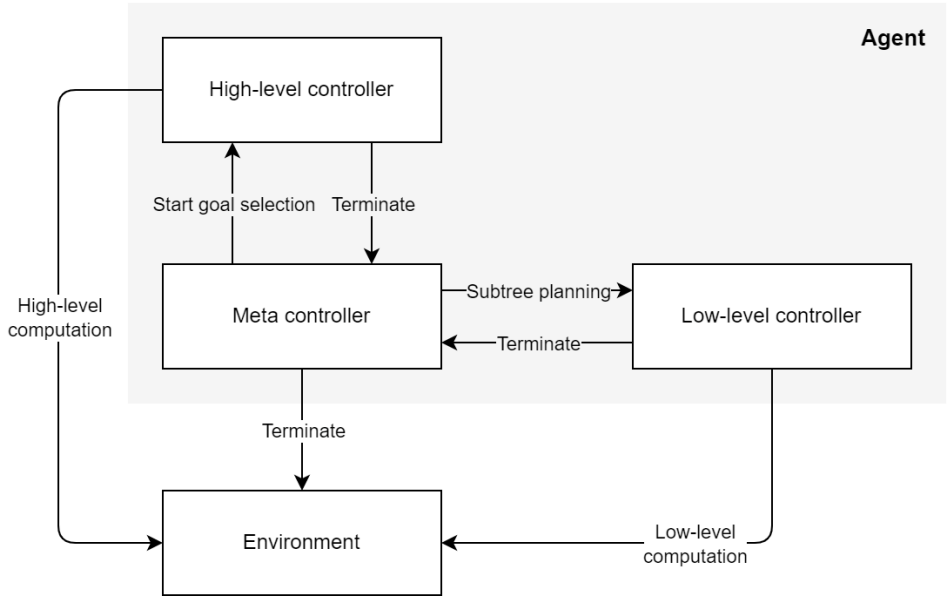


Figure 2.1: Flow diagram of the hierarchically discovered planning strategy. The high-level controller decides on goal computations. The low-level controller decides the computations to be performed within a goal. The meta controller switches between high-level and low-level controller and terminates planning.

Formally, this is accomplished by decomposing the meta-MDP defining the strategy discovery problem into two meta-MDPs with smaller state and action spaces. Constructing meta-MDPs for goal-setting and path planning is easy when there is a small set of candidate goals. Such candidate goals can often be identified based on prior knowledge or the structure of the domain (Schapiro et al., 2013, Solway et al., 2014). A low-level controller solves the low-level MDP whereas the high-level controller solves the high-level MDP. When a controller is in active, it selects and performs a computation from the corresponding meta-MDP. The meta controller looks at the expected reward of the current goal with the expected reward of the next best goal and decides when control from the high-level controller should be switched to the low-level controller. Hence, when the low-level controller discovers that the current goal is not as valuable as expected, the meta controller allows for goal switching. This is implemented by comparing the current best low-level path with the average expected reward of the best alternative goal, and returning to the goal-selection stage when the expected reward of the current best path is lower.

The high-level meta-MDP model of the sub-problem of goal selection (Section 2.2.1) only includes computations for estimating the values of a small set of candidate goal states ($V(g_1), \cdots, V(g_M)$). This means that goals are chosen without considering how costly it would be to achieve them. This makes sense when all goals are known to be achievable and the differences between the values of alternative goal states are substantially larger than the differences between the costs of reaching them. This is arguably true for many challenges people face in real life. For instance, when a high school student plans one's career, the difference between the long-term values of studying computer science versus becoming a janitor is likely much larger than the difference between the costs of achieving either goal. This is to be expected when the time it will take to achieve the goals is short relative to a person's lifetime.

The low-level MDP (Section 2.2.2) only includes computations that update the estimated costs of alternative paths to the chosen goal by determining the costs or rewards of state-action pairs

$r(s, a)$ that lie on those paths. Extending upon the example of pursuing a computer science degree, low-level actions towards this goal could be deciding between different universities. This selection of computations within a selected goal leads to a possible issue of ignoring some computations that can be irrelevant in the low-level MDP but be highly valuable when considering the complete problem. One such example is when considering computations that reveal the value of nodes lying on an unavoidable path to the selected goal. This problem gets further accentuated if such a node has a possibility of having a highly positive or negative reward. To rectify this problem, a meta controller has been introduced to facilitate goal switching. A real-world example of the necessity to switch goals after discovering an unlikely highly negative event could be, for example, to switch from investing in the stock market to investing in real estate after discovering a likely stock market crash.

Decomposing the strategy discovery problem into these two components reduces the number of possible computations that the metareasoning method has to choose between from $M \cdot N$ to $M + N$, where M is the number of possible final destinations (goals) and N is the number of steps to the chosen goal. Perhaps the most-promising metareasoning method for automatic strategy discovery is the Bayesian Metalevel Policy Search algorithm (BMPS; (Callaway et al., 2018a, Kemtur et al., 2020)). To solve the two types of meta-MDPs introduced below more effectively, my collaborators and I also introduce an improvement of the BMPS algorithm in Section 2.3. An additional computational optimization is described in Section 2.4.

2.2.1 Goal-setting meta-MDP

The optimal strategy for setting the goal can be formalized as the solution to the meta-MDP $\mathcal{M}^{\mathrm{H}} = \langle \mathcal{B}^{\mathrm{H}}, \mathcal{C}^{\mathrm{H}}, T^{\mathrm{H}}, R^{\mathrm{H}} \rangle$, where the belief state $b^{\mathrm{H}}(g) \in \mathcal{B}^{\mathrm{H}}$ denotes the expected cumulative reward that the agent can attain by achieving goal state $g \in \mathcal{G}$. The high-level computations are $\mathcal{C}^{\mathrm{H}} = \{\xi_1, ..., \xi_M, \perp^{\mathrm{H}}\}$, where ξ_g reveals the value $V(g)$ of the goal node g. $\perp^{\mathrm{H}}$ terminates the high-level planning, leading to the agent to select the goal with the highest value according to its current belief

state. The reward function is $R^{\mathrm{H}}(b^{\mathrm{H}}, c^{\mathrm{H}}) = -\lambda^{\mathrm{H}}$ for $c^{\mathrm{H}} \in \{\xi_1, ...\xi_M\}$ and $R^{\mathrm{H}}(b^{\mathrm{H}}, \perp^{\mathrm{H}}) = \max_{k \in \mathcal{G}} \mathbb{E}[b^{\mathrm{H}}(k)]$.

2.2.2 Low-level meta-MDP

Having set a goal to pursue, the agent has to find the optimal planning strategy to achieve the goal. This planning strategy is formalized as the solution to the meta-MDP $\mathcal{M}^{\mathrm{L}} = \langle \mathcal{B}^{\mathrm{L}}, \mathcal{C}^{\mathrm{L}}, T^{\mathrm{L}}, R^{\mathrm{L}} \rangle$, where the belief state $b \in \mathcal{B}^{\mathrm{L}}$ denotes the expected reward for each node. The agent can only perform a subset of meta-actions $\mathcal{C}_{g,\mathrm{L}} = \{c_{g,1}, ..., c_{g,N}, \perp^{\mathrm{L}}\}$, where $c_{g,n}$ reveals the reward at node n in the goal set $h_g \in \mathcal{H}$. A goal set $h_g \in \mathcal{H}$ refers to all nodes, including the goal node, which lie on all paths leading to goal $g \in \mathcal{G}$. Furthermore, $\perp^{\mathrm{L}}$ terminates planning and leads the agent to select the path with the highest expected sum of rewards according to the current belief state. The reward function is $R^{\mathrm{L}}(b, c_g) = -\lambda^{\mathrm{L}}$ for $c_g \in \{c_{g,1}, ..., c_{g,N}\}$ and $R^{\mathrm{L}}(b, \perp^{\mathrm{L}}) = \max_{p \in \mathcal{P}} \sum_{n \in p} \mathbb{E}[b_n]$, where $\mathcal{P}$ is the set of all paths, and b_n is the belief of the reward for node n.

2.3 Hierarchical Bayesian Metalevel Policy Search

Having introduced the hierarchical problem decomposition, I now present how this decomposition can be leveraged to make BMPS and other automatic strategy discovery methods more scalable. BMPS approximates the value of computation (VOC) according to Equation 1.3. My collaborators and I propose to utilize BMPS to solve the high-level meta-MDP and the low-level meta-MDP separately. The meta controller decides which meta-MDP should be investigated and when planning should terminate.

High-level policy search The VOC for the high-level policy is approximated using three features: (1) the myopic utility for performing a goal state evaluation ($\mathrm{VOI}^{\mathrm{H}}_1$), (2) the value of perfect

information about all goals (VPI^{H}), and (3) the cost of the respective computation (cost^{H}).

$$\widehat{\text{VOC}}^{\text{H}}\left(c^{\text{H}}, b^{\text{H}}; \mathbf{w}^{\text{H}}\right) = w_1^{\text{H}} \cdot \text{VOI}_1^{\text{H}}\left(c^{\text{H}}, b^{\text{H}}\right) + w_2^{\text{H}} \cdot \text{VPI}^{\text{H}}\left(b^{\text{H}}\right) - w_3^{\text{H}} \cdot \text{cost}^{\text{H}}\left(c^{\text{H}}\right), \tag{2.1}$$

where $w_1^{\text{H}}, w_2^{\text{H}}$ are constrained to a probability simplex set, $w_3^{\text{H}} \in \mathbb{R}_{[1,M]}$, and M is the number of goals. Additionally, the cost $\text{cost}^{\text{H}}(c^{\text{H}})$ is defined as

$$\text{cost}^{\text{H}}(c^{\text{H}}) = \begin{cases} \lambda^{\text{H}}, & \text{if } c^{\text{H}} \in \{\xi_1, ..., \xi_M\}. \\ 0, & \text{if } c^{\text{H}} = \perp^{\text{H}}. \end{cases} \tag{2.2}$$

Low-level policy search Similarly as for the high-level policy, the value of computation for the low-level policy is approximated by using a mixture of VOI features and the anticipated cost of the current computation and future computations, that is:

$$\begin{aligned} \widehat{\text{VOC}}^{\text{L}}(c, b, g; \mathbf{w}^{\text{L}}) = w_1^{\text{L}} \cdot \text{VOI}_1^{\text{L}}(c, b, g) + w_2^{\text{L}} \cdot \text{VPI}^{\text{L}}(b, g) \\ + w_3^{\text{L}} \cdot \text{VPI}_{\text{sub}}^{\text{L}}(c, b, g) - w_4^{\text{L}} \cdot \text{cost}^{\text{L}}(c, g, \mathbf{w}^{\text{L}}) \end{aligned} \tag{2.3}$$

, where the weights $w_1^{\text{L}}, w_2^{\text{L}}, w_3^{\text{L}}$ are constrained to a probability simplex set, $w_4^{\text{L}} \in \mathbb{R}_{[1,|h_g|]}$, and $|h_g|$ is the number of nodes in goal set h_g. The weight values for both levels are optimized in 100 iterations with Bayesian Optimization (Mockus, 2012) using the GPyOpt library (The GPyOpt authors, 2016).

The cost feature of the original BMPS algorithm introduced by Callaway et al. (2018a) only considered the cost of a single computation, whereas its VOI features consider the benefits of performing a sequence of computations. As a consequence, policies learned with the original version of BMPS are biased towards inspecting nodes that many paths converge on, even when the values of those nodes are irrelevant. To rectify this problem, my collaborators redefine the cost feature so that

it considers the costs of all computations assumed by the VOI features. Concretely, to compute the low-level policy, my collaborators and I define the cost feature of BMPS as the weighted average of the costs of generating the information assumed by the VOI features $\mathcal{F} = \{\text{VOI}_1^{\text{L}}, \text{VPI}^{\text{L}}, \text{VPI}_{\text{sub}}^{\text{L}}\}$, that is

$$\text{cost}^{\text{L}}(c, g, \mathbf{w}^{\text{L}}) = \sum_{f \in \mathcal{F}} w_f^{\text{L}} \cdot \sum_{n}^{|b_g|} \mathbb{I}(c, f, n) \cdot \text{cost}(c) \tag{2.4}$$

where $\mathbb{I}(c, f, n)$ returns 1 if node n is relevant when computing feature f for computation c and 0 otherwise.

2.4 Tree contraction method for faster BMPS feature computation

To further increase the scalability of BMPS, I make an additional improvement to how it computes the features used to approximate the value of computation (Callaway et al., 2018a). Specifically, I improve the computational efficiency by combining nodes in the meta MDP according to a set of predefined conditions, ultimately reducing the complexity of the necessary computations. The node combination is performed by merging two nodes into a single new node with a probability distribution that represents their combined reward value. The aggregated state is then used to speed up the calculation of VOC features and performed online for each considered feature. The main planning operations are still performed in the full, uncontracted meta MDP, the contraction is only applied to the VOC feature calculation.

The algorithm consists of three different operations that combine node distributions. A list of conditions determines an operation to apply and the algorithm stops when the distributions of all nodes within the MDP are collapsed to a single root node. Graphical examples of how the contraction operations are applied can be found in Figure 2.2.

- **Add**: Combines the distribution of two consecutive nodes by adding their distributions.

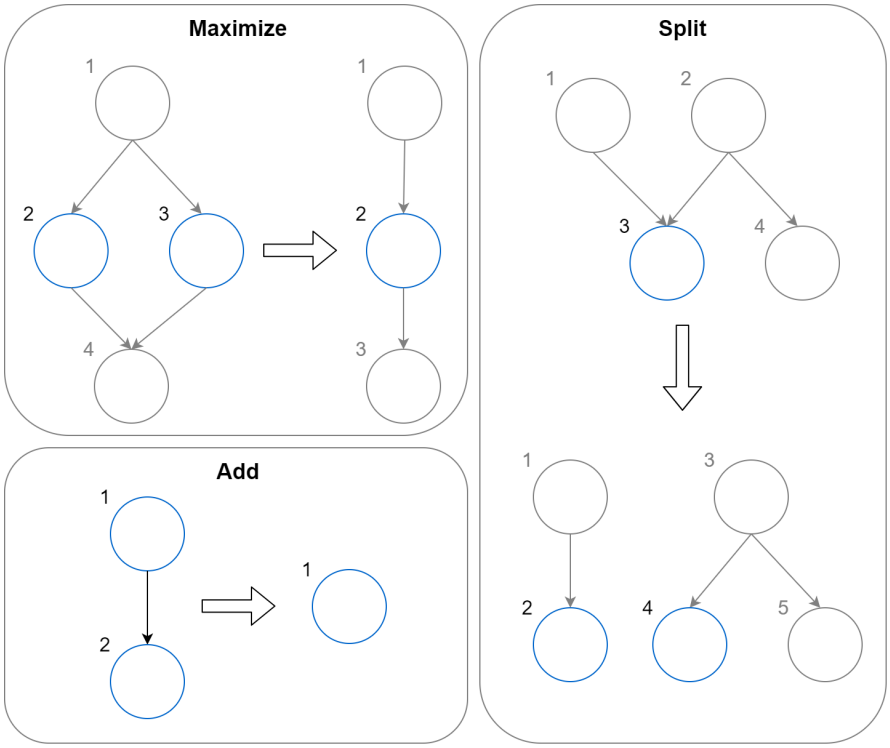


Figure 2.2: Graphical examples of the three contraction steps: Add, Maximize, and Split. Blue nodes highlight which nodes in the tree are affected by the contraction step while greyed out nodes remain unchanged.

Two nodes are consecutive if they are in a direct parent-child relation. This operation can be applied to two consecutive nodes in the tree, as long as the parent node does not have other child nodes and the child node does not have other parents. The two nodes are combined by taking the Cartesian product of their categorical reward distribution, where the rewards are added together and the probabilities are multiplied.

Example: Assuming both node 1 and 2 in the *Add* example of Figure 2.2 follow a categorical distribution with possible outcomes $\{-5\ (p = 0.5), 5\ (p = 0.5)\}$, there are four possible combinations of node values: $\{(-5, -5), (-5, 5), (5, -5), (5, 5)\}$. Adding the reward values and multiplying the corresponding probabilities results in the categorical distribution for the combined node: $\{-10\ (p = 0.25), 0\ (p = 0.5), 10\ (p = 0.25)\}$.

- **Maximize**: Combines two parallel nodes by taking the maximum value for each combination of values of nodes can take, combining the nodes distributions while taking into account that the optimal path will always lead through the higher node of the two. This operation can be applied to two nodes that have a single identical parent and, optionally, child node. For this operation, when combining the two nodes' categorical reward distribution, the maximum of the reward values is used for each combination of outcomes. The probabilities of the outcomes are multiplied.

  Example: Similar to the example for the add operation, I assume both node 2 and 3 in the *Maximize* example of Figure 2.2 follow a categorical distribution with possible outcomes $\{-5\ (p = 0.5), 5\ (p = 0.5)\}$ with four possible combinations of node values: $\{(-5, -5), (-5, 5), (5, -5), (5, 5)\}$. For each combination, the maximum value is used to create a new node: $\{-5\ (p = 0.25), 5\ (p = 0.75)\}$.

- **Split**: Splits a child node into two separate nodes by duplicating that node. The whole tree is then duplicated as many times as the node has possible values, fixing the node's distribution to each possibility. The duplicated trees are then individually reduced to single root nodes, and the individual root nodes are combined to a single tree by pairwise application of the add operation. This operation can be applied to nodes that have multiple parent nodes, where each of the individual nodes after splitting is only connected to one of its parent nodes.

  Example: I again define node 3 in the *Split* example of Figure 2.2 to follow a categorical distribution with possible outcomes $\{-5\,(p = 0.5), 5\,(p = 0.5)\}$. Applying the *Split* operation results in two copies of the shown tree structure with 5 nodes. In one of the two copies, both node 2 and 4 will be assigned the value -5 and in the other copy both nodes will be assigned the value 5. This operation simplifies the tree structure in a way that allows subsequent applications of *Add* and *Maximize* rules.

The split operation is the most computationally expensive operation and is therefore only applied when the add and maximize operations are insufficient to reduce the tree to a single node. Specifically, this happens when a node that needs to be reduced by the multiply operation has an additional parent or child node. Since the structure of the environment stays identical while the rewards and discovered states vary, I precompute the necessary operations to reduce the tree and then apply the reduction individually for each problem instance.

The adjustment is purely algorithmic and does not change the value of computation. Therefore, it does not impair the performance of the discovered strategies. An additional effect of the tree contraction method is that it extends the types of environments solvable by BMPS. Previously, BMPS was only able to handle environments with a branching tree structure: nodes can have multiple children but never multiple parents. The new formulation allows computing the BMPS features for tree structures in which nodes have multiple parent nodes as well. This is possible through the application of the maximize operation, which allows combining multiple parent nodes into a single node, making them solvable through the value of computation calculation. The range of solvable environments is therefore extended from trees to directed acyclic graphs. This extension is especially relevant for environments containing goal nodes, since it is often the case that multiple intermediate nodes converge to the same goal node. The tree contraction improvement is compatible with both hierarchical-BMPS formulation and the original version of BMPS. In the following evaluations, I applied tree contraction to the original BMPS version as well to allow comparisons on environments hierarchical-BMPS was unable to solve before.

2.5 Evaluating the performance, scalability, and robustness of our method for discovering hierarchical planning strategies

I evaluated the new method for discovering resource-rational strategies for human planning in large and complex sequential decision problems in terms of the resource-rationality of the discovered strategies and the scope of its applicability. Concretely, I measure the resource-rationality of the discovered strategies by the resource-rationality score defined in Equation 1.1. In brief, my collaborators and I found that our new method makes it possible to discover resource-rational strategies for substantially larger, and hence more naturalistic, sequential decision problems than previous strategy discovery methods could handle. Despite the approximations that went into making our method so scalable, the discovered strategies are no worse than the strategies discovered by state-of-the-art strategy discovery methods and substantially more resource-rational than both the strategies that people use intuitively and standard planning algorithms that people could be thought to use instead. These findings hold even when the structure of the sequential decision problem violates the assumptions of our method. This section presents the details of this evaluation. Readers who are primarily interested in the effects of teaching the automatically discovered strategies to people may skip ahead to the next section.

To evaluate our method for discovering resource-rational strategies for human planning in large sequential decision problems, my collaborators and I benchmarked its performance in the two types of environments illustrated in Figure 2.3 and Figure 2.6. The first type of environment conforms to the structure that motivated our hierarchical problem decomposition (i.e., the variability of rewards increases from each step to the next) and the second type does not. In the second type of environment, I introduced a high-risk node with a small chance of incurring a highly negative reward on the path to each goal (see Figure 2.6). This violates the assumption that motivated the hierarchical decomposition and makes goal switching essential for a high resource-rationality score.

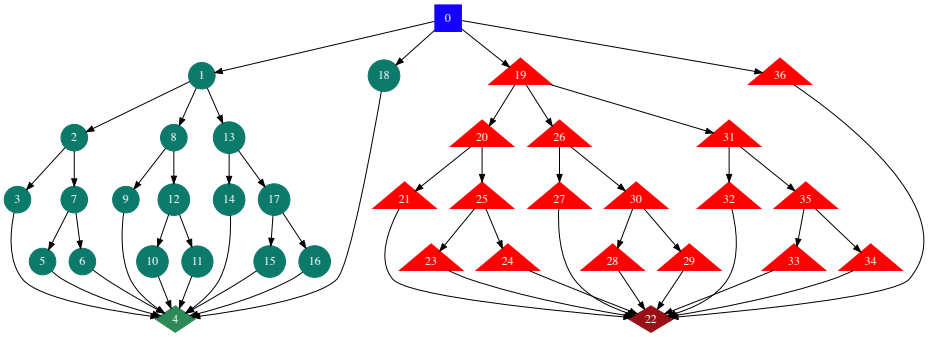


Figure 2.3: Mouselab-MDP environment with 2 goals. Nodes associated with each goal are denoted in green (circles) and red (triangles), respectively. The goal nodes have darker shades of green and red (diamonds), and the root node's color is blue (square).

For each environment, my collaborators and I assess the degree to which the automatically discovered strategies are resource-rational against the resource-rationality of human planning, existing planning algorithms, and the strategies discovered by state-of-the-art strategy discovery methods. The Mouselab MDP planning task and the corresponding meta-MDP are set up in such a way that the scores of people and automatically discovered strategies measure their level of resource-rationality rather than just their performance. This is because the score in this task is the sum of the external rewards along the chosen path minus the cost of planning that was invested to select the path. I therefore refer to it as the *resource-rationality score*. A strategy will achieve the highest resource-rationality score if it selects the most informative planning operations (computations/clicks) to maximally improve its plan with as few computations as necessary. Each planning operation has an associated cost, which is used to model the cost of thinking. A strategy is said to be the most computationally efficient if it achieves the highest possible resource-rationality score.

In addition to evaluating the resource-rationality of the discovered strategies, my collaborators and I also evaluated the scalability of our method to larger planning problems with longer horizons in terms of the largest environments for which it can discover resource-rational heuristics.

Hierarchical-BMPS can solve meta-MDPs with many times more possible belief states than what previous methods were able to handle. This improvement increases the size of the largest problem for which resource-rational planning strategies can be discovered to planning 5 steps ahead when there are five possible actions in each state that each lead to a different outcome.

Analysis of the speed-up achieved by the tree-contraction method Table 2.1 shows an example computation of a single VPI feature (Callaway et al., 2018a) computation. The computational speedup allows us to solve larger environments in the same amount of time and contributes to scaling the algorithm to more realistic problems.

Environment size	With tree contraction	Without tree contraction
Branching (3,3,3), 27 nodes	0.004s	0.018s
Branching (5,5,5), 125 nodes	0.007s	12.27s
Branching (6,6,6), 216 nodes	0.013s	328.22s

Table 2.1: Comparison of computation time for a single VPI feature calculation on different environment sizes. The environments follow a three-step branching structure where all nodes in the environment have 3, 5 or 6 child nodes depending on the environment size.

2.5.1 Evaluation in environments that conform to the assumed structure

My collaborators and I first evaluate the performance of our method in environments whose structure conforms to the assumptions that motivated the hierarchical problem decomposition. To do so, I compare the resource-rationality of the discovered strategies against the resource-rationality of existing planning algorithms, the strategies discovered by previous strategy discovery methods, and human resource-rationality in four increasingly challenging environments of this type with 2-5 candidate goals. The reward of each node is sampled from a normal distribution with mean 0. The variance of rewards available at non-goal nodes was 5 for nodes reachable within a single step (level 1) and doubled from each level to the next. The variance of the distribution from which the reward

associated with the goal node was sampled starts from 100 and increases by 20 for every additional goal node, up to a maximum of 180, after which the variance resets to 100. The environment was partitioned into one sub-graph per goal. Each of those sub-graphs contains 17 intermediate nodes, forming 10 possible paths that reach the goal state in at most 5 steps (see Figure 2.3). The cost of planning is 1 point per click ($\lambda = 1$).

To estimate an upper bound on the resource-rationality score of existing planning algorithms on the benchmark problems, my collaborators and I selected Backward Search and Bidirectional Search (Russell and Norvig, 2002) because – unlike most planning algorithms – they start by considering potential final destinations, which is optimal for planning in our benchmark problems. These search algorithms terminate when they find a path whose expected return exceeds a threshold, called its aspiration value. The aspiration was selected using Bayesian Optimization (Mockus, 2012) to get the best possible performance from the selected planning algorithm. My collaborators and I also evaluated the resource-rationality score of a random-search algorithm, which chooses computations uniformly at random from the set of meta-level operations that have not been performed yet.

In addition to those planning algorithms, our baselines also include three state-of-the-art methods for automatic strategy discovery: the greedy myopic VOC strategy discovery algorithm (Lin et al., 2015), which approximates the VOC by its myopic utility (VOI_1), BMPS (Callaway et al., 2018a), and the Adaptive Metareasoning Policy Search algorithm (AMPS) (Svegliato and Zilberstein, 2018) which uses approximate metareasoning to decide when to terminate planning. My collaborators implemented the AMPS algorithm using a deep Q-network (Mnih et al., 2013) to estimate the difference between values to stop planning and continue planning, respectively. It learns this estimate based on the expected termination reward of the best path. The planning operations are selected by maximizing the myopic value of information (VOI_1). When applying hierarchical-BMPS to this environment, goal switching is ineffective since the cumulative variance of the intermediate nodes was less than the variance of the goal nodes, rendering goal-switching unnecessary.

Therefore, goal switching does not occur in the automatically discovered strategies for solving this environment. To illustrate the versatility of the hierarchical problem decomposition, my collaborators and I also applied it to the greedy myopic VOC strategy discovery algorithm.

2.5.2 How resource-rational are the automatically discovered strategies?

Table 2.2 and Figure 2.5 compare the resource-rationality score of the strategies discovered by hierarchical-BMPS and the hierarchical greedy myopic VOC method against the resource-rationality score of the strategies discovered by the two state-of-the-art methods, two standard planning algorithms, and human resource-rationality scores (see Section 2.5.4) on the benchmark problems described above (Section 2.5.1). These results show that the strategies discovered by hierarchical-BMPS* outperform extant planning algorithms and the strategies discovered by the AMPS algorithm across all benchmark problems ($p < .01$ for all pairwise Wilcoxon rank-sum tests). Critically, imposing hierarchical constraints on the strategy search of BMPS and the greedy myopic VOC method had no negative effect on the resource-rationality score of the resulting strategies ($p > .770$ for all pairwise Wilcoxon rank-sum tests).

As illustrated in Figure 2.4, the planning strategy our hierarchical-BMPS algorithm discovered for this type of environment is qualitatively different from all existing planning algorithms. In general, the strategy is as follows: it first evaluates the goal nodes until it finds a goal node with a sufficiently high reward. Then, it plans backward from the chosen goal to the current state. In evaluating candidate paths from the goal to the current state, it discards each path from further exploration as soon as it encounters a high negative reward on that path. This phenomenon is known as pruning and has previously been observed in human planning (Huys, et al., 2012). While this strategy was discovered assuming that the cost of evaluating a potential goal node is the same as the cost of evalu-

*The performance with and without goal switching is identical due to the lack of goal switching performed by the discovered optimal strategy in the given environment structure.

Type	Name	2 Goals	3 Goals	4 Goals	5 Goals
S	**hierarchical-BMPS**	108.79	**150.63**	178.98	206.45
S	**Non-hierarchical-BMPS**	**111.53**	148.01	**182.38**	204.37
S	**Hierarchical greedy myopic VOC**	108.48	150.13	178.81	**206.57**
S	**Non-hierarchical greedy myopic VOC**	107.98	150.41	180.35	205.40
S	Adaptive Metareasoning Policy Search	77.08	109.39	127.01	141.34
P	Depth-first Search	74.99	109.13	129.66	143.45
P	Breadth-first Search	87.62	112.83	127.68	137.40
P	Bidirectional Search	88.59	115.07	134.08	154.24
P	Backward Search	87.85	114.29	134.43	156.56
P	Random Policy	52.73	80.05	89.31	101.15
	Human Baseline	45.42	88.06	39.32	124.89

Table 2.2: Resource-rationality scores of various strategy discovery methods (*S*) and existing planning algorithms (*P*). The highest resource-rationality score for each environment setting (column) is formatted in **bold**. The four best methods performed significantly better than the other methods, but the differences between them are not statistically significant.

ating an intermediate node, my collaborators and I found that the discovered strategy remained the same as we increased the cost of evaluating goal nodes to 2, 5, or 10. The non-hierarchical version of BMPS also discovered this type of planning strategy. This suggests that goal-setting with backward planning is the resource-rational strategy for this environment, rather than an artifact of our hierarchical problem decomposition. Unlike this type of planning, most extant planning algorithms plan forward and the few planning algorithms that plan backward (e.g., Bidirectional Search and Backward Search) do not preemptively terminate a path exploration.

2.5.3 Evaluation in risky environments

To accommodate environments whose structure violates the assumption that more distant rewards are more variable than more proximal ones, the hierarchical strategies discovered by our method can alternate between goal selection and goal planning. I now demonstrate the benefits of this goal

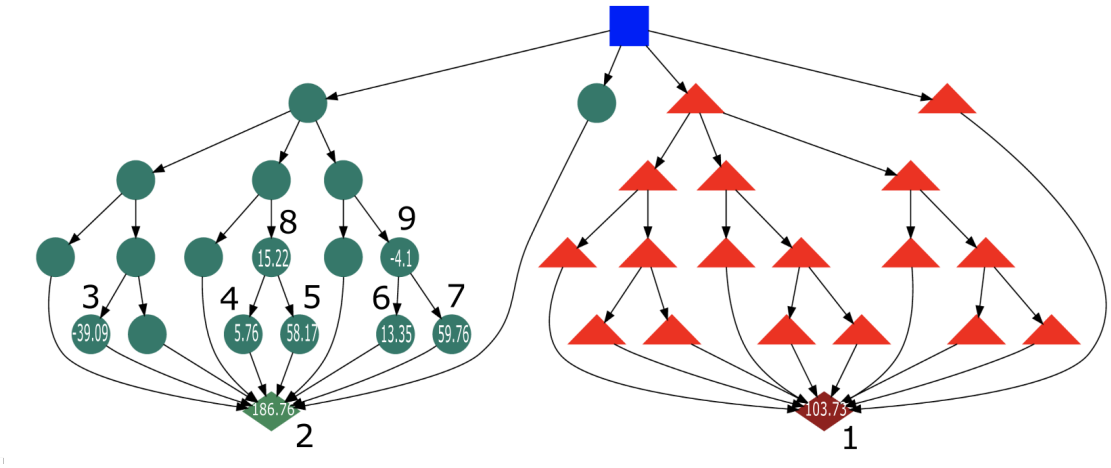


Figure 2.4: Sequence of nodes revealed in particular environment. The numbers above the nodes indicate the sequence in which the nodes were revealed. The numbers in each revealed node indicates its reward.

switching functionality by comparing the resource-rationality score of our method with versus without goal switching. In particular, I demonstrate that switching goals leads to better performance if the assumption of increasing variance is violated and does not harm performance when that assumption is met. Firstly, I compare the resource-rationality score of the two algorithms in an environment where switching goals should lead to an improvement in performance. This environment has a total of 60 nodes split into four different goals, each consisting of 15 nodes in the low-level MDP. The difference to previously used environments is that one of the unavoidable intermediate nodes has a 10% probability to harbor a large loss of -1500 (see Figure 2.6). The cost of computation in this environment is 10 points per click ($\lambda = 10$).

The optimal strategy for this environment selects a goal, checks this high-risk node on the path leading to the selected goal, and switches to a different goal if it uncovers the large loss. I compare the performance of hierarchical-BMPS with goal-switching to the performance of hierarchical-BMPS without goal-switching, the non-hierarchical-BMPS method with tree contraction[†], and

[†]Without our tree contraction method, the original version of BMPS would not have been scalable

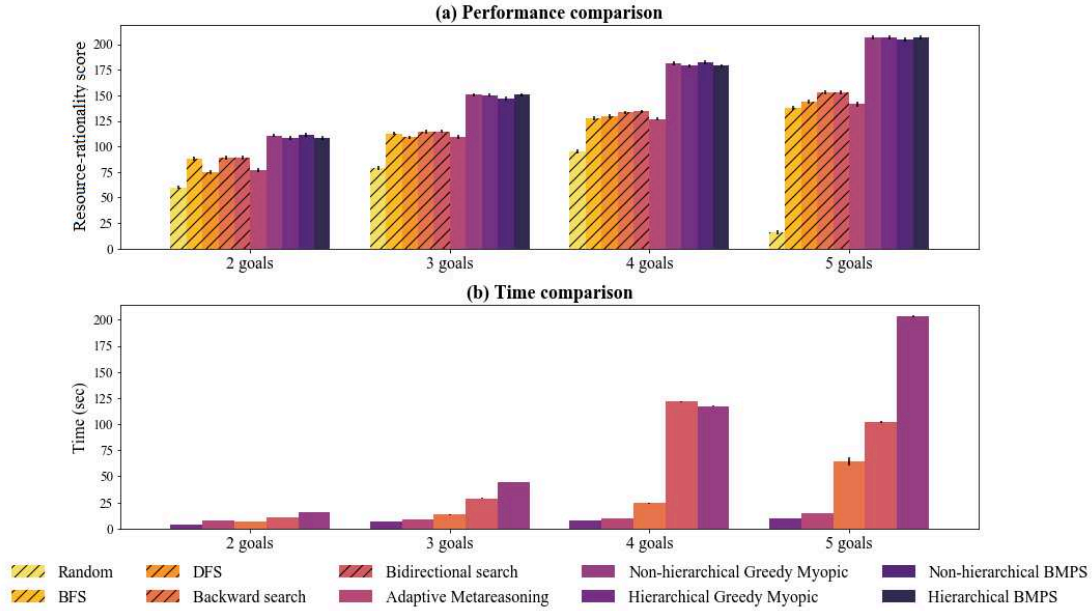


Figure 2.5: (a) Resource-rationality scores of existing planning algorithms (striped bars) versus planning strategies discovered by various strategy discovery methods (bars without stripes). (b) Comparison of the mean time for various strategy discovery methods (in seconds).

human performance. The three strategy discovery algorithms were all trained in the same environment following the same training steps. Their resource-rationality scores are noted in Table 2.3.

All resource-rationality scores do not follow a normal distribution as tested with a Shapiro-Wilk test ($p < .001$ for each). The performance between the individual algorithms was compared with a Wilcoxon rank-Sum rest, adjusting the critical alpha value via Bonferroni correction. Comparing the score of goal-switching to both our method without goal-switching ($W = 14.07, p < .001$) and the original BMPS algorithm (W=11.38, p<.001), shows a significant benefit of goal-switching. Comparing the performance of the original BMPS method to the no-goal-switching algorithm, the original BMPS version performs significantly better ($W = 18.7, p < .001$).

To show that enabling our algorithm's capacity for goal-switching has no negative effect on its

enough to handle this environment.

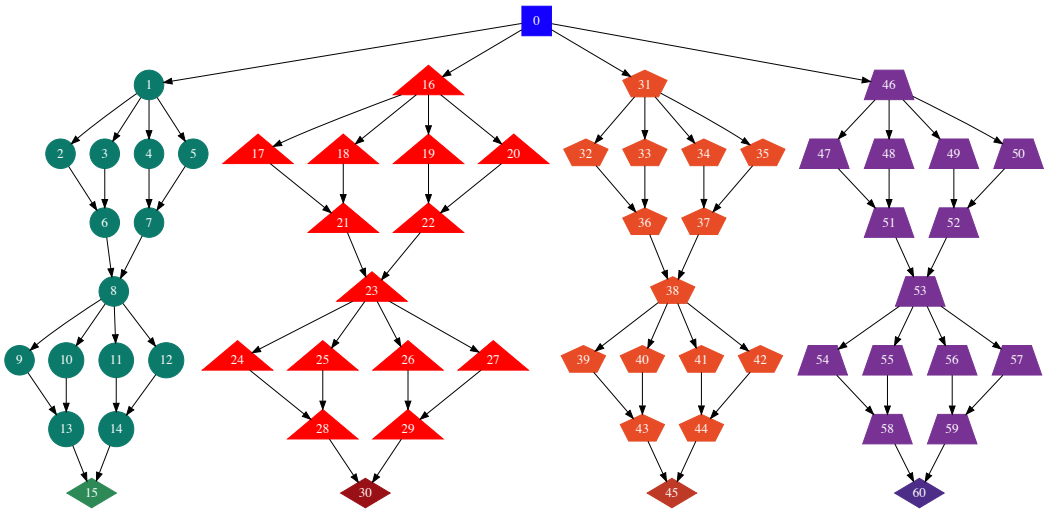


Figure 2.6: Environment that demonstrates the utility of goal switching. High-risk nodes (8, 23, 38 and 53) follow a categorical reward distribution of -1500 with a probability of 0.1 and 0 with a probability of 0.9. Goal nodes (15, 30, 45 and 60) have a categorical uniform distribution over the possible reward values 0, 25, 75, and 100. The first node in each subtree (1, 16, 31 and 46) as well as the root node (0) have a fixed reward of 0. All other intermediate nodes follow a categorical uniform distribution over the possible values of -10, -5, 5, and 10.

performance even when the assumption of the hierarchical decomposition is met, I perform a second comparison on the two-goal environment with increasing variance as in Section 2.5.2. Since in this environment the rewards are most variable at the goal nodes, switching goals should usually be unnecessary. Therefore, due to the environment structure, I do not expect the goal-switching strategy to perform better than the purely hierarchical strategy. By comparing the resource-rationality score in this environment, I observe that both versions of the algorithm perform similarly well (see Table 2.4). A Wilcoxon rank-Sum rest ($W = 0.03, p = .98$) shows no significant difference between the two. This demonstrates that the addition of goal switching to the algorithm does not impair the performance, even when goal switching is not beneficial.

Type	Algorithm	N	RR-score	Std
S	No goal-switching hierarchical-BMPS	5000	-80.38	446.47
S	Goal-switching Hierarchical BMPS	5000	51.33	32.2
S	Non-hierarchical-BMPS	5000	39.29	40.93
S	Hierarchical greedy myopic VOC	5000	-162.85	36.82
S	Non-hierarchical greedy myopic VOC	5000	25.48	41.12
S	Adaptive Metareasoning Policy Search	5000	-137.88	40.48
P	Depth-First Search	5000	-242.84	27.28
P	Breadth-First Search	5000	-253.59	41.55
P	Bidirectional Search	5000	-251.59	34.38
P	Backward Search	5000	-230.76	40.26
P	Random Policy	5000	-387.07	206.42
	Human baseline	26	-79.92	74.06

Table 2.3: Resource-rationality score (RR-score) and standard deviation (Std) of the automatically discovered hierarchical planning strategy with and without goal-switching, as well as the strategy discovered by the original BMPS algorithm averaged over 5000 random instances of the high-risk environment with four goal states. A human baseline is gathered in an online experiment with a lower number of samples (see Section 2.5.4).

2.5.4 How does the performance of the automatically discovered strategies compare to human planning?

To be able to compare the performance of the automatically discovered planning strategies to the performance of people, my collaborators and I conduct experiments on Amazon Mechanical Turk (Litman et al., 2017).

Methods My collaborators and I measured human resource-rationality scores in Flight Planning tasks that are analogous to the environments used to evaluate the method (see Figure 2.7).

For the environments that conform to the increasing variance structure (e.g. Figure 2.3), my collaborators and I recruited 78 participants for each of the four environments (average age 34.71 years, range: 19–70 years; 46 female). Participants were paid $2.00 plus a performance-dependent bonus

Algorithm	N	RR-score	Std
No goal-switching	5000	108.84	95.37
Goal-switching	5000	108.78	95.37

Table 2.4: Average resource-rationality scores (RR-score) of the planning strategies discovered with the meta-controller method and the hierarchical problem decomposition and their standard deviation (Std) based on their performance on 5000 random instances of the increasing variance environment with two goal states.

(average bonus $1.52). The average duration of the experiment was 25.1 min. For the risky environment (see Figure 2.6), I recruited 48 participants (average age 36.98 years, range: 19–70 years; 25 female). Participants were paid $1.75 plus a performance-dependent bonus (average bonus $0.34). The average duration of the experiment was 14.86 min. Following instructions that informed the participants about the range of possible reward values, participants were given the opportunity to familiarize themselves with the task in 5 practice trials of the Flight Planning task. After this, participants were evaluated on 15 test trials of the Flight Planning task for the first types of environments (increasing variance) and 5 test trials for the second types of environments (goal-switching). To ensure high data quality, my collaborators and I applied the same pre-determined exclusion criterion throughout all presented experiments. I excluded participants who did not make a single click on more than half of the test trials because not clicking is highly indicative of a participant not engaging and speeding through the task. In the first environment type, my collaborators and I excluded 3 participants and in the second environment type, I excluded 22 participants.

Results The results of this experiment are shown in the last row of Table 2.2 and Table 2.3. Surprisingly, my collaborators and I found that contrary to what has been observed in small sequential decision problems (Callaway et al., 2018b, 2020), human performance is far from resource-rational in large sequential decision problems.

Concretely, in the 15 increasing variance environments, human participants performed much worse than the strategy discovered by our hierarchical method regardless of the number of goals

($p < .02$ for all pairwise Wilcoxon rank-sum tests). Compared to the human baseline, the hierarchical strategies discovered by our method appear to achieve a higher level of computational efficiency on multiple instances of environments with an increasing-variance reward structure. The computational efficiency of a strategy is defined by its resource-rationality score, which combines the quality of the solution (the reward earned by moving through the environment) with the computational cost of planning.

In the high-risk environment, the average human resource-rationality score was only -79.92 (see Table 2.3) whereas our method achieved a resource-rationality score of 41 on the same 15 environment instances participants were evaluated on. A Shapiro-Wilk test detected no significant violation of the assumption that participants' average scores are normally distributed ($p. = 33$). Therefore, my collaborators and I compared the average human performance to our method in a one-sample t-test. I found that human participants performed significantly worse than the strategy discovered by our method ($t(25) = -13.06, p < .001$). This suggests that, at least within the setting of a limited number of training trials for human participants, the strategy discovered by hierarchical-BMPS performs on a superhuman level at achieving the trade-off between gathering information and the associated time cost, resulting in a higher overall resource-rationality score.

2.6 Improving human decision-making by teaching automatically discovered planning strategies

Having shown that hierarchical-BMPS discovers planning strategies that achieve a super-human resource-rationality score (i.e., the strategy discovered by our method performs fewer clicks than the strategies that people use intuitively), my collaborators and I now evaluate whether we can improve human decision-making by teaching them the automatically discovered strategies. Building on the Mouselab-MDP paradigm (Callaway et al., 2017), my collaborators and I investigate this question

in the context of the Flight Planning task illustrated in Figure 2.7. Participants are tasked to plan the route of an airplane across a network of airports. Each flight gains a profit or a loss. Participants can find out how much profit or loss an individual flight would generate by clicking on its destination for a fee of $1. The participant's goal is to maximize the sum of the flights' profits minus the cost of planning. Participants can make as few or as many clicks as they like before selecting a route using their keyboard.

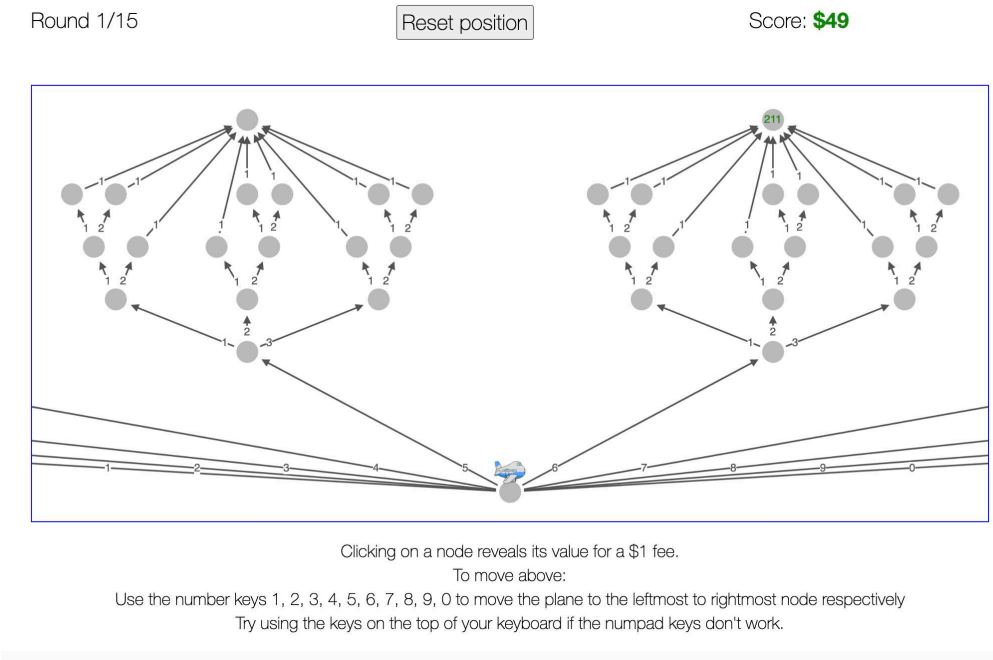


Figure 2.7: Screenshot of the Flight Planning task used to assess people's planning skills in Experiments 1. Participants can drag and zoom into the environment to show different portions.

To teach people the automatically discovered strategies, my collaborators and I developed cognitive tutors that show people step-by-step demonstrations of what the optimal strategies for different environments would do to reach a decision (see Figure 2.8). In each step, the strategy selects one click based on which information has already been revealed. At some point, the tutor stops clicking and moves the airplane down the best route indicated by the revealed information. Moving

forward, I will refer to cognitive tutors teaching the hierarchical planning strategies discovered by hierarchical-BMPS as *hierarchical tutors* and refer to the tutors teaching the strategies discovered by non-hierarchical-BMPS as *non-hierarchical tutors*.

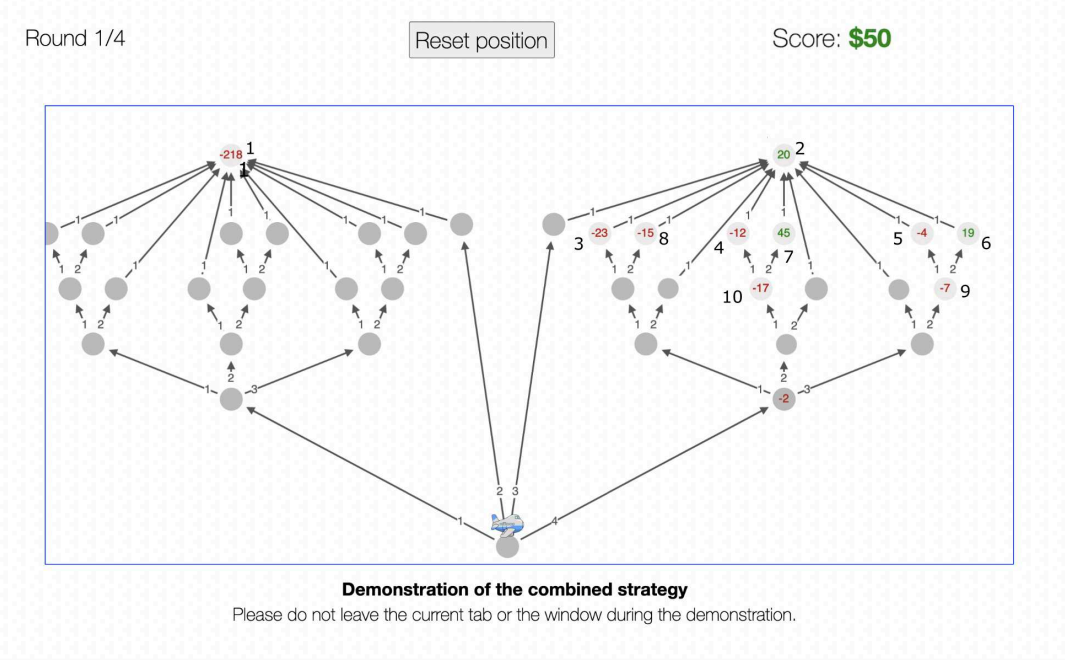


Figure 2.8: Screenshot of the cognitive tutor for demonstrating the non-hierarchical strategy evaluated in Experiment 1. The numbers beside the node indicate the sequence in which the clicks were performed.

To evaluate the effectiveness of these demonstration-based cognitive tutors, my collaborators and I conducted two training experiments in which participants were taught the optimal strategies for flight planning problems equivalent to the two types of environments in which my collaborators and I evaluated hierarchical-BMPS in Section 2.5. To assess the potential benefits of the hierarchical tutors enabled by hierarchical-BMPS, these experiments compare the performance of people who were taught by hierarchical tutors against the performance of people across three control conditions. The control conditions are: people who were taught by non-hierarchical tutors in a smaller environment with two goals, people who were taught by the original feedback-based tutor in a small

environment of similar structure (Lieder et al., 2020, 2019a), and people who practiced the task on their own in the full environment. My collaborators and I developed the best version of each tutor possible, given the limited scalability of the underlying strategy discovery method. The increased scalability of hierarchical-BMPS enabled the hierarchical tutor to demonstrate the optimal strategy for the task participants faced, whereas the other tutors could only show demonstrations on smaller versions of the task. My collaborators and I found that showing people a small number of demonstrations of the optimal planning strategy significantly improved their decision-making not only when the assumption underlying our method's hierarchical problem decomposition is met (Experiment 1) but also when it is violated (Experiment 2). In Section 2.6.3, my collaborators and I perform an additional experiment where we show that teaching our method through demonstrations leads to improved decision-making and closer similarity with the optimal strategy than teaching people with random demonstrations that reveal similar information about the environment's structure to the participants.

2.6.1 Training Experiment 1: Teaching people the optimal strategy for an environment with increasing variance

In Experiment 1, participants were taught the optimal planning strategy for an environment in which 10 final destinations can be reached through 9 different paths comprising between 4 and 6 steps each (see Figure 2.7). The most important property of this environment is that the variance of available rewards doubles from each step to the next, starting from 5 in the first step. Therefore, in this environment, the optimal planning strategy is to first inspect the values of alternative goals, then commit to the best goal one could find, and then plan how to achieve it without ever reconsidering other goals.

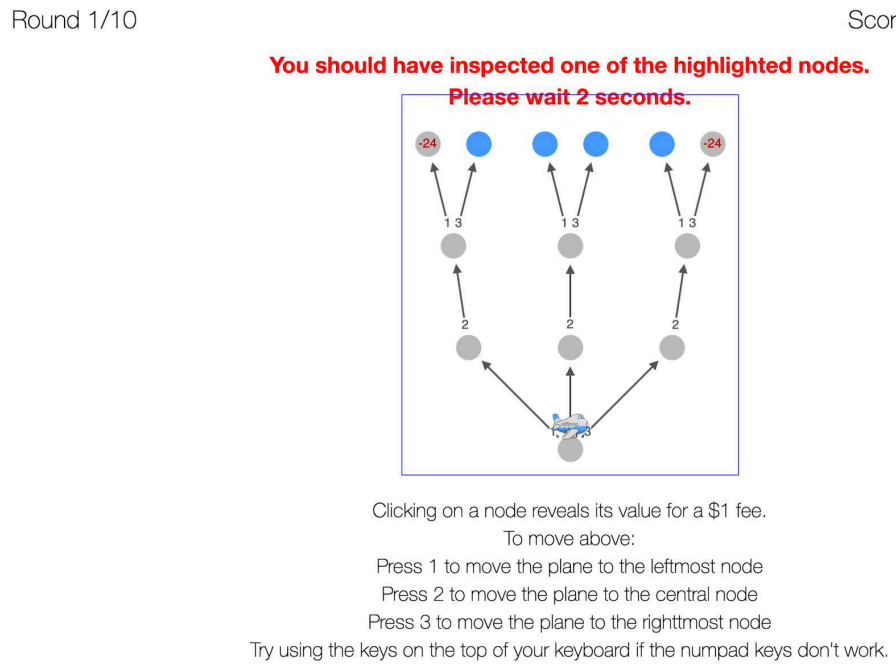


Figure 2.9: Screenshot of the feedback-based tutor against which my collaborators and I evaluated our scalable demonstration-based tutor. This tutor gives people feedback on the planning operations they perform in a smaller version of the environment. This small environment is the largest one for which optimal feedback can be computed with the currently available methods.

METHODS My collaborators and I recruited 168 participants on Amazon Mechanical Turk (average age 34.9 years, range: 19–70 years; 98 female) (Litman et al., 2017). Participants were paid $2.50 plus a performance-dependent bonus (average bonus $2.86). The average duration of the experiment was 46.9 min. Participants could earn a performance-dependent bonus of 1 cent for every 10 points they won in the test trials.

All participants had to first agree to a consent form stating they were above 18, a US citizen residing in the USA, and fluent in English. After this, instructions about the range of possible rewards (−250 to 250), cost of clicking ($1), and the movement keys were presented. Then, participants went through 5 trials to familiarize themselves with the experiment. Following this, participants were either given additional 10 practice trials or had 10 trials with the cognitive tutor, depending on

their experimental condition. Finally, the participant was given 15 test trials in the flight planning task with 10 possible final destinations, illustrated in Figure 2.7. Participants started with 50 points at the beginning of the test block.

To evaluate the efficacy of cognitive tutors, participants were assigned to 4 groups. In the experimental group, participants were taught by the hierarchical tutor. The first control group was taught by the non-hierarchical tutor. The second control group was taught by the feedback-based cognitive tutor (Lieder et al., 2019a, 2020) illustrated in Figure 2.9. The third control group practiced the Flight Planning task 10 times without feedback. The hierarchical tutor taught the strategy discovered by the hierarchical-BMPS algorithm discovered for the task participants had to perform in the test block. It first demonstrated 3 trials with the goal selection strategy; it then showed three demonstrations of the goal-planning strategy, and finally presented 4 demonstrations of the complete strategy combining both parts. The non-hierarchical tutor showed 10 demonstrations of the strategy that non-hierarchical-BMPS discovered for the largest version of the Flight Planning task it could handle (i.e., 2 goals instead of 10 goals). Computational bottlenecks confined the feedback-based tutor to a three-step planning task with six possible final destinations, shown in Figure 2.9 (Lieder et al., 2020, 2019a). Participants received feedback on each of their clicks and their decision when to stop clicking, as illustrated in Figure 2.9. When the participant chose a suboptimal planning operation, they were shown a message stating which planning operation the optimal strategy would have performed instead. In addition, they received a timeout penalty whose duration was proportional to how suboptimal their planning operation had been.

Counterbalanced assignment ensured that participants were equally distributed across four experimental conditions (i.e., 42 participants per condition). To ensure high data quality, my collaborators and I applied a pre-determined exclusion criterion. We excluded 7 participants who did not make a single click on more than half of the test trials because not clicking is highly indicative of speeding through the experiment without engaging with the task.

On each trial, the participant's expected resource-rationality score was calculated as the expected reward of the path they chose minus the cost of the clicks they had made. As in the simulations, the cost per click in this environment was 1. I analyzed the data using the robust `f1.ld.f1` function of the `nparLD` R package (Noguchi et al., 2012). The function performs a non-parametric ANOVA-type statistic with Box approximation (Box, 1954) to determine the main effect and additional ANOVA-type statistics (ATS) with the denominator degrees of freedom set to infinity for post hoc comparisons (Noguchi et al., 2012).

Results Table 2.5 shows the average resource-rationality scores of the four groups on the test trials. According to a Shapiro-Wilk test, participants' scores on the test trials were not normally distributed in any of the four groups (all $p < .001$). Therefore, I tested our hypothesis using a non-parametric ANOVA-type statistic with Box approximation (Box, 1954) to test if there are any significant differences between the groups in the repeated-measures design. I found that people's performance differed significantly across the four experimental conditions ($F(2.85, 144.17) = 3.92$, $p = .0113$). Pairwise post hoc ATS comparisons confirmed that teaching people strategies discovered by the hierarchical method significantly improved their performance (204.48 points/trial) compared to the no-feedback control condition (177.36 points/trial, $F(1) = 7.57, p = .006$), the feedback-based cognitive tutor (167.02 points/trial, $F(1) = 11.89, p < .001$), and the non-hierarchical demonstration (181.58 points/trial, $F(1) = 5.02, p = .025$). By contrast, neither the feedback-based cognitive tutor ($F(1) = 0.57, p = .45$) nor the non-hierarchical demonstration ($F(1) = 0.19, p = .67$) were more effective than letting people practice the task on their own.

I investigated which percentage of participants followed the general strategy of planning which goal to pursue and then planning a path to the chosen goal. I measured this by checking if the participant first clicked any number of goal nodes and then clicked one of the nodes leading to the best discovered goal. A participant is considered to have learned this aspect of the strategy, if

the described behavior is shown in over half of the five test trials. The results in Table 2.6 show that the participants from the non-hierarchical demonstration (95%) and hierarchical demonstration (97.56%) conditions learned goal planning to a high degree. Participants of the no-feedback (78.05%) and feedback-based tutor (76.92%) conditions still performed well in mastering this general aspect of the optimal strategy. I compared the proportion of participants who learned this behavior and received the hierarchical demonstration to other conditions using a z-test and corrected the p-values for multiple comparisons using the Benjamini-Hochberg method (Benjamini and Hochberg, 1995). This showed a significant difference between participants of the hierarchical demonstration condition versus the no-feedback condition ($z = 2.7, p = .01$) and the feedback-based tutor condition ($z = 2.79, p = .01$). I observed no significant difference between participants of the hierarchical demonstration condition and non-hierarchical demonstration condition ($z = 0.61, p = .542$).

To test how well participants learn to follow the optimal strategy, I further calculated to what extent each participant's planning operations match the near-optimal strategy discovered by our method. These additional agreement measures were calculated per planning operation the participant performed. For each planning operation, I reconstructed the current belief state and then used hierarchical-BMPS to calculate the set of optimal actions and compare them to the action taken by the participant. I found that participants in the hierarchical demonstration condition match the automatically discovered strategy better (across all test trials performed by participants of this condition, participants chose one of the optimal planning operations in 35.84% of their actions with a standard deviation of $\pm$15%) than participants in the no-feedback control condition ($22.08\% \pm 12\%$), the feedback condition ($27.44\% \pm 13\%$), and the non-hierarchical demonstrations condition ($28.51\% \pm 13\%$). The ANOVA-type statistic with Box approximation (Box, 1954) showed a significant effect of the condition on click agreement ($F(3.00, 156.70) = 21.73$, $p < .001$). Post hoc ATS comparisons showed significant differences between hierarchical demon-

strations and the control conditions: the no-feedback condition ($F(1) = 64.00, p < .001$), the feedback-based cognitive tutor ($F(1) = 20.90, p < .001$), and the non-hierarchical demonstration ($F(1) = 16.59, p < .001$). Additional significant effects are present between the no-feedback condition and the feedback-based cognitive tutor ($F(1) = 11.44, p < .001$) as well as the non-hierarchical demonstration ($F(1) = 16.36, p < .001$). No significant difference was found between the participants who were taught by the feedback-based cognitive and the non-hierarchical demonstration ($F(1) = 0.35, p = .56$), respectively.

I further analyzed which aspects of the optimal strategy participants learned. Results of the extended analysis consisting of 6 additional measures can be found in Table 2.6. In this analysis, I split the click agreement measure into three sub-categories: goal agreement, path agreement, and termination agreement. Goal agreement measures how well participants' clicks match the optimal strategy when planning which goal to pursue, and path agreement measures how well participants plan how to achieve the selected goal. The higher their agreement with the optimal strategy was, the better they performed. I further split the termination measure into the overall termination agreement, the goal termination agreement, and the path termination agreement. The overall termination agreement measured how well participants stop planning when it is no longer beneficial. Goal and path termination agreement measured how well participants switch between planning which goal to pursue and how to achieve the selected goal. When calculating path agreement and path termination agreement, I allowed for mistakes in goal selection (i.e. planning an optimal path to a suboptimally chosen goal will still result in a high path agreement score). The termination agreement measures were calculated using the balanced accuracy measure to equally account for both not terminating too early and not terminating too late. Participants of all conditions were able to match the optimal strategy for selecting goals roughly equally well, with agreement scores ranging from 32% to 38%. Participants trained by the feedback-based tutor performed slightly better at selecting goals matching the optimal strategy with 38.08% than participants of the hierarchical demonstra-

tion condition (34.86%). These differences were not statistically detectable ($F(2.94, 151.95) = 2.6$, $p = .056$). The agreement measure for planning how to achieve the selected goal showed significant differences ($F(2.95, 152.54) = 28.45, p < .001$). Participants trained by the hierarchical demonstration achieved a higher agreement score (44.45%) than participants of other conditions: no-feedback (17.82%; $F(1) = 74.57, p < .001$), feedback-based tutor (23.00%; $F(1) = 42.39$, $p < .001$), and non-hierarchical demonstration (30.37%; $F(1) = 15.88, p < .001$).

Overall, path and goal agreement were relatively low across conditions, with under 50% of participants' clicks matching the resource-rational strategy demonstrated by our cognitive tutor. My collaborators and I identified the complicated reward structure of the goal nodes as a potential cause for the low overall agreement. The increase in the variance of goal rewards from 100 to 180 (see Section 2.5.1) resulted in an optimal strategy that first investigates goals 5 and 10, while investigating other goal nodes first was slightly suboptimal. This detail of the optimal strategy was not well understood by participants: while participants learned to click a goal node as the first planning action in the vast majority of trials (93.66% for the no-feedback condition, $>$ 99% for all other conditions), participants learned to click either goal 5 or goal 10 as the first action in only slightly above half of the trials (50.73% for participants of the no-feedback condition, 56.58% for participants of the feedback-based tutor condition, 57.67% for participants of the non-hierarchical demonstration condition, and 62.11% for participants of the hierarchical demonstration condition).

Lastly, I compared the termination agreement for overall termination, goal termination, and path planning termination. Across all termination agreement measures and conditions, participants learned when to terminate planning quite well ($>$ 60% agreement). Participants trained by the hierarchical demonstration matched the optimal strategy slightly more closely than participants of the other conditions for all three measures. For termination agreement, conditions differed significantly ($F(2.93, 151.19) = 3.29, p = .023$): the post hoc ATS showed that participants of the hierarchical demonstration condition achieved a significantly higher agreement than participants of

the non-hierarchical demonstration condition ($F(1) = 11.22, p < .001$), but not the no-feedback condition or the feedback-based tutor condition ($p > .1$ for both). I did not detect a significant difference between the conditions for either goal termination agreement ($F(2.90, 148.66) = 2.14$, $p = .099$) or path termination agreement ($F(2.84, 144.14) = 2.62, p = .057$). Overall, these results showed that the hierarchical method is better able to discover and teach resource-rational strategies for environments in which previous methods failed.

		RR-score		Click Agreement	
Algorithm	N	Mean	Std	Mean	Std
No-feedback	41	177.36	95.04	0.2208	0.12
Feedback-based cognitive tutor	39	167.02	103.42	0.2744	0.13
Non-hierarchical demonstration	40	181.58	93.63	0.2851	0.13
Hierarchical demonstration	41	**204.48**	83.16	**0.3584**	0.15

Table 2.5: Experimental results for Training Experiment 1. For each condition, my collaborators and I report the number of participants, the mean and standard deviation of the resource-rationality score (RR-score), and the mean and standard deviation of the click agreement measure.

	Goal	Goal Agr.		Path Agr.		Term. Agr.		Goal Term. Agr.		Path Term. Agr.	
Condition	Plan	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std
No-feedback	0.7805	0.3253	0.21	0.1782	0.17	0.7541	0.21	0.7656	0.27	0.6862	0.23
Feedback-based tutor	0.7693	0.3808	0.19	0.2300	0.22	0.7646	0.18	0.7545	0.25	0.6820	0.23
Non-hier. demonstration	0.9500	0.3702	0.23	0.3037	0.20	0.7213	0.18	0.7268	0.28	0.6569	0.20
Hier. demonstration	0.9756	0.3486	0.18	0.4445	0.26	0.7822	0.18	0.8286	0.20	0.7161	0.23

Table 2.6: Extended strategy analysis for Training Experiment 1. For each condition, my collaborators and I report the proportion of participants following a goal planning strategy and the mean and standard deviation of the goal, path, and termination agreement measures.

2.6.2 Training Experiment 2: Teaching people the optimal strategy for a risky environment

In Experiment 2, participants were taught the strategy hierarchical-BMPS discovered for the 8-step decision problem illustrated in Figure 2.6. Critically, in this environment, each path contains one risky node that harbors an extreme loss with a probability of 10%. Therefore, the optimal strategy for this environment inspects the risky node while planning how to achieve the selected goal and then switches to another goal when it encounters a large negative reward on the path to the initially selected goal.

Method To test whether our approach can also improve people's performance in environments with this more complex structure, I created two demonstration-based cognitive tutors that teach the strategies discovered by hierarchical-BMPS with goal-switching and hierarchical-BMPS without goal-switching, respectively, and a feedback-based tutor that teaches the optimal strategy for a 3-step version of the risky environment.

This experiment used a Flight Planning Task that is analogous to the environment described in Section 2.5.3 (see Figure 2.6). Specifically, the environment comprises 4 goal nodes and 60 intermediate nodes (i.e., 15 per goal). Although each goal can be reached through multiple paths, all of those paths lead through an unavoidable node that has a 10% risk of containing a large loss of -1500. The aim of this experiment is to verify that I am still able to improve human planning even when the environment requires a more complex strategy that occasionally switches goals during planning. To test this hypothesis, I showed the participants demonstrations of the strategy discovered by hierarchical-BMPS in the experimental condition, and compared their performance to the resource-rationality scores of three control groups. The first control group was shown demonstrations of the strategy discovered by the version of hierarchical-BMPS without goal switching; the second control

group discovered their own strategy in five training trials; the third control group practiced planning on a three-step task with a feedback-based tutor (Lieder et al., 2020, 2019a). The environment used by the feedback-based tutor mimicked the high-risk environment. To achieve this I changed the reward distribution of the intermediate nodes so that there is a 10% chance of a negative reward of -96, a 30% chance of -4, a 30% chance of +4, and a 30% chance of +8. I then recomputed the optimal feedback using dynamic programming (Lieder et al., 2020, 2019a).

I recruited 201 participants (average age 34.01 years, range:19–70 years; 101 female) on Amazon's Mechanical Turk (Litman et al., 2017) over three consecutive days. All but two of them completed the assignment. Applying the same pre-determined exclusion criterion as used in Experiment 1 (i.e., excluding participants who do not engage with the environment in more than half of the test trials) led to the exclusion of 30 participants (15%), leaving 169 participants. Participants were paid $1.30 and a performance-dependent bonus of up to $1. The average bonus was $0.56 and the average time of the experiment was 16.28 minutes. Participants were randomly assigned to one of four experimental conditions determining their training in the planning task. All groups were tested in five identical test trials. The data was analyzed using the robust `f1.ld.f1` function of the `nparLD` R package (Noguchi et al., 2012). As in Training Experiment 1, the participant's resource-rationality score for each trial was calculated as the expected reward of the path they chose minus the cost of the clicks they had made. As in the simulations, the cost per click was 10.

Results The results of the experiment are summarized in Table 2.7. Since the Shapiro-Wilk test showed that none of the four conditions are normally distributed ($p < .001$ for all), I again used the non-parametric ANOVA-type statistic with Box approximation (Box, 1954) to evaluate the data. The test showed significant differences between the four groups ($F(2.79, 150.28) = 20.19$, $p < .001$). Pairwise robust ATS post hoc comparisons showed that participants trained with demonstrations of the strategy discovered by hierarchical-BMPS with goal-switching significantly

outperform all other conditions: participants trained by purely hierarchical demonstrations without goal switching ($F(1) = 58.72, p < .001$), participants who did not receive demonstrations ($F(1) = 31.13, p < .001$), and participants who had practiced planning with optimal feedback on a smaller analogous environment ($F(1) = 32.07, p < .001$). The performance of the three control groups was statistically indistinguishable (all $p \geq 0.18$). Participants trained by purely hierarchical demonstrations (i.e. without goal switching) did not perform significantly better than participants that trained with optimal feedback ($F(1) = 1.82, p = .18$) or participants that did not receive demonstrations ($F(1) = 0.34, p = .56$). Additionally, I detected no significant difference between the optimal feedback condition and the no-feedback condition ($F(1) = 0.28, p = .6$).

The participants from the goal-switching demonstration condition learned goal planning to a high degree (i.e., 60%). I again used a z-test and applied the Benjamini-Hochberg method (Benjamini and Hochberg, 1995) to compare the goal planning performance between conditions. This showed that participants of the goal-switching demonstration condition learned goal planning better than the other conditions: no-feedback (35.71%; $z = 2.265, p = .023$), feedback-based tutor (20.93%; $z = 3.725, p < .001$), and hierarchical demonstration (25.64%; $z = 3.164, p = .002$).

I repeated the statistical analysis for the click agreement measure, finding a significant effect of the experimental condition ($F(2.58, 132.19) = 47.45, p < .001$). Comparing participants' strategies to the strategy discovered by our method using the post hoc ATS showed that the planning operations of participants trained with our goal-switching strategy matched those of the optimal strategy more often ($59.94\% \pm 29\%$) than those of participants in the no-feedback condition ($24.33\% \pm 24\%$; $F(1) = 56.62, p < .001$), the feedback tutor condition ($17.22\% \pm 14\%$; $F(1) = 164.80, p < .001$), and the hierarchical strategy without goal-switching ($42.98\% \pm 23\%$; $F(1) = 10.82, p = .001$). I found additional significant differences between the no-goal switching hierarchical demonstration condition and the other two control conditions: optimal feedback ($F(1) = 82.61, p < .001$) and no-feedback ($F(1) = 22.80, p < .001$). Comparing the optimal feedback and no-feedback condition

revealed no significant difference ($F(1) = 3.01, p = .082$).

I again analyzed which aspects of the optimal strategy participants learned with the additional click agreement measures reported in Section 2.6.1. The results for the additional measures can be found in Table 2.8. I found significant differences in how well participants planned which goal to pursue ($F(2.41, 120.22) = 32.33, p < .001$). Participants in the goal-switching demonstration condition (78.21%) and the hierarchical demonstration condition (77.73%) performed best at selecting goals according to the optimal strategy, with no significant difference being detectable ($F(1) = 0.05, p = .83$). Participants in the goal-switching demonstration condition had a significantly higher goal agreement than participants in the no-feedback condition (57.76%; $F(1) = 11.36, p < .001$), and participants in the feedback tutor condition (36.01%; $F(1) = 110.5$, $p < .001$). The path planning agreement also showed significant differences for the different conditions ($F(2.45, 117.43) = 19.12, p < .001$). Participants trained by the goal-switching demonstration achieved an agreement score of 72.71%, whereas the other conditions achieved a significantly lower agreement: no-feedback (24.47%; $F(1) = 42.53, p < .001$), hierarchical demonstration (21.50%; $F(1) = 36.96, p < .001$), and feedback tutor (30.12%; $F(1) = 37.89, p < .001$).

Similarly, the participants trained by the goal-switching demonstration achieved higher termination agreement scores, matching the optimal strategy's termination rule more closely than participants of the no-feedback, feedback-based tutor, and hierarchical demonstration conditions across all three termination agreement measures. For all three termination measures, the differences between conditions are statistically significant: termination agreement ($F(2.96, 161.39) = 10.69$, $p < .001$), goal termination agreement ($F(2.77, 145.41) = 17.62, p < .001$), and path termination agreement ($F(2.86, 152.32) = 10.54, p < .001$). When deciding when to stop planning (termination agreement), participants who received goal-switching demonstrations matched the optimal strategy to a higher percentage (76.35%) than participants of the no-feedback condition (69.19%; $F(1) = 5.65, p = .017$), the hierarchical demonstration condition (68.35%; $F(1) = 7.41$,

$p = .006$), and the feedback tutor condition (63.05%; $F(1) = 34.31, p < .001$). When deciding when to stop planning which goal to pursue (goal termination agreement), participants of the goal-switching demonstration condition (84%) performed better than participants who received no-feedback (71.75%; $F(1) = 8.62, p = .003$) and participants who were in the feedback-tutor condition (57.31%; $F(1) = 61.80, p < .001$), but I did not detect a significant difference to participants who received hierarchical demonstrations (78.19%; $F(1) = 2.65, p = .103$). Lastly, participants who received the goal-switching demonstration (79.66%) also performed significantly better than all other conditions when deciding when to stop planning how to achieve the pursued goal (path term agreement): no-feedback (61.85%; $F(1) = 12.26, p < .0.001$), hierarchical demonstration (54.91%; $F(1) = 21.35, p < .001$), and feedback-tutor (58.21%; $F(1) = 26.83, p < .001$).

The results of this experiment showed that strategy discovery can be used to significantly improve human decision-making by showing people demonstrations of the automatically discovered hierarchical planning strategy with goal-switching. This is a unique advantage of hierarchical-BMPS because none of the other approaches were able to improve people's decision-making in this large and risky environment. By comparing human performance to the optimal performance of our algorithm in the same environment (see Table 2.3), it is clear that even though participants' performance significantly increased from training with the hierarchical tutor, they still did not fully understand the strategy based on the demonstrations alone. This reveals the limitations of teaching planning strategies purely with demonstrations, especially for more complex strategies. The different click agreement measures further showed that while participants who received demonstrations of our method did not entirely master the optimal strategy, they were able to find the optimal action in a significantly higher number of trials. This motivates the feedback-based cognitive tutors presented in Chapters 3 and 4.

Algorithm	N	RR-score Mean	RR-score Std	Click Agreement Mean	Click Agreement Std
No-feedback	42	-103.31	173.69	0.2433	0.24
Goal-switching demonstration	45	**-26.71**	121.87	**0.5994**	0.29
Hierarchical demonstration	39	-94.41	51.75	0.4298	0.23
Feedback tutor	43	-108.49	179.64	0.1722	0.14

Table 2.7: Experimental results for Training Experiment 2 using the high-risk environment shown in Figure 2.6. For each condition, I report the number of participants, the mean and standard deviation of the resource-rationality score (RR-score), and the mean and standard deviation of the click agreement measure.

Condition	Goal Plan	Goal Agr. Mean	Goal Agr. Std	Path Agr. Mean	Path Agr. Std	Term. Agr. Mean	Term. Agr. Std	Goal Term. Agr. Mean	Goal Term. Agr. Std	Path Term. Agr. Mean	Path Term. Agr. Std
No-feedback	0.3571	0.5776	0.38	0.2447	0.35	0.6919	0.19	0.7175	0.30	0.6185	0.30
Goal-switching demonstr.	0.6000	0.7821	0.30	0.7271	0.36	0.7635	0.21	0.8400	0.23	0.7966	0.30
Hierarchical demonstr.	0.2564	0.7773	0.29	0.2150	0.35	0.6835	0.19	0.7819	0.27	0.5491	0.33
Feedback tutor	0.2093	0.3601	0.34	0.3012	0.29	0.6305	0.14	0.5731	0.30	0.5821	0.27

Table 2.8: Extended strategy analysis for Training Experiment 2. For each condition, I report the proportion of participants following a goal planning strategy and the mean and standard deviation of the goal, path, and termination agreement measures.

2.6.3 Control experiment

To investigate a potential confound, my collaborators and I ran an additional experiment using the 8-step decision problem shown in Figure 2.6 (i.e., the environment used in Experiment 2). The goal of this experiment was to investigate whether participants truly benefit from demonstrations of the near-optimal strategy, or if they merely learn about the environment's structure and then use this knowledge to inform their own strategy.

Methods The hypothesis my collaborators and I want to verify in this experiment is that participants trained by our tutor would learn to follow the optimal strategy more closely than participants who learn the same information about the environment structure (e.g. potential rewards of

different nodes) but do not see any demonstrations of the strategy discovered by our method. To test this hypothesis, my collaborators and I compared the performance of participants who were shown demonstrations of our method (analogous to the *goal-switching demonstration* condition in Experiment 2) to a new control condition. Participants in the control condition were shown random demonstrations where the demonstrator clicks random nodes and then moves along a random path through the environment. The actions of the random demonstrations were matched to the types of nodes clicked in the demonstration of our method (e.g., if the demonstration of our method revealed two goal nodes and two risky nodes, the random demonstration in that environment would reveal two random goal nodes and two random risky nodes as well). Additionally, the random demonstrations also contained 4-6 randomly selected additional clicks that were not included in our method's demonstration, arguably teaching participants more information about the structure of the environment than the demonstration of our method.

My collaborators and I recruited 100 participants (average age 25.06 years, range: $18 - 48$, 53 female) on Prolific (http://www.prolific.co)[2021]. Applying the exclusion criterion used in Experiment 1 (participants who didn't engage with the task) led to the exclusion of 20 participants (20%). Two additional participants were removed because they indicated they did not try their best to achieve a high score. Participants were paid £2.00 plus a performance dependent bonus of up to £1.00. The average bonus was £0.50 and the average completion time 18.9 minutes. Participants were randomly assigned one of the two conditions. The data was analyzed analogous to Experiment 1 and Experiment 2 using the `nparLD` R package (Noguchi et al., 2012).

Results The results for this experiment can be found in Table 2.9 and Table 2.10. I used the non-parametric ANOVA-type statistic with Box approximation (Box, 1954) to analyze the data. Our analysis showed a significant difference between the resource-rationality scores of the two conditions ($F(1, 74.86) = 5.53, p = .021$). Participants of the goal-switching demonstration condi-

Algorithm	N	RR-score Mean	RR-score Std	Click Agreement Mean	Click Agreement Std
Random demonstration	36	-79.78	148.49	0.2528	0.23
Goal-switching demonstration	42	**-69.14**	211.60	**0.4871**	0.28

Table 2.9: Experimental results for the control experiment using the high-risk environment, shown in Figure 2.6. For each condition, my collaborators and I report the number of participants, the mean and standard deviation of the resource-rationality score (RR-score), and the mean and standard deviation of the click agreement measure.

Condition	Goal Plan	Goal Agr. Mean	Goal Agr. Std	Path Agr. Mean	Path Agr. Std	Term. Agr. Mean	Term. Agr. Std	Goal Term. Agr. Mean	Goal Term. Agr. Std	Path Term. Agr. Mean	Path Term. Agr. Std
Random demonstration	0.2222	0.5355	0.39	0.3994	0.38	0.6603	0.23	0.6537	0.32	0.6700	0.31
Goal-switching demonstr.	0.5952	0.7161	0.32	0.5778	0.40	0.7297	0.19	0.7192	0.31	0.6751	0.36

Table 2.10: Extended strategy analysis for the control experiment. For each condition, my collaborators and I report the proportion of participants following a goal planning strategy and the mean and standard deviation of the goal, path, and termination agreement measures.

tion achieved a higher resource-rationality score than participants who observed random demonstrations. With 59.52%, a significantly higher percentage of participants of the goal-switching demonstration condition learned to follow the general goal planning strategy of first selecting a goal and then planning how to achieve the goal than participants of the control condition (22.22%, $z = 3.324, p < .001$). I again compared how well the participants matched the optimal strategy (click agreement) and observed a higher match for the goal-switching demonstration condition ($48.71\% \pm 28\%$) than the random demonstration condition ($25.28\% \pm 23\%$). Analogous to the analysis of the previous Experiments, I used the ANOVA-type statistic to confirm the statistical significance of this difference ($F(1, 74.92) = 29.43, p < .001$). Comparing the detailed click agreement measures reported in Table 2.10, participants who received the goal-switching demonstration of the optimal strategy achieved a significantly higher match with the optimal strategy across the following measures: goal agreement ($F(1, 64.17) = 8.09, p = .006$), path agreement ($F(1, 75.59) = 5.38, p = .023$), and termination agreement ($F(1, 72.97) = 4.73, p = .033$). I did

not detect a significant difference between the two conditions for the goal termination agreement ($F(1, 71.91) = 1.77, p = .188$), and path termination agreement ($F(1, 74.26) = 0.39, p = .53$).

The results of this experiment showed that teaching participants the discovered strategy leads to significantly better resource-rationality scores than purely teaching them about the reward structure of the environment. While participants of the random demonstration condition received even more information about the environment's rewards, they were unable to achieve the same level of resource-rationality. The click agreement measures further showed that the participants of the goal-switching demonstration condition learned the optimal strategy to a higher extent. I observed that participants in the goal-switching demonstration condition of this experiment performed worse than participants of the same condition in Training Experiment 2 (see Table 2.7). Apart from some randomness due to sampling from a different online crowdsourcing platform, I believe this has mostly been caused by a small number of outliers. Comparing the median expected scores shows a much smaller discrepancy between the experiments: -57.5 (interquartile range: 145) for this experiment and -20 (interquartile range: 140) for Training Experiment 2. Importantly, in both Experiments, I observed a statistically significant improvement in expected resource-rationality score and click agreement.

2.7 Conclusion

To make good decisions in complex situations, people and machines have to use efficient planning strategies because planning is costly. Efficient planning strategies can be discovered automatically, but computational challenges confined previous strategy discovery methods to tiny problems. To overcome this problem, I devised a more scalable machine learning approach to automatic strategy discovery. The central idea of the method is to decompose the strategy discovery problem into discovering goal-setting strategies and discovering goal achievement strategies. In addition, I made a

substantial algorithmic improvement to the state-of-the-art method for automatic strategy discovery (Callaway et al., 2018a) by introducing the tree-contraction method. My collaborators and I found that this hierarchical decomposition of the planning problem, together with the tree contraction method, drastically reduces the time complexity of automatic strategy discovery without compromising on the quality of the discovered strategies in many cases. Furthermore, by introducing the tree contraction method, I have extended the set of environment structures that automatic strategy discovery can be applied to from trees to directed acyclic graphs. These advances significantly extend the range of strategy discovery problems that can be solved by making the algorithm faster, more scalable, and applicable to environments with more complex structures. This is an important step towards discovering efficient planning strategies for real-world problems.

Recent findings suggest that teaching people automatically discovered efficient planning strategies is a promising way to improve their decisions (Callaway et al., 2022a). Due to computational limitations, this approach was previously confined to sequential decision problems with at most three steps (Callaway et al., 2022a). The strategy discovery methods developed in this chapter make it possible to scale up this approach to larger and more realistic planning tasks. As a proof-of-concept, my collaborators and I showed that our method makes it possible to improve people's decisions in planning tasks with up to 7 steps and up to 10 final goals. My collaborators and I evaluate the effectiveness of showing people demonstrations of the strategies discovered by our method in two separate experiments, where the environments were so large that previous methods were unable to discover planning strategies within a time budget of eight hours. Thus, the best one could do at training people with previous methods was to construct cognitive tutors that taught people the optimal strategy for a smaller environment with a similar optimal strategy or having people practice without feedback. Evaluating hierarchical-BMPS against these alternative approaches, my collaborators and I found that our approach was the only one that was significantly more beneficial than having people practice the task on their own. To the best of my knowledge, this makes hierarchical-

BMPS the only strategy discovery method that can improve human performance on sequential decision problems of this size. This suggests that hierarchical-BMPS based tutors make it possible to leverage reinforcement learning to improve human decision-making in problems that were out of reach for previous cognitive tutors.

The method's hierarchical decomposition of the planning problem exploits that people can typically identify potential mid- or long-term goals that might be much more valuable than any of the rewards they could attain in the short run. This corresponds to the assumption that the rewards available in more distant states are more variable than the rewards available in more proximal states. When this assumption is satisfied, hierarchical-BMPS discovers planning strategies much more rapidly than previous methods, and the discovered strategies are as good as or better than those discovered with the best previous methods. When this assumption is violated, the goal switching mechanism can compensate for that mismatch. This allows the discovered strategies to perform almost as well as the strategies discovered by BMPS. Hierarchical-BMPS switches goals more often the more strongly its increasing variance assumption is violated. In doing so, it automatically trades off its computational speed-up against the quality of the resulting strategy. This shows that the method is robust to violations of its assumptions about the structure of the environment; it exploits simplifying structure only when it exists.

Some aspects of hierarchical-BMPS share similarities with recent work on goal-conditioned planning (Nasiriany et al., 2019, Pertsch et al., 2020), although the problem I solved is conceptually different. For comparison, both aforementioned methods optimize the route to a given final location, whereas hierarchical-BMPS learns a strategy for solving sequential decision problems where the strategy chooses the final state itself. Furthermore, while Nasiriany et al. (2019) specified a fixed strategy for selecting the sequence of goals, hierarchical-BMPS learns such a strategy itself. Critically, while policies learned by Nasiriany et al. (2019) select physical actions (e.g., move left), the meta-level policies learned by hierarchical-BMPS select planning operations (i.e., simulate the out-

come of taking action a in state s and update the plan accordingly). Finally, hierarchical-BMPS explicitly considers the cost of planning to find algorithms that achieve the optimal trade-off between the cost of planning and the quality of the resulting decisions.

The method's scalability has its price. Since my approach decomposes the full sequential decision problem into two sub-problems (goal-selection and goal-planning), its accuracy can be limited by the fact that it never considers the whole problem space at once. This is unproblematic when the environment's structure matches the assumption that the rewards of potential goals are more variable than more proximal rewards. But it could be problematic when this assumption is violated too strongly. I mitigated this potential problem by allowing the strategy discovery algorithm to switch goals. Even with this adaptation, the discovered strategy is not optimal in all cases: since the representation of the alternative goal reward is defined as its average expected reward, the algorithm will only switch goals if the current goal's reward is below average. However, if the current goal's expected return is above average, the discovered strategy will not explore other goals even when that would lead to a higher reward. Overall, I believe that the scalability of our method to large environments outweighs this minor loss in resource-rationality score.

The advances presented in this chapter open up many exciting avenues for future extensions. For instance, the approach could be extended to plans with potentially many levels of hierarchically nested subgoals. Future work might also extend the problem decomposition so that any state can be selected as a goal. In its current form, the algorithm always selects only the environment's most distant states (leaf nodes) as candidate goals. Future versions might allow the set of candidate goals to be chosen more flexibly, such that some leaf nodes can be ignored and some especially important intermediate nodes in the tree can be considered as potential sub-goals. A more flexible definition and potentially a dynamic selection of goal nodes could increase the strategy discovery algorithm's performance and possibly allow us to solve a wider range of more complex problems. This would mitigate the limitations of the increasing variance assumption by considering all potentially valuable

states as (sub)goals, regardless of where they are located.

In this chapter, I presented hierarchical-BMPS, a scalable strategy discovery method that overcomes the computational bottleneck that prevents existing strategy discovery problems from being applied to large planning problems. Hierarchical-BMPS is an important step towards intelligent systems with a (super)human-level computational efficiency and understanding how people make decisions. The increased scalability of strategy discovery methods constitutes an essential component of being able to leverage artificial intelligence to improve human decision-making in the real world. At a high level, my findings support the conclusion that incorporating cognitively informed hierarchical structure into reinforcement learning methods can make them more useful for real-world applications.

3

Discovering planning strategies for partially observable environments

This chapter introduces MGPO, a strategy discovery method adapted to partially observable environments, and demonstrates its usefulness in computer simulations and a human training experiment. The chapter is based on my preprint "An intelligent tutor for planning in large partially observable environments" Heindrich et al. (2023), on which I collaborated with Saksham Consul

and Falk Lieder.

3.1 Introduction

While Chapter 2 demonstrated how strategy discovery and cognitive tutors can be effectively scaled to larger environments than previously possible, this chapter addresses another limitation of current strategy discovery methods: all previous research on strategy discovery was conducted in fully observable environments. Discovering cognitive strategies for real-world planning tasks would likely require metacognitive models that feature partial observability, since evaluating the future effects of actions usually involves a high amount of uncertainty.

Previous models of human planning assumed that a computation always reveals the true reward of the simulated action (Callaway et al., 2022b, Griffiths et al., 2019). However, in the real world, thinking about an action's outcomes rarely leads to certainty (Vul et al., 2014, Guez et al., 2012). Instead, each time a person thinks about an action's outcome, the scenario they imagine might be different, and the more uncertain the outcome is, the more variable the imagined outcomes will be. I incorporate this variability into the meta-MDP model of human planning (Callaway et al., 2022b). In my model of planning in partially observable environments, computations produce samples from the agent's probability distribution on what the true reward might be. Specifically, each observation is sampled from a Normal distribution with a fixed precision around the true reward value. This allows observing a single node in the graph multiple times, obtaining multiple samples, and iteratively increasing the accuracy of one's probabilistic belief about the node's true reward. The belief state is modelled as the set of Gaussian posterior distributions of each node's expected reward given the values observed so far. When inspecting a node, the corresponding posterior distribution is updated by conditioning on the new observation. This allows computations to be repeated multiple times, iteratively increasing the accuracy of one's belief. I believe that advancing automatic strategy discov-

ery to partially observable environments is an important stepping stone on the path to leveraging automatic strategy discovery to improve human decision-making in the real world.

To improve human planning in partially observable environments, I make the following contributions. First, I formalize the metacognitive model of human planning in partially observable environments as a meta-MDP (Griffiths et al., 2019, Hay et al., 2014). Second, I introduce the first algorithm that can discover efficient cognitive strategies for human planning in partially observable environments. Third, I develop an adaptive and intelligent tutoring system that teaches people to use the discovered strategies in a fully automated manner and evaluate it in a large online experiment. The results show that the planning strategies people intuitively use in partially observable environments are highly suboptimal, and that teaching people the planning strategies discovered by my method substantially improves the quality of their decisions. This constitutes an important step towards improving human decision-making in the real world.

3.2 Formalizing the strategy discovery problem as a meta-level MDP

The goal of strategy discovery methods is to discover a planning strategy that specifies which computation people should select next given the currently known information about the environment. For the partially observable strategy discovery problem, I again utilize the meta-MDP framework (Griffiths et al., 2019, Hay et al., 2014). The planning problem is defined as the *meta-level* problem, in which actions are stochastic simulations that update the agent's belief about rewards in the *object-level* problem. The object-level problem is deterministic and can be described by a directed acyclic graph $G = (V, E)$. The graph has a start node v_0 and a set of goal nodes $V_G \subseteq V$ that have no child nodes. Nodes $g \in G$ have fixed ground-truth rewards drawn from a multivariate Normal distribution: $(t_1, ..., t_N) \sim (\mathcal{N}(\mu_1, \tau_1), ..., \mathcal{N}(\mu_N, \tau_N))$. I define Paths$(G)$ as the set of paths that start at the start node and end at a goal node. The object-level actions then consist of selecting a path

to traverse, accumulating the ground-truth rewards of the visited nodes.

Following previous work on strategy discovery in other domains (Griffiths et al., 2019, Callaway et al., 2022b), I model the problems of discovering planning strategies for partially observable environments as a meta-level Markov Decision Process $M = (\mathcal{B}, \mathcal{C}, T_{\text{meta}}, r_{\text{meta}})$ (Hay et al., 2014), consisting of a belief state $b \in \mathcal{B}$, a set of computations $\mathcal{C}$, a reward function r_{meta}, and transition probabilities T_{meta}. I now define each of the four components of the meta-level MDP formulation of the strategy discovery problem in turn.

Belief states In general, the belief state $b \in \mathcal{B}$ of a meta-level MDP represents the agent's current knowledge about the environment's reward. I model the agent's beliefs about the rewards at the N locations of the partially observable environment as a multivariate Normal distribution. The entries in the belief state $b_i = (\mu_i, \tau_i)$ represent the agent's posterior distribution about the reward of node $i \in V$ given the information that has been observed so far. The initial belief state is $b^{(0)} = (\mathcal{N}(\mu_1^{(0)}, \tau_1^{(0)}), ..., \mathcal{N}(\mu_N^{(0)}, \tau_N^{(0)}))$. This belief reflects the statistics of the environment, in the sense that the ground-truth rewards of each problem instance are drawn from the same distribution.

Meta-level actions (computations) The meta-level actions $\{c_1, ..., c_n, \perp\} \in \mathcal{C}$ are computations. They comprise planning operations c_i that reveal information about the environment, as well as the termination operation $\perp$ that represents terminating planning and executing the plan with the highest expected reward. Unlike in fully observable environments, in partially observable environments, each meta-level action can be repeated arbitrarily many times.

Meta-level reward function All planning operations have a fixed cost $r_{\text{meta}}(b, c_i) = -\lambda$. Termination leads to a reward that is received from adding the ground truth rewards t_i along the path $p_{\max} \in \text{Paths}(G)$ with the highest expected reward according to the current belief state (Equation 3.1 and 3.2).

$$p_{\max}(b) = \underset{p \in \text{Paths}(g)}{\text{argmax}} \sum_{i \in p} \mu_i \tag{3.1}$$

$$r_{\text{meta}}(b, \bot) = \sum_{i \in p_{\max}} \mathtt{t}_i \tag{3.2}$$

Meta-level transition probabilities The transition probabilities $T_{\text{meta}}(b, c, b')$ specify how the belief state is updated by different meta-level actions. Each computation c_i reveals noisy information $o_i \sim \mathcal{N}(t_i, \tau_{\text{obs}})$ about the expected reward of the corresponding node represented by belief state b_i. The precision parameter τ_{obs} is a global parameter of the environment that specifies how accurate planning operations are. An observation o_i is added to the current belief about that state b_i by computing the posterior mean and precision of the updated belief state and updating the parameters (μ_i, τ_i) (Equation 3.3).

$$\left(\mu_i^{t+1}, \tau_i^{t+1}\right) = \left(\frac{\tau_i^t \mu_i^t + \tau_{\text{obs}} o_i}{\tau_i^t + \tau_{\text{obs}}}, \tau_i^t + \tau_{\text{obs}}\right) \tag{3.3}$$

Together, the initial belief state, the set of computations, the transition probabilities, and the meta-level reward function formalize the problem of discovering resource-rational planning strategies for partially observable environments. I now turn to the algorithm for approximating the optimal policy π^* (see Equation 3.4), the *meta-greedy policy for partially observable environments* (MGPO).

$$\pi^* = \arg\max_{\pi} \mathbb{E}\left[\sum_{t=0}^{T} r_{\text{meta}}\left(b^{(t)}, \pi\left(b^{(t)}\right)\right)\right] \tag{3.4}$$

3.3 The MGPO algorithm

Partial observability considerably increases the difficulty of solving the meta-level MDP because the number of possible sequences of computations is no longer limited by the number of nodes. None of the previous methods could handle this challenge. To overcome this problem, I developed the meta-greedy policy for partially observable environments (MGPO), a meta-greedy planning method that selects computations using a myopic approximation of the VOC. The myopic VOC only considers the immediate benefit of computations: how likely it is that the new information will change the agent's plan. Myopic policies are well established in the fully observable problem setting (Russell and Wefald, 1991a, Lin et al., 2015) and present a natural candidate for adaptation to partially observable settings due to their computational efficiency. To estimate the VOC, it is necessary to estimate the probability that the next noisy observation reveals information that will change the agent's plan after integrating the information into the agent's posterior belief state. I calculate the probabilities and the expected improvement of these events in the partially observable setting by applying the rules of probability theory. Using the myopic VOC calculation, the overall planning strategy is constructed by iteratively choosing the computation with the highest VOC until the cost of planning becomes higher than the expected improvement, at which point planning is terminated, and the plan with the highest expected reward is chosen. A major benefit of MGPO is its computational efficiency, which allows computing the approximated VOC of computations in an online fashion, a critical component to building a feedback-based cognitive tutor.

To discover resource-rational planning strategies in partially observable environments, I approximate the meta-level Q-value $Q(c_i, b)$ of a planning operation c_i in belief state b using a greedy myopic approximation to the value of computation (VOC) (Russell and Wefald, 1991a). Since in the myopic setting, only actions that change the expected best path $p_{\max}$ have a positive VOC, I evaluate the VOC of operation c_i by calculating the probability that an observation o_i will change $p_{\max}$.

Specifically, the VOC is approximated by calculating the probability of changing the expected path times the benefit of changing the expected best path minus the cost of planning.

I define $r_{\max}(b)$ as the expected reward of $p_{\max}$ (Equation 3.5), $r_i(b)$ as the expected reward of the expected best path leading through node i (Equation 3.6), and $r_{\text{alt}}(b)$ as the expected reward of the expected best path that does not lead through node i (Equation 3.7).

$$r_{\max}(b) = \max_{p \in \text{Paths}(g)} \sum_{i \in p} \mu_i \tag{3.5}$$

$$r_i(b) = \max_{(p \in \text{Paths}(G): i \in p)} \sum_{i \in p} \mu_i \tag{3.6}$$

$$r_{\text{alt}}(b) = \max_{(p \in \text{Paths}(G): i \notin p)} \sum_{i \in p} \mu_i \tag{3.7}$$

There are two scenarios in which an observation o_i of the node i can change the optimal path. If the node i does not lie on the path with the highest expected value ($r_{\text{alt}} = r_{\max}$), the best path can only change through observation o_i if the observed value raises the expected value of r_i above r_{alt}. If the node i lies on the path that currently has the highest expected value ($r_i = r_{\max}$), the best path changes if the observed value lowers the expected value of $r_{\max}$ below the value of the best alternative path r_{alt}. In each case, I define a threshold value t that specifies an upper/lower bound on how small/large the observation would have to be to change $p_{\max}$.

For the case where node i does not lie on the optimal path, Equation 3.8 shows how the current observation changes the value of the path through node i from the previous expected value for node i (i.e., μ_i) to the expected value after the observation (i.e., μ_i'; see Equation 3.3). If the left-hand side of Equation 3.8 is larger than its right-hand side, the path through v_i now has a higher expected value than the previous best path. Solving for the observation o_i that makes both sides equal gives us

the threshold t defined in Equation 3.9.

$$r_i - \mu_i + \frac{\tau_i \mu_i + \tau_{\text{obs}} o_i}{\tau_i + \tau_{\text{obs}}} > r_{\text{alt}} \tag{3.8}$$

$$o_i > t = \left(1 + \frac{\tau_i}{\tau_{\text{obs}}}\right)(r_{\text{alt}} - r_i) + \mu_i \tag{3.9}$$

I will now present the computations for the first case (node i does not lie on the expected best path) in detail (see line 6 to 10 of Algorithm 1). For the alternative case (node i lies on the expected best path), the VOC can be computed analogously (see line 11 to 15 of Algorithm 1).

So far, I have calculated the threshold t an observation o_i needs to exceed to change the expected best path. I now move to estimating the potential improvement in the new expected best path through the updated belief state b_i. The probability of receiving an observation o_i that is higher than t is given by the Normal distribution's cumulative distribution function (Equation 3.10). Assuming the threshold is reached, the expected value of an observation that lies above the threshold is then given by the expected value of a Normal distribution that has been truncated at the threshold (see Equation 3.11). Using o_{change} as the expected value of such an observation, I use Equation 3.3 to compute the conditional posterior expectation μ_i' given that the observation turns the inspected path into the optimal path.

To compute the myopic VOC, I calculate the overall expected improvement in r_{max} (the probability of changing the expected best path times the benefit of changing the expected best path) while taking the cost of planning into account. Equation 3.12 shows the calculation, where the expected improvement $(r_i + \mu_i' - r_{\text{alt}} - \mu_i)$ is the difference between the value of the updated path through node i and the value of the previous best path r_{alt}. The expected improvement is multiplied by the probability of observing an improvement (i.e., p_{change}; see Equation 3.10) and the cost of the planning operation is subtracted to arrive at the myopic VOC.

Since the VOC calculation is purely myopic, it can underestimate the VOC due to not taking the value of additional future information into account. This is especially problematic when deciding whether to terminate planning (i.e. whether the VOC of the expected best computation exceeds the cost of planning λ). To circumvent the issue of terminating planning too early, I introduce a new hyperparameter $w_\lambda \in [0, 1]$, which can be used to adjust the amount of planning before the algorithm terminates by scaling the (imagined) cost of planning in the VOC calculation. I optimize w_λ using 50 steps of Bayesian optimization (Mockus, 2012).

$$p_{\text{change}} = 1 - \Phi\left(\frac{t - \mu_i}{\sigma_{\text{obs}}}\right) \tag{3.10}$$

$$o_{\text{change}} = \mu_i + \sigma_{\text{obs}} \frac{\varphi\left(\frac{t-\mu_i}{\sigma_{\text{obs}}}\right)}{1 - \Phi\left(\frac{t-\mu_i}{\sigma_{\text{obs}}}\right)} \tag{3.11}$$

$$\hat{voc}(c_i, b) = (1 - w_\lambda)(p_{\text{change}})(r_i - r_{\text{alt}} - \mu_i + \mu_i') - w_\lambda \lambda \tag{3.12}$$

3.4 Evaluating MGPO in simulations

To evaluate how resource-rational the strategy discovered by MGPO is, I compared its resource-rationality against two baseline methods in two simulation experiments.

3.4.1 Benchmark problems: strategy discovery for partially observable environments

The simulation experiments are run on the 4 benchmark environments of different sizes introduced in Section 2. In line with their previous use, the environments are built from 2 to 5 blocks of 18 nodes, each leading to a different goal node. The environments all follow an increasing variance

Algorithm 1 Myopic VOC calculation for a single action

function MYOPIC_VOC(c_i, b)
  $r_i \leftarrow \max_{(p \in \text{Paths}(G) | v_i \in p)} \sum_{v_i \in p} \mu_i$
  $r_{\text{alt}} \leftarrow \max_{(p \in \text{Paths}(G) | v_i \notin p)} \sum_{v_i \in p} \mu_i$
  $t \leftarrow \left(1 + \frac{\tau_i}{\tau_{\text{obs}}}\right)(r_{\text{alt}} - r_i) + \mu_i$
  $\sigma_{\text{obs}} \leftarrow \sqrt{\left(\frac{1}{\sqrt{\tau_{\text{obs}}}}\right)^2 + \left(\frac{1}{\sqrt{\tau_i}}\right)^2}$
  if $r_i > r_{\text{alt}}$ **then**
    $p_{\text{change}} \leftarrow \Phi\left(\frac{t-\mu_i}{\sigma_{\text{obs}}}\right)$
    $o_{\text{change}} \leftarrow \mu_i - \sigma_{\text{obs}} \frac{\varphi\left(\frac{t-\mu_i}{\sigma_{\text{obs}}}\right)}{\Phi\left(\frac{t-\mu_i}{\sigma_{\text{obs}}}\right)}$
    $\mu_i' \leftarrow \frac{\mu_i \tau_i + o_{\text{change}} \tau_{\text{obs}}}{\tau_i + \tau_{\text{obs}}}$
    $\text{voc} \leftarrow \left(p_{\text{change}}\right)\left(r_{\text{alt}} - r_i + \mu_i - \mu_i'\right)$
  else if $r_i <= r_{\text{alt}}$ **then**
    $p_{\text{change}} \leftarrow 1 - \Phi\left(\frac{t-\mu_i}{\sigma_{\text{obs}}}\right)$
    $o_{\text{change}} \leftarrow \mu_i + \sigma_{\text{obs}} \frac{\varphi\left(\frac{t-\mu_i}{\sigma_{\text{obs}}}\right)}{1-\Phi\left(\frac{t-\mu_i}{\sigma_{\text{obs}}}\right)}$
    $\mu_i' \leftarrow \frac{\mu_i \tau_i + o_{\text{change}} \tau_{\text{obs}}}{\tau_i + \tau_{\text{obs}}}$
    $\text{voc} \leftarrow \left(p_{\text{change}}\right)\left(r_i - r_{\text{alt}} - \mu_i + \mu_i'\right)$
  end if
  return $(1 - w_\lambda)\text{voc} \text{ - } w_\lambda \lambda$
end function

structure, where the uncertainty about the reward of the node increases the further away it is from the starting node (see Figure 3.1). The rewards for each environment instance were initialized randomly following an increasing-variance reward structure, where nodes closer to the leaves had a higher variance in their reward (see Figure 3.1 for an example of the smallest evaluation environment and an explanation of the reward structure). To make the environment partially observable, the observations generated by the planning operations were randomly sampled from Normal distributions centered around the environment's ground truth values with a fixed precision parameter $\tau = 0.005$. The cost of computations was fixed to $\lambda = 1$ (matching the previous experiment in Section 2) for the first simulation experiment, and $\lambda = 0.05$ for the second simulation experiment. All methods were evaluated by their average meta-level return (i.e. their expected termination reward) on the same 5000 randomly generated environment instances.

3.4.2 Baseline methods

For the baseline methods, I selected a discretized version of the myopic VOC as used in Section 2.5 and PO-UCT (Silver and Veness, 2010), a sample-based planning algorithm utilizing Monte-Carlo tree search that is suitable for partially observable environments. The reason for choosing the discretized myopic VOC method was that it allowed me to apply an established baseline method to partially observable environments. While PO-UCT has not been applied to meta-MDPs yet, it allows planning multiple steps into the future through sampling. I chose PO-UCT over the full POMCP algorithm presented by Silver and Veness (2010) because the meta-MDP model allows access to the exact updated belief state.

Discretization The first baseline method approximates the meta-greedy policy (Russell and Wefald, 1991a) using discretization. When calculating the VOC, I discretized the continuous observation that each planning operation generated into a categorical variable with four possible values

based on the current belief of the expected mean μ and variance σ for each node i: $(\mu_i - 2\sigma_i, \mu_i - \frac{2}{3}\sigma_i, \mu_i + \frac{2}{3}\sigma_i, \mu_i + 2\sigma_i)$. I calculated the probability that the observation would fall into a given bin using the Normal distribution's cumulative distribution function. Using these 4 possible outcomes for every planning operation, the action with the highest immediate increase in expected reward is chosen while assuming that planning will terminate after the selected planning operation has been carried out:

$$c = \arg\max_{c \in \mathcal{C}} \sum_{(p,b') \in \text{disc}(T(b,c,\cdot))} (p)(r(b', \perp) - r(b, \perp) + r(b, c)) \tag{3.13}$$

where $\text{disc}(T(b, c, \cdot))$ is the vector containing the probabilities and posterior belief states of the transitions that can result from witnessing the four possible values of discretized observation.

Environment (Goals)		2	3	4	5	2	3	4	5
Cost	Steps	Exploration Coefficient				Rollout Depth			
0.05	10	100.0	100.0	100.0	100.0	0.0	3.0	3.0	0.0
	100	100.0	100.0	10.0	5.0	3.0	3.0	3.0	3.0
	1000	1.0	5.0	10.0	100.0	3.0	3.0	3.0	3.0
	5000	5.0	50.0	5.0	5.0	0.0	0.0	3.0	3.0
1.00	10	100.0	100.0	100.0	100.0	0.0	3.0	3.0	0.0
	100	10.0	100.0	5.0	50.0	0.0	0.0	0.0	0.0
	1000	100.0	10.0	50.0	100.0	0.0	0.0	0.0	0.0
	5000	5.0	100.0	100.0	50.0	0.0	0.0	0.0	0.0

Table 3.1: Exploration coefficient and rollout depth for PO-UCT simulations selected by a hyperparameter grid search.

Monte-Carlo Tree Search Secondly, I also evaluated the method against the Monte-Carlo tree search algorithm PO-UCT (Silver and Veness, 2010). My implementation used the same discretized state representation with 4 bins per node when running the Monte-Carlo simulations as described in the discretized approximation of the meta-greedy policy. I discretized the simulated belief updates during the sample-based simulations to allow the tree search planning to reach states

multiple times and plan multiple steps into the future. I evaluated 4 different versions of the algorithm that have different numbers of evaluation steps per action (10 steps, 100 steps, 1000 steps, 5000 steps), of which the lowest was set to roughly match the computation time of my method. While traversing the search tree, nodes are selected using UCB1 (Silver and Veness, 2010). When a leaf node is reached, a rollout policy is used to estimate the node's value. The rollout policy consists of performing a fixed number of uniformly random computations $c \in \mathcal{C}$, after which the termination action $\perp$ is selected. I ran a hyperparameter search over 500 training environments with the rollout depth set to either 0 or 3 and exploration coefficients set to one of $\{0.5, 1, 5, 10, 50, 100\}$. For each computational budget used in the simulation experiment, I selected the parameter settings that achieved the highest resource-rationality score and evaluated PO-UCT with the selected parameters on the 5000 test environments. The selected parameters for each environment and cost configuration are listed in Table 3.1.

3.4.3 Results

Table 3.2 shows each method's average resource-rationality scores (*RR-score*). I analyzed the results using a three-way ANOVA to determine the main effect of the algorithm on the *RR-score* for different planning costs and different sizes of the environment. The ANOVA revealed that some methods tended to discover significantly more resource-rational planning strategies than others ($F(5, 239952) = 32682.5, p < .001$). A Tukey-HSD post-hoc comparison showed that MGPO performs significantly better than the meta-greedy policy ($p = .001$), the PO-UCT algorithm with 5000 steps ($p = .008$), and all PO-UCT algorithm configurations with less than 5000 steps (all $p < .001$).

On average, MGPO took 0.03 seconds to select the next computation. The only baseline method with a comparable runtime is PO-UCT with 10 steps (0.02 seconds). However, with only 10 computation steps, PO-UCT fails to perform any meaningful planning and performs much worse than

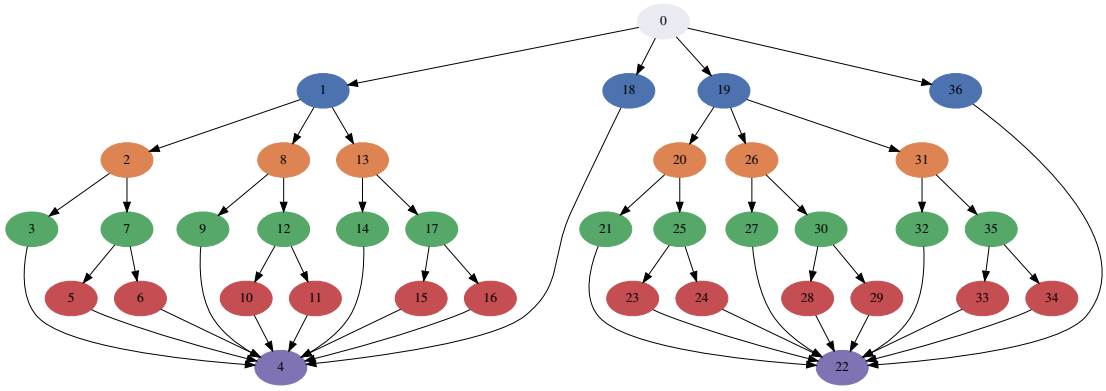


Figure 3.1: Environment used in the simulation evaluation. The reward of each node is independently sampled from a Normal distribution: $\mathcal{N}(0, 5)$ for nodes colored in blue, $\mathcal{N}(0, 10)$ for orange nodes, $\mathcal{N}(0, 20)$ for green nodes, $\mathcal{N}(0, 40)$ for red nodes, $\mathcal{N}(0, 100)$ for node 4, and $\mathcal{N}(0, 120)$ for node 22.

all other tested methods (see Table 3.2). The other baseline methods required a computational budget more than a 100 times higher than the budget used by the MGPO policy: the meta-greedy policy required 4.89 seconds, PO-UCT with 100 steps 0.22 seconds, PO-UCT with 1000 steps 3.22 seconds, and PO-UCT with 5000 steps 14.19 seconds.

In summary, my results show that MGPO is consistently better than alternative methods with a similar computational budget and much faster. Additional analysis of the interaction effects can be found in the supplementary information.

3.5 A cognitive tutor for planning in partially observable environments

In this section, I explain the functionality of the cognitive tutor, starting with the computation of metacognitive feedback in partially observable environments, and then detailing the learning curriculum inspired by Skinner's shaping method (Skinner, 1953).

Cost	Environment (Goals) Algorithm	2	3	4	5
0.05	MGPO (ours)	**118.97**	**158.28**	**191.14**	**223.84**
	Meta-greedy policy	115.47	154.33	186.91	218.45
	PO-UCT 10 steps	55.00	55.00	55.03	55.20
	PO-UCT 100 steps	114.94	152.42	186.17	218.75
	PO-UCT 1000 steps	115.46	155.96	188.76	220.57
	PO-UCT 5000 steps	117.41	156.53	189.51	220.94
1.00	MGPO (ours)	**104.75**	**142.27**	**173.75**	**205.27**
	Meta-greedy policy	103.69	141.46	173.23	204.27
	PO-UCT 10 steps	-135.95	-135.67	-135.89	-135.64
	PO-UCT 100 steps	98.11	133.79	160.86	189.61
	PO-UCT 1000 steps	102.23	138.33	171.64	202.69
	PO-UCT 5000 steps	100.66	140.10	172.20	203.21

Table 3.2: Mean resource-rationality score for both cost settings and all four environments. Results are averaged over 5000 environment instances. The PO-UCT versions differ by the number of simulation steps given for each computation.

3.5.1 Giving metacognitive feedback on how people plan in partially observable environments

To teach planning strategies to people, I adapted the cognitive tutor introduced by (Callaway et al., 2022a). Since their tutor used dynamic programming to compute the exact Q-function of the meta-level MDP, their approach was intractable on larger planning environments. Moreover, their tutor cannot handle partially observable environments. While the tutor presented in Chapter 2 teaches planning strategies solely through video demonstrations, the added complexity of partial observability is better suited to feedback-based cognitive tutors that can correct users' mistakes during learning.

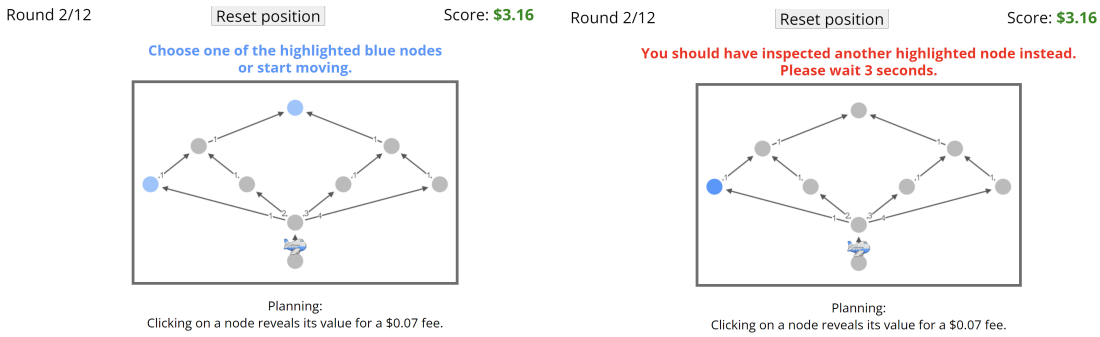
I applied the MGPO algorithm to improve the cognitive tutor introduced by (Callaway et al., 2022a) so that it can discover and teach good planning strategies for environments that are large and partially observable. Since the VOC values computed by MGPO are only approximate, I decided

to impose the same delay penalty ($d(c_i)$) for all planning operations c_i who have a negative VOC or whose VOC differs by more than a fixed threshold of 0.05 from the action with the highest VOC according to MGPO. This threshold was selected to strike a balance between teaching users near optimal planning strategies while also avoiding to get stuck in negative feedback loops when repeatedly choosing the lesser of two almost identical choices and accounting for minor errors caused by the myopic approxmation of MGPO. Choosing the termination action $\perp$ when MGPO does not terminate planning leads to an additional delay penalty that was proportional to an estimate of how valuable it would have been to do more planning in the current belief state. The termination delay penalty is calculated by taking the proportion between the VOC of the best meta-action in the current belief state and the highest VOC observed in any earlier planning step (see Equation 3.14). Intuitively, this proportion describes how valuable planning in the current belief state is when compared to the initial belief state, where the delay for falsely terminating planning is set to $d_{\max}$. In my implementation, I set d_c to 3 seconds and $d_{\max}$ to 4 seconds. These simplifications to the delay calculation make it feasible to compute metacognitive feedback for environments, for which this was previously intractable because of their large size and partial observability.

$$d\left(\perp, b^{(i)}\right) = d_c + d_{\max}\frac{\max_{c\in C}\text{MYOPIC\_VOC}\left(c, b^{(i)}\right)}{\max_{c\in C, j\in[0,i)}\text{MYOPIC\_VOC}\left(c, b^{(j)}\right)} \tag{3.14}$$

3.5.2 A shaping method for teaching cognitive strategies

To further help people learn planning strategies, I introduce an improved teaching methodology and training schedule to the cognitive tutor. The cognitive tutors developed by (Lieder et al., 2019a, Callaway et al., 2022a) let the learner freely choose their next planning operation as they practice. This causes a potential problem in large environments: the more actions the learner can choose from, the more attempts it will take them to stumble upon the optimal planning operation, and



(a) The tutor offers participants a choice between two computations.

(b) After selecting an incorrect computation, the tutor gives feedback and highlights the correct choice.

Figure 3.2: Example of the cognitive tutor in the simplified 8-node environment.

the more errors and delay penalties they must endure along the way. For environments with many possible planning operations, the learning process can thus become very slow and very frustrating.

To make the cognitive tutor more suitable for teaching planning in large environments, I leverage the method of successive approximations developed in research on animal learning (Skinner, 1953). That is, the training starts with a minimal version of the planning task, and as the training progresses, the planning task becomes increasingly more similar to the scenario in which people's planning strategies shall be improved. This gradual approximation of a complex planning problem proceeds along two dimensions: the number of planning operations the learner can choose between and the size of the environment.

Concretely, I created three smaller versions of the full 60-node environment (see Figure 3.3) and increased the number of planning operations the learner is asked to choose between in tandem with the environment size: an 8-node environment with 1 goal node and 2 choices, a 16-node environment with 2 goal nodes and 3 choices, and a 30 node environment with 2 goal nodes and 4 choices (see Figure 3.2 for an example of the 8-node environment).

To offer meaningful choices (i.e., choices with different VOC values), the choices presented to the learner are selected by grouping available computations by their VOC value. The set of choices

always includes the planning operation with the highest VOC. The remaining choices are filled up by iteratively adding a computation with a VOC value different from the VOC values of the computations already included in the choice set, uniformly at random. In addition to selecting one of the offered choices, participants were also always given the option to terminate planning. To ensure that participants experienced the complete planning strategy, choosing this option prematurely did not actually lead to termination when MGPO's strategy did not terminate planning itself and instead only resulted in a delay penalty as described in Section 3.5.1.

Besides training trials with feedback on the selected choices, the cognitive tutor also demonstrated the taught strategy through step-by-step demonstrations. The demonstrations were created by repeatedly performing the computation selected by MGPO's planning strategy.

3.6 Assessing the cognitive tutor in an experiment

Based on the superior performance and computational speed of MGPO, I evaluated the cognitive tutor in an online experiment. In the experiment, I tested if human resource-rationality can be improved by teaching them MGPO's strategy in interactive training sessions. To implement the experiment, I again used the Mouselab-MDP paradigm (Callaway et al., 2017, 2022b, Jain et al., 2022), in which the internal human planning process is externalized through information retrieval actions.

Participants I recruited 330 participants through Prolific (www.prolific.co), of which 37 were excluded through predefined exclusion criteria (for details, refer to the study's pre-registration: https://aspredicted.org/RL3_YDD). The average age was 28.2 and of the 330 participants, 164 identified as female. All participants were over 18 years old, fluent in the English language, and gave informed consent to participate in the experiment. Participants were paid £3.80 and a performance dependent bonus of up to £2 (average bonus £0.91). The median duration of the experiment was 40.48 minutes, resulting in an overall average wage of £6.98 per hour.

Procedure Participants were randomly assigned one of three conditions: the experimental condition (*Choice Tutor*) where participants trained with a cognitive tutor teaching MGPO's strategy [*], a control condition without a cognitive tutor (*No Tutor*), and a control condition with a *Dummy Tutor*. Participants first received instructions explaining the flight planning task; these instructions were identical across all conditions. Then, participants underwent 12 rounds of training. The training trials utilized the smaller environments described in Section 3.5.2, slowly building up to the full problem complexity: the first 3 training trials used the 8-node environment, trials 4 to 6 used the 16-node environment, trials 7-9 used the 30-node environment, and the final 3 training trials used the full 60-node environment. The type of training depended on the assigned condition: participants of the *Choice Tutor* and *Dummy Tutor* conditions received one demonstration and two feedback trial for each environment size, while participants in the *No Tutor* condition practiced unassisted in three practice trials for each environment size. After completing the training section, participants of all conditions were evaluated in 10 unassisted test trials in the full 60-node environment.

Materials The evaluation took place in the flight planning scenario introduced in Section 2.6. In this experiment, participants plan the route of an airplane over a network of 60 airports (see Figure 3.3). I adapted the scenario to a partially observable setting. Concretely, in the partially observable version, inspecting an airport does not reveal the true cost or reward of traversing it; instead, it generates a stochastic observation of what that reward might be. Each observation was drawn from

[*]The version of MGPO used in the experiment was not the most recent version of my method. Its VOC calculation differed from the current version in two ways. First, the sigma value for observations was calculated taking only the precision of observations into account: $\sigma_{\text{obs}} \leftarrow \frac{1}{\sqrt{\tau_{\text{obs}}}}$ (cf. Line 5 of Algorithm 1). Second, the returned VOC value was calculated without optimizing the cost weight parameter: w_λ (cf. Line 15 of Algorithm 1). These discrepancies make the strategy taught by the cognitive tutor inferior to the strategy discovered by the current version of MGPO by an average difference in resource-rationality score of 5.61 (evaluated over 500 instances of the 60 node environment, a precision of 0.005, and a cost value of 0.05). Therefore, if I had taught participants the more resource-rational strategy discovered by the newer version of MGPO, the benefit of the cognitive tutor would potentially have been even larger. This makes the results of the experiment a lower bound on the effectiveness of the cognitive tutor.

a list of 200 precomputed samples from the observation distribution of the corresponding meta-level MDP. To make the task manageable for people, the posterior means of the airports' rewards were shown to the participant and updated with each observation. To make the performance of the three groups as comparable as possible, all three conditions used the same set of precomputed random observations and environment rewards. Furthermore, I changed the environment's reward distributions to the increasing variance structure described in Figure 3.3.

The cost of clicking and the precision of observations were randomized across participants. The cost values were sampled from a Normal distribution $\lambda \sim \mathcal{N}(0.05, 0.002)$ and clipped to $[0.01, 0.5]$, and precision values were sampled from a Normal distribution $\tau \sim \mathcal{N}(0.005, 0.002)$ and clipped to $[0.0001, 0.1]$. These parameters and clipping ranges were chosen manually to ensure a reasonable amount of planning in the resulting optimal strategies. A set of 100 parameter combinations was pre-generated to allow generating observation samples in advance and ensure a comparable distribution of parameters between conditions.

Participants in the *Choice tutor* condition were trained by the intelligent cognitive tutor described in Section 3.5. The Dummy Tutor was designed to reveal similar information about the environment's structure as the Choice Tutor, without revealing the strategy taught by the Choice Tutor. To achieve this, the Dummy Tutor differed from the Choice Tutor in three ways. First, whereas the Choice Tutor always includes the optimal planning operation in the choice set, the Dummy Tutor asks the learner to choose from a randomly generated set of planning operations. Concretely, the Dummy Tutor first randomly selects the distance from the starting position (e.g., 3 steps) and then randomly selects two airports that are that far away from the starting position. Second, when video demonstrations are shown to the participant, the meta-level actions are selected at random. In the video demonstrations and practice trials of the dummy tutor condition, the overall number of meta-level actions was fixed to the number of meta-level actions performed by MGPO's strategy. Third, unlike in the Choice Tutor condition, the number of choices offered by the Dummy Tutor was kept

at 2 throughout training. The reason was that because participants in the Dummy Tutor condition received random feedback, they cannot be expected to master even the simplified selection between only two choices.

Data analysis I calculated the five performance measures (i.e., *RR-score*, *click agreement*, *termination agreement*, *repeat agreement*, and *goal planning*) as follows. The *RR-score* measures how resource-rational a planning strategy is by computing the expected value of executing a plan while taking the cost of the used planning operations into account. The participant's *RR-score* was calculated by adding the posterior expectation given the participant's observations of all nodes on the chosen object-level path and subtracting the total cost of all performed meta-level actions. The *click agreement* score quantifies how similar to MGPO's strategy the participant's chosen meta-level actions are. This measure is computed for each trial by taking the proportion of meta-level actions chosen by the participant that match the meta-level action chosen by MGPO in the same belief state. Similarly, *repeat agreement* measures to which extent the participant's strategy and MGPO's strategy agreed on whether to collect more information on an already inspected node versus inspecting a new node. I defined this measure as the proportion of the participant's meta-level actions that either repeated a meta-level action when MGPO's strategy would also perform a (potentially different) repeated meta-level action, or performed a non-repeated meta-level action when MGPO's strategy would also perform a (potentially different) not repeated meta-level action. I considered a meta-level action as repeated if it had been chosen before at any earlier time step. Lastly, *termination agreement* measures if participants learned when to terminate selecting meta-level actions if and only if the strategy discovered by MGPO would terminate planning. This measure was calculated using a balanced accuracy measure (Brodersen et al., 2010) where true positive cases are defined as terminating planning when MGPO's strategy also terminates planning; true negative cases are defined as not terminating when MGPO's strategy also did not terminate planning; false positive cases

are defined as terminating planning when MGPO's strategy continued to plan; and false negative cases are defined as not terminating planning when MGPO's strategy terminated planning.

We also investigated if participants learned to follow a general *goal planning* strategy of first examining nodes furthest away from the root node and then planning backwards, which is a component of MGPO's strategy. For each trial, I calculated whether a participant followed the *goal planning* strategy by testing whether their selected meta-level actions are consistent with the increasing variance reward structure of the environment (see Figure 3.3). Specifically, I tested whether participants performed at least one meta-level action in the nodes with the highest variance (goal nodes) before investigating a node with a lower variance, and whether participants investigated at least one node with a medium variance before investigating a node with a low variance. Participants were considered to have learned this goal-directed planning strategy if they applied it in more than half of the test trials.

I analyzed the main differences between conditions for the *RR-score*, *click agreement*, *repeat agreement*, and *termination agreement* using a robust ANOVA-type statistic with Box approximation (Box, 1954) implemented in the nparLD R package (Noguchi et al., 2012) and used additional ANOVA-type statistics for pairwise post hoc comparisons. I used z-tests to compare the proportion of participants who learned to follow the *goal planning* strategy and corrected for multiple comparisons using the Benjamini-Hochberg method (Benjamini and Hochberg, 1995).

3.6.1 Results

The results of the experiment are summarized in Table 3.3. I start by comparing the unaided human performance (*No Tutor* condition) to the performance of MGPO when evaluated on the same set of environments. MGPO's strategy achieves a median *RR-score* of 22.70 (interquartile range: 21.25), compared to the lower *RR-score* of 11.46 achieved by participants in the *No Tutor* condition. A Wilcoxon signed-rank test confirmed that this difference is significant ($T = 102373, p < .001$),

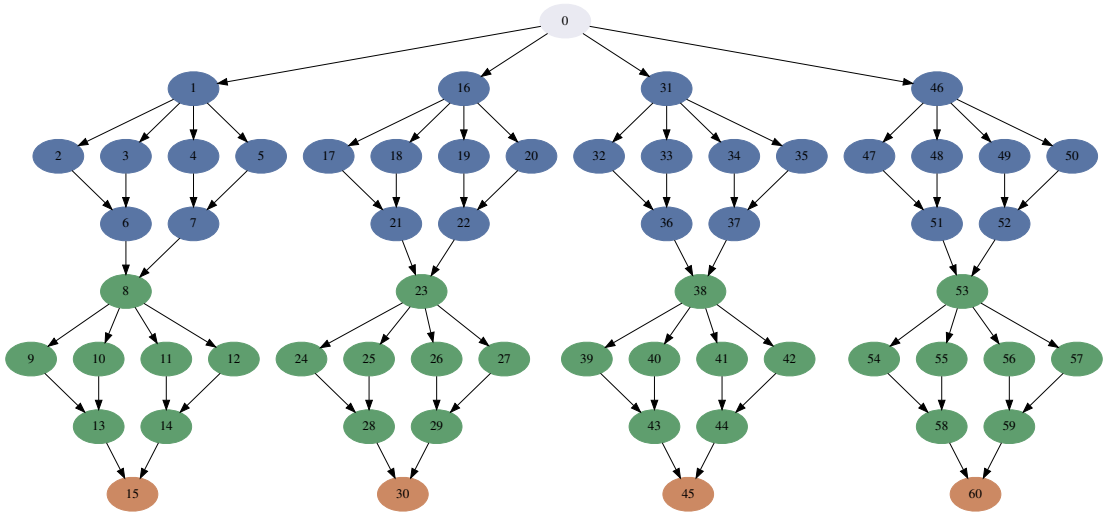


Figure 3.3: The environment used in the online experiment. Rewards are sampled from Normal distributions depending on the node: $\mathcal{N}(0, 5)$ for nodes close to the root (blue), $\mathcal{N}(0, 10)$ for intermediate nodes (green), and $\mathcal{N}(0, 20)$ for nodes furthest from the root (orange).

indicating that people's strategies for solving partially observable meta-level MDPs are suboptimal and can potentially be improved by teaching them MGPO's superior planning strategy. Comparing participants in the *Choice tutor* to MGPO's strategy confirmed the effectiveness of the cognitive tutor. Participants in the *Choice tutor* condition achieved a *RR-score* of 22.67 while MGPO achieved a *RR-score* 22.41 (interquartile range: 21.98) on the same environments. Even though participants of the *Choice Tutor* condition didn't learn to follow the demonstrated strategy exactly, training with the cognitive tutor taught them a similarly efficient planning strategy that numerically even slightly surpassed MGPO's strategy in the evaluation environments. Using a Wilcoxon signed-rank test, I did not find a significant difference between participants in the *Choice tutor* condition and MGPO's strategy ($p = .453$). I now continue the analysis with a detailed comparison between the three conditions.

Participants' *RR-scores* on the test trials differed significantly between the three conditions ($F(1.9, 259.58) = 33.22, p < .001$). Post hoc ANOVA-type statistics showed that the perfor-

Condition	*Goal Strategy*	*RR-Score*		*Click Agreement*		*Termination Agreement*		*Repeat Agreement*	
		Median	IQR	Median	IQR	Median	IQR	Median	IQR
Choice Tutor	71.4%	22.67	26.14	0.32	0.24	0.94	0.05	0.0	0.33
No Tutor	16.1%	11.46	19.68	0.12	0.25	0.93	0.07	0.0	0.00
Dummy Tutor	43.1%	14.11	23.34	0.25	0.30	0.92	0.09	0.0	0.00

Table 3.3: Median results and Inter Quartile Range (IQR) of participants' performance in the experiment. The *goal planning* strategy measure is reported as the percentage of participants who learned the desired behavior.

mance of participants in the *Choice Tutor* condition was significantly higher than the performance of participants in the *No Tutor* ($F(1) = 57.90, p < .001$) and participants in the *Dummy Tutor* condition ($F(1) = 50.52, p < .001$). There was no significant difference in the performance of the participants of the two control conditions ($F(1) = 1.14, p < .29$).

The strategy discovered by MGPO followed a general goal-directed planning strategy of first examining nodes the furthest away from the root node to select one or more potential goals (goal-setting). Planning backward from promising goals, it then inspects intermediate nodes, and lastly the nodes closest to the root node. My analysis showed that significantly more participants in the *Choice Tutor* condition learned the *goal planning* component of MGPO's strategy than participants in the *No Tutor* condition ($z = 7.69, p < .001$) and participants in the *Dummy Tutor* condition ($z = 4.04, p < .001$).

I investigated how closely participants followed MGPO's strategy using a measure called *click agreement*. I found significant differences in *click agreement* between conditions using the ANOVA-type statistic with Box approximation (Box, 1954) ($F(1.98, 283.42) = 49.67, p < .001$). Post hoc ANOVA tests showed that participants in the *Choice Tutor* condition had a significantly higher *click agreement* than participants in the *No Tutor* condition ($F(1) = 107.42, p < .001$) and *Dummy Tutor* condition ($F(1) = 16.14, p < .001$).

To evaluate how well participants learned when to stop planning, I compared when planning

was terminated by participants versus MGPO's strategy. The ANOVA-type statistic with Box approximation (Box, 1954) confirmed significant differences between conditions ($F(1.93, 267.49) = 10.39, p < .001$). Post hoc ANOVA tests showed that participants in the *Choice Tutor* condition achieved a significantly higher *termination agreement* than participants in the *No Tutor* condition ($F(1) = 4.75, p = .029$) and *Dummy Tutor* condition ($F(1) = 23.63, p < .001$).

Lastly, I investigated if participants understood the value of investing additional computations by repeating their clicks as often as MGPO's strategy repeats computations. The ANOVA-type statistic with Box approximation (Box, 1954) revealed significant differences in *repeat agreement* between conditions ($F(1.96, 273.48) = 15.41, p < .001$). Post hoc ANOVA-type statistics showed that participants in the *Choice Tutor* condition achieved a significantly higher *repeat agreement* than participants in the *No Tutor* condition ($F(1) = 22.45, p < .001$) and the *Dummy Tutor* condition ($F(1) = 25.33, p < .001$). Overall, participants struggled to learn when to repeat a computation, resulting in a median *repeat agreement* of 0. Differences between conditions are therefore only visible in the mean performance scores, with participants of the *Choice tutor* condition achieving a mean repeat agreement of 0.18, compared to 0.08 for the *No tutor* condition, and 0.09 for the *Dummy tutor* condition.

3.6.2 Analysis of the discovered strategy

To better understand the strategy discovered by MGPO, I inspected the computations it performs in a sample environment (see Figure 3.4 for the performed computations and Figure 3.3 for the reward distribution of the different nodes). In this environment, all paths to a final goal at the bottom of the graph have to pass through a distinct bottleneck in the middle of the graph. Therefore, each final goal is naturally associated with one specific subgoal. The planning strategy discovered by MGPO follows a general pattern: 1) first selecting a final goal by inspecting the orange nodes in Figure 3.3, 2) then selecting a path from the corresponding subgoal to the final goal by inspecting the

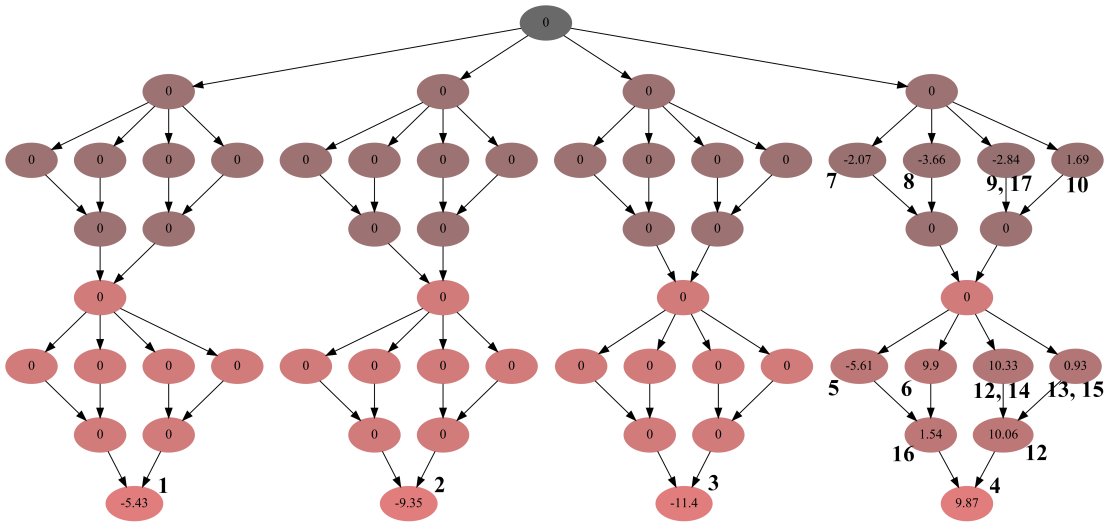


Figure 3.4: Example of the computed strategy in a specific environment instance. The numbers in the nodes show the expected reward of the nodes according to the current belief state. The node's color indicates the uncertainty, ranging from no uncertainty (gray) to high uncertainty (red). The numbers next to the nodes indicate in which order the nodes have been selected by the planning strategy discovered by my method.

green nodes in Figure 3.3, and 3) planning how to reach the subgoal by inspecting the blue nodes in Figure 3.3. This order reflects that the variance of the reward distribution is highest for the final goals, second highest for the nodes between the subgoal and the final goal, and lowest for the nodes between the starting location and the subgoals. Inspecting the nodes between the subgoal and the final goal could, in principle, overturn the chosen final goal, but inspecting the nodes before the subgoal cannot.

In the first step, the strategy inspects potential final goals until a good goal has been identified. This can include repeating computations for one or more goals if they are sufficiently close in expected reward. In the second step, intermediate nodes leading to the selected goal are investigated until a sufficiently good subgoal has been identified. Here, again, nodes can be sampled multiple times if they show similar expected rewards. In the third and final step, the nodes close to the root node that lead to the selected goal are investigated. The algorithm then tries to improve upon the

current best path by repeatedly sampling additional alternatives if their VOC is sufficiently high.

3.6.3 What did people learn from the tutor?

Understanding the strategy discovered by MGPO allowed me to investigate which parts of the strategy participants in the experiment did and did not learn. To achieve this, I analyzed to which extent participants' click sequences exhibited the key three-step structure of the MGPO's strategy described in Section 3.6.2. I already reported which percentage of participants follow the general structure of starting to plan by identifying a goal, then moving towards intermediate nodes, and ending with planning a path in immediate nodes in Section 3.6.1 (see *Goal Strategy*). This section provides additional detail investigating the errors participants made.

Setting a final goal I first evaluated how well participants learned the first step of the MGPO strategy (i.e., starting by selecting a final goal) using two measures. First, I compared the percentage of participants who started their planning by inspecting a goal node. In the *Choice tutor* condition, 84% of participants selected a goal node first, while only 55% of the *Dummy tutor* condition and 23% of the *No tutor* condition learned this part of the strategy. Especially in the *No tutor* condition, participants failed to learn which nodes to prioritize and started with the least valuable node type, immediate nodes, in 64% of test trials.

Second, I investigated which percentage of participants learned the full goal selection aspect of MGPO's strategy. To calculate this measure, I computed participants' *click agreement* for consecutive goal node selections at the start of planning. In the *Choice tutor* condition, participants reached a *click agreement* of 57%, compared to 40% for participants in the *Dummy tutor* condition and 16% for participants in the *No tutor* condition. While the most common errors in the *No tutor* and *Dummy tutor* conditions were to not start with goal planning at all (77% and 45% of test trials), in the *Choice tutor* condition the most common error was overplanning which goal to pursue (69% of

test trials), a less costly mistake since incurring an additional cost of computation results in a smaller decrease in *RR-score* compared to not knowing which goal to pursue.

Planning the path from the corresponding subgoal to the final goal The second step of the MGPO's strategy was to inspect intermediate nodes to plan how to get to the final goal from the corresponding subgoal (bottleneck). Therefore, I analyzed which percentage of participants learned to inspect an intermediate node after inspecting at least one final goal node but before inspecting any immediate nodes. In the *Choice tutor* condition, 67% of participants learned this aspect of the strategy, compared to 37% in the *Dummy tutor* condition and only 15% in the *No tutor* condition. The most common mistake made by participants in the *Dummy tutor* (33% of test trials) and the *No tutor* (69% of test trials) conditions was to plan intermediate nodes before planning which goal to pursue. The most common mistake of participants in the *Choice tutor* condition was to select an intermediate node only after having already selected at least one immediate node (19% of test trials).

Planning how to reach the subgoal The third step of the MGPO strategy is to plan the path to the subgoal by inspecting immediate nodes after identifying a promising goal and intermediate nodes. The percentage of participants who, like the MGPO strategy, only selected an immediate node after already selecting at least one goal node and intermediate node was 69% for the *Choice tutor* condition, 41% for the *Dummy tutor* condition, and 17% for the *No tutor* condition. In both the *Dummy tutor* condition (31% of test trials) and the *No tutor* condition (73% of test trials), the most common mistake was to plan within immediate nodes before planning which goal to pursue. In the *Choice tutor* condition, the most common mistake was failing to select an intermediate node before inspecting immediate nodes (16% of test trials).

Participants in the two control conditions made costly mistakes in their inspections of interme-

diate and immediate nodes because expending resources to plan a path to a potentially bad goal is highly inefficient. In comparison, the mistakes made by participants in the *Choice tutor* condition were arguably less costly since the participant had already identified a worthwhile goal, and any nodes on the path to that goal have the potential of increasing the participant's *RR-score*.

3.7 Conclusion

In this chapter, I developed the first cognitive tutor for discovering and teaching resource-rational planning strategies for partially observable environments, and showed that the tutor improved people's resource-rationality by teaching them the planning strategy discovered by MGPO. People trained by the cognitive tutor learned significantly better planning strategies compared to people who practiced by themselves or were trained with a similar tutor that had no knowledge of MGPO's strategy. The technical contributions towards this achievement were (1) extending the meta-level MDP model of optimal human planning strategies to partially observable environments, (2) developing MGPO, a metareasoning algorithm that derives near-optimal strategies for human planning in partially observable environments, and (3) creating and evaluating a novel cognitive tutor that teaches the discovered planning strategies by approximating optimal metacognitive feedback.

To achieve a high computational efficiency, I limited MGPO's meta-planning to a single step. This myopic way of meta-planning performed well in my tests and allowed it to compute metacognitive feedback in an online fashion that adapts directly to the learner's current belief state. However, to discover a truly optimal planning strategy, it is necessary to consider all possible sequences of future computations and the observations they might generate. More practically, the method could plan for a fixed number of computations or approximate VOC features that contain groups of nodes (Callaway et al., 2018a).

In summary, this chapter presents a cognitive tutor that leverages human-centered AI to improve

human planning and decision-making in partially observable environments and demonstrated its effectiveness through simulations and a training experiment. This constitutes an important step towards improving human decision-making in complex real-world tasks requiring farsighted planning under uncertainty. The cognitive tutor makes it possible to teach highly complicated planning strategies to people, and MGPO can discover strategies that are substantially better than the strategies people intuitively use. It can, therefore, be used to characterize the principles of good decision-making under uncertainty that we could previously only speculate about. MGPO is also computationally efficient enough to be used by the cognitive tutor for interactive tutoring, making it possible to provide metacognitive feedback based on a learner's current information about the planning task.

4

Project selection

This chapter introduces the project selection problem and MGPS, a strategy discovery method adapted to project selection, and demonstrates its effectiveness through a human training experiment. The Chapter is based on my preprint "Leveraging automatic strategy discovery to teach people how to select better projects" Heindrich and Lieder (2024), on which I collaborated with my advisor Falk Lieder.

4.1 Introduction

In this chapter, I apply strategy discovery to a real-world decision problem: project selection. Corporations and individuals commonly have to select a project out of multiple alternatives. These project selection problems are usually high-stakes decisions that can be highly impactful for the future of an organization. For example, an organization looking for a sustainable investment project (Khalili-Damghani and Sadi-Nezhad, 2013) can benefit both financially and by improving its public image by selecting an impactful and profitable project, or incur major losses by selecting an unsuitable project.

Previous research on project selection recommends that candidate projects should be evaluated by multiple experts (Coldrick et al., 2002, Khalili-Damghani and Sadi-Nezhad, 2013, Liu et al., 2017), and many structured approaches to integrate the experts' opinions exist (de Souza et al., 2021). However, existing project selection techniques are not well utilized in the real world (Schmidt and Freeland, 1992, Liu et al., 2017, Henriksen and Traynor, 1999). I hypothesize this underutilization is partly caused by the complexity and implementation costs of normative project selection strategies. Scoring approaches, which score alternative projects on a number of relevant selection criteria, have been proposed as a less complex selection method that can be applied to real-world problems more easily (Henriksen and Traynor, 1999, Khalili-Damghani and Sadi-Nezhad, 2013, Coldrick et al., 2002, Sadi-Nezhad, 2017). However, even scoring approaches face significant challenges when applied to real-world problems: precisely evaluating projects on multiple criteria can be a time-consuming and expensive process, but existing selection techniques often pay little attention to the information-gathering costs and instead assume expert opinions on all criteria are readily available and instead focus on how to integrate existing expert evaluations into a final project choice (Abdel-Basset et al., 2019, Khalili-Damghani and Sadi-Nezhad, 2013).

Due to the limited utilization of complex project selection methods, decision-makers, therefore,

often rely on their intuition and much simpler techniques like brainstorming (Kornfeld and Kara, 2013). This is concerning because the intuitive decisions of groups and individuals are highly susceptible to biases and unsystematic errors (Kahneman et al., 1982), such as weighing one's own opinion too strongly (Yaniv and Kleinberger, 2000), being overconfident, and weighting the recommendations of advisors by their confidence rather than their competence (Bonaccio and Dalal, 2006). However, teaching people prescriptive decision strategies discovered by strategy discovery methods that take cognitive limitations into account offers an opportunity to prevent people's errors in decision-making (Lieder and Griffiths, 2020a, Callaway et al., 2022a).

In this chapter, I present a novel decision support method with the goal of discovering prescriptive decision-strategies that can be taught to people to improve how they select projects. Project selection is challenging because many crucial attributes of the candidate projects, such as their expected profitability, cannot be observed directly. Instead, they have to be inferred from observations that are not fully reliable. Therefore, I formalize project selection strategies as policies for solving a partially observable meta-MDP.

To achieve this, I model a realistic project selection task as a metareasoning problem. The metareasoning consists in deciding which information one should request from which advisors when information is highly limited, uncertain, and costly. I develop an efficient algorithm for solving this problem based on MGPO (see Section 3.3), and apply it to derive a prescriptive decision strategy for a project selection problem a financial institution faced in the real world (Khalili-Damghani and Sadi-Nezhad, 2013).

Finally, I develop a cognitive intelligent tutor (Callaway et al., 2022a) that teaches the decision strategy discovered by my method to people. I evaluated the approach by letting the cognitive tutor teach the automatically discovered project selection strategy to about 100 people, and then evaluated the quality of their decisions in realistic project selection problems against two control groups. My results indicate that the approach can successfully improve human decisions in real-world problems

where people are reluctant to let machines decide for them.

4.2 Background

Project selection In the project selection problem, a decision-maker aims to select the best-fitting project out of several candidates (Sadi-Nezhad, 2017). Evaluation of candidate projects is usually based on a number of criteria relevant to the decision-maker, such as financial profits, risks, environmental impacts, or the alignment with organizational goals (Carazo et al., 2012, de Souza et al., 2021). This problem can be formalized as multi-criteria decision-making (MCDM) (de Souza et al., 2021, Mohagheghi et al., 2019). Projects can be evaluated using a scoring technique, which evaluates relevant criteria and then combines them to a weighted sum (Sadi-Nezhad, 2017). Common approaches to solving the project selection problem include techniques such as the analytic hierarchy process, the analytic network process, real options analysis, and TOPSIS (see de Souza et al., 2021 for a review). These methods are commonly combined with fuzzy sets to account for uncertainty (Khalili-Damghani and Sadi-Nezhad, 2013). However, these methods are rarely used in real-world problems because implementing them would require gathering and integrating a lot of information through a time-consuming process, which is often incompatible with the organizational decision process (Schmidt and Freeland, 1992, Liu et al., 2017). Instead, organizations often rely on simpler, less structured methods like brainstorming (Kornfeld and Kara, 2013). In addition, while the detailed information required by these methods can be costly and time-consuming to acquire in real-world settings, the methods often don't take into account how much information is needed to make an efficient decision.

Judge-advisor systems In a Judge Advisor System (JAS) (Bonaccio and Dalal, 2006), typically, a single decision-maker has to make a decision, and multiple advisors support the decision-maker by offering advice. Variations of the task can include costly advice (Yaniv and Kleinberger,

2000, Gino, 2008), or advisors with varying reliability (Olsen et al., 2019). This is a common situation when CEOs decide which project their company should launch next. Unfortunately, decision-makers are known to be highly susceptible to systematic errors, such as weighing one's own opinion too strongly, overconfidence, egocentric advice discounting, and weighting the recommendations of advisors by their confidence rather than their competence (Bonaccio and Dalal, 2006, Ronayne et al., 2019, Yaniv and Kleinberger, 2000). I model the project selection problem within the JAS framework by letting project evaluators take the role of advisors with varying reliability.

4.3 Formalizing optimal decision strategies for human project selection as the solution to a meta-level MDP

In this section, I introduce my general resource-rational model of project selection, which I expect to be widely applicable to concrete, real-world project selection problems.

My model of project selection consists of two decision problems, an object-level decision-problem and a meta-level MDP (Callaway et al., 2018a, Hay et al., 2014). The two decision problems separate the actions the decision-maker has to choose between (object level), such as executing one project versus another, from decision operations that represent thinking about which of those object-level actions to perform (meta-level), such as gathering information about the projects' attributes. This allows to solve both problems separately. The object-level decision problem is a MCDM problem, where a set of N_P potential projects $\mathcal{P} = \{p_1, ...p_{N_P}\}$ are evaluated using N_C relevant criteria $C = [c_1, ...c_{N_C}]$ weighted by fixed predetermined weights $W = [w_1, ...w_{N_C}]$. Actions in the object-level problem represent selecting the corresponding project ($\mathcal{A} = \{a_1, ..., a_{N_P}\}$). The reward of a selecting a project is computed by summing the weighted criteria scores of the selected project: $r^O(a_i) = \sum_c w_c c_{c,i}$ (Coldrick et al., 2002).

While the object-level decision problem is my model of the project selection task, the meta-level

MDP is my formalization of the problem of discovering resource-rational project selection strategies. It models the task of gathering information about deciding which project to select. States in the meta-level MDP are belief states that represent the current information about each project's attributes. I model belief states using a multivariate Normal distribution to quantify the estimated value and uncertainty about the N_P projects' scores on the N_C criteria:

$$b = \left[\left(\mu_{1,1}, \sigma_{1,1} \right), \cdots, \left(\mu_{N_P,N_C}, \sigma_{N_P,N_C} \right) \right]. \tag{4.1}$$

The actions (decision operations) of the meta-level MDP gather information about the different attributes of projects by asking one of the N_E experts for their estimate of how one of the projects scores on one of the criteria. Experts provide discrete estimates from min_{obs} to max_{obs}, and expert estimates can differ in their reliability and their cost. Specifically, the available actions are $A^M = \{a_{1,1,1}, \cdots, a_{N_P,N_C,N_E}, \perp\}$, where the meta-level action $a_{i,j,k}$ represents asking the expert e_k for their estimate of criterion c_j of project p_i. After receiving information obs from an expert, the current belief state $b_{p_i c_j} = \mathcal{N}(\mu, \sigma)$ is updated by applying the update equation for a Gaussian likelihood function with standard deviation σ_e (i.e. the expert's reliability) and a conjugate Gaussian prior (i.e., the current belief), that is:

$$\hat{\mu} \leftarrow \left(\frac{w_{c_i} \cdot \mu}{(w_{c_i} \cdot \sigma)^2} + \frac{w_{c_i} \cdot obs}{(w_{c_i} \cdot \sigma_e)^2} \right) \cdot \left((w_{c_i} \cdot \sigma)^2 + (w_{c_i} \cdot \sigma_e)^2 \right) \tag{4.2}$$

and

$$\hat{\sigma} \leftarrow \sqrt{\frac{1}{\frac{1}{(\sigma \cdot s)^2} + \frac{1}{(\sigma_e \cdot s)^2}}}. \tag{4.3}$$

The reward of these meta-level actions is the negative cost $r^M(a_{i,j,k}) = -\lambda_{e_k}$ of asking the expert e_k for help. Finally, the meta-level action $\perp$ is the termination action, which corresponds to terminating the decision-making process and selecting a project. The reward of the termination action is

the expected reward of selecting the best project according to the current belief-state. An optional budget parameter N_a specifies the maximum number of available meta-level actions, after which the termination action is performed automatically.

Meta-level MDPs are notoriously difficult to solve due to their extremely large state space (Hay et al., 2014). In the project selection task, the meta-level MDP has $(max_{obs} - min_{obs} + 2)^{N_P \cdot N_C \cdot N_E}$ possible belief states and $N_P \cdot N_C \cdot N_E + 1$ possible meta-level actions. The project selection meta-MDP introduces multiple new intricacies that existing metareasoning methods and MGPO aren't equipped to handle, making strategy discovery in this setting especially difficult. Compared to previously formulated meta-level MDPs (Callaway et al., 2018a, Hay et al., 2014, Mehta et al., 2022), the meta-level description of project selection differs in the following ways: (1) the maximum amount of meta-level actions is constrained with a budget, (2) the project selection task features multiple consultants who differ in both the quality of their advice and the cost of their services, (3) consultants in the project selection task offer discrete estimates of each criterion, requiring that (4) criteria ratings are scaled to allow weighting the criteria according to their importance.

4.4 A new metareasoning algorithm for discovering decision strategies for human project selection

Previous metareasoning methods are unable to handle some of the intricacies of the project selection problem. Therefore, I developed a new strategy discovery method based on MGPO (see Section 3.3), which overcomes the limitations that prevent MGPO from being applicable to project selection. To reflect the commonalities and innovations, I call the new strategy discovery method MGPS (meta-greedy policy for project selection). Similar to MGPO, the method approximates the value of computation (VOC) (Russell and Wefald, 1991a) by myopically estimating the immediate improvement in decision quality. Improving upon MGPO, MGPS calculates the myopic approxi-

mation to the VOC in a way that accounts for discrete criteria ratings, criteria scaling, and multiple sources of information with different costs and reliabilities.

Algorithm 2 MGPS VOC calculation for an action a_{p_i,c_i,e_i}

1: **function** MYOPIC_VOC(p_i, c_i, e_i, b)
2: $r_p \leftarrow \mathbb{E}[r^O(p_i)]$
3: $r_{alt} \leftarrow \max_{p_j \in \mathcal{P}-\{p_i\}} \mathbb{E}[r^O(p_j)]$
4: $\mu, \sigma \leftarrow b_{p_i c_i}$
5: **for** *obs* **from** min_{obs} **to** max_{obs} **do**
6: **if** $min_{obs} < obs < max_{obs}$ **then**
7: $p_{obs} \leftarrow \Phi\left(\frac{w_{c_i}\cdot(obs+0.5-\mu)}{\sqrt{(w_{c_i}\cdot\sigma)^2+(w_{c_i}\cdot\sigma_e)^2}}\right) - \Phi\left(\frac{w_{c_i}\cdot(obs-0.5-\mu)}{\sqrt{(w_{c_i}\cdot\sigma)^2+(w_{c_i}\cdot\sigma_e)^2}}\right)$
8: **else if** $obs == min_{obs}$ **then**
9: $p_{obs} \leftarrow \Phi\left(\frac{w_{c_i}\cdot(obs+0.5-\mu)}{\sqrt{(w_{c_i}\cdot\sigma)^2+(w_{c_i}\cdot\sigma_e)^2}}\right)$
10: **else**
11: $p_{obs} \leftarrow 1 - \Phi\left(\frac{w_{c_i}\cdot(obs-0.5-\mu)}{\sqrt{(w_{c_i}\cdot\sigma)^2+(w_{c_i}\cdot\sigma_e)^2}}\right)$
12: **end if**
13: $\hat{\mu}_{obs} \leftarrow \left(\frac{w_{c_i}\cdot\mu}{(w_{c_i}\cdot\sigma)^2} + \frac{w_{c_i}\cdot obs}{(w_{c_i}\cdot\sigma_e)^2}\right) \cdot ((w_{c_i}\cdot\sigma)^2 + (w_{c_i}\cdot\sigma_e)^2)$
14: **end for**
15: **if** $r_p > r_{alt}$ **then**
16: voc $\leftarrow \sum_{obs=min_{obs}}^{max_{obs}} p_{obs}(r_{p_{alt}} + \mu - r_p - \hat{\mu}_{obs}) \cdot 1(r_p - \mu + \hat{\mu}_{obs} < r_{alt})$
17: **else**
18: voc $\leftarrow \sum_{obs=min_{obs}}^{max_{obs}} p_{obs}(r_p + \hat{\mu}_{obs} - \mu - r_{p_{alt}}) \cdot 1(r_p - \mu + \hat{\mu}_{obs} > r_{alt})$
19: **end if**
20: **return** $(1 - w_\lambda)$voc - $w_\lambda \lambda_{e_i}$
21: **end function**

MGPS calculates a myopic approximation to the VOC of asking an expert about a specific criterion of a single project according to Algorithm 2. To account for discrete advisor outputs, Algorithm 2 iterates over the discrete set of possible ratings the expert might give and estimates the probability p_{obs} of each rating (*obs*) and the belief update that would result from it $\hat{\mu}_{obs}$. The probability of each rating is computed using the cumulative distribution function (Φ) of the belief state for the selected project's criterion score (see Line 7). Here, the standard deviation σ_e of the likelihood

function encodes the expert's reliability, the prior ($\mathcal{N}(\mu, \sigma)$) is the current belief about the project's score on the evaluated criterion, and the weights w_{c_i} convert the criteria into a common currency. The belief update that would result from the observation ($\hat{\mu}_{obs}$) is computed by applying the belief state update (see Line 13 and Equation 4.2). For the highest and lowest possible ratings, I instead calculate p_{obs} using the open interval (see Lines 9 and 11). The updated expected value of the belief state according to an observation *obs* is then calculated using Bayesian inference to integrate the new observation into the belief state (see Line 13).

The VOC calculation depends on the posterior belief states that would result from the different possible observations ($\hat{\mu}_{obs}$), weighted by their probabilities. If the evaluated project currently has the highest expected reward (see Line 15), the VOC calculation expresses the probability of observing a value low enough that the second-best project becomes the most promising option (see Line 16). In the case where the evaluated project did not have the highest expected termination reward, the VOC calculation expresses the probability of observing a value high enough to make the evaluated project the most promising option (see Line 18). Finally, the cost of the requested expert λ_{e_i} is weighted using a free cost weight parameter w_λ and subtracted from the VOC estimate (see Line 20).

The full meta-greedy policy consists of calculating the VOC for all possible meta-level actions and iteratively selecting the meta-level action with the highest VOC. If no action has a positive VOC, the termination action $\perp$ is chosen.

4.5 Improving human project selection

Having developed a general metareasoning method for discovering resource-rational strategies for human project selection, I now extend it to an intelligent cognitive tutor for teaching people how to select better projects. My goal is to evaluate whether humans can utilize the strategies discovered

by MGPS and to provide a proof of concept for a general AI-powered boosting approach that can be used to improve human decision-making across a wide range of project selection problems. I first introduce a general approach for teaching people the project selection strategies discovered by MGPS, and then apply it to a real-world project selection problem. The results demonstrate that people who are taught the strategy discovered by MGPS perform significantly better.

4.5.1 MGPS Tutor: A cognitive tutor for teaching people how to select better projects

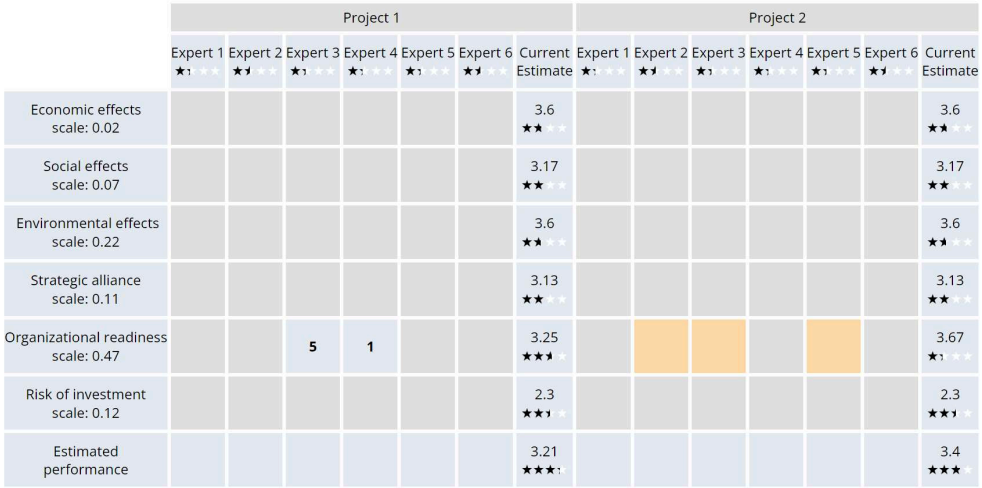


Figure 4.1: Example of the MGPS tutor offering a choice between requesting information from three different experts (highlighted in orange) in the simplified training task of deciding between two project alternatives.

The cognitive tutor for project selection (*MGPS Tutor*) trains people to select the near-optimal decision operations identified by MGPS. To achieve this, it lets people practice on a series of project selection problems and gives them feedback. MPGS Tutor leverages MPGS in two ways: i) to pedagogically construct the set of queries the learner is asked to choose from, and ii) to give the learner feedback on their chosen query.

Extending the choice tutor presented in Section 3.5, the tutor repeatedly asks the learner to choose from a pedagogically chosen set of decision operations (see Figure 4.1) that always includes the query that MGPS would have performed. Moreover, it leverages MGPS's VOC calculation (Algorithm 2) to score the chosen query, and then provides binary feedback on its quality. If learners select a suboptimal expert, project, or criterion, they receive feedback indicating the correct choice and have to wait for a short time. The unpleasantness of having to wait serves as a penalty (Callaway et al., 2022a). Otherwise, they receive positive reinforcement and the next choice is displayed. To receive positive reinforcement, the learner must select a query whose VOC is sufficiently close, as determined by a tolerance parameter t, to the VOC of the optimal action. I manually set the tolerance to $t = 0.001$ to strike a balance between teaching learner's the strategy as accurately as possible while avoiding annoying and confusing feedback for extremely minor mistakes.

The tutor teaches the strategy discovered by MGPS using an extended learning schedule building upon the tutor presented in Section 3.5.2, which fosters learning by incrementally increasing the complexity of the training task based on the shaping methodology (Skinner, 1953). The training schedule varies the numbers of projects, how many different expert assessments learners have to choose between, and the specific types of expert assessments offered as choices. In total, my tutor teaches the discovered project selection strategy using ten training trials. The first seven training trials use a smaller version of the project selection task with only two projects, while the last three trials use the full environment with five projects. The number of choices gradually increases throughout training, from 1 in the first training trial to 9 in the last three training trials. The tutor also varies the types of choices across trials. After an initial trial with only a single choice, the tutor offers choices that focus on different criteria within the same project for two trials. Then, the tutor offers choices that focus on different experts within the same project for two trials. The remaining trials combine both types of highlights while sometimes also featuring queries about different projects and also increasing the overall number of choices. While the tutor primarily offers choices within the

same project, later trials with higher a number of choices sometimes feature queries about different projects. The optimal choice is always included in the choice set.

4.5.2 Evaluating the effectiveness of MGPS Tutor in a training experiment

To evaluate if AI-powered boosting can improve human project selection, I evaluated the MPGS tutor in a training experiment. I tested if people trained by MPGS tutor learn more resource-rational project selection strategies. To make the assessment task as naturalistic as possible, I modelled it on a real project selection problem that was faced by an Iranian financial institution (Khalili-Damghani and Sadi-Nezhad, 2013). I will first describe how I modeled this real-world problem, and then the training experiment.

A project-selection problem from the real world Khalili-Damghani and Sadi-Nezhad (2013) worked on the real-world problem of helping a financial institution select between five potential projects with an eye to sustainability. Each project was evaluated by six advisors, who assigned scores from one to five on six different criteria. For my model of the task, I use the same number of experts, criteria, and projects, and the same criteria weights as the financial institution. The remaining parameters of the meta-level MDP were estimated as follows. I initialized the beliefs about the project's attributes by calculating the mean and the standard deviation of all expert ratings for each criterion according to (Khalili-Damghani and Sadi-Nezhad, 2013). I estimated the reliability of each expert by calculating the standard deviation from the average distance of their ratings from the average rating of all other experts, weighted by the number of occurrences of each guess. I estimated the cost parameter λ of the meta-level MDP by $\lambda = \frac{cost}{stakes} \cdot r(\bot)$ to align the meta-level MDP's cost-reward ratio to its real-world equivalent. Using the expected termination reward of the environment $r(\bot) = 3.4$ and rough estimates for the stakes $stakes = \$10000000$ and expert costs $cost = \$5000$, this led to $\lambda = 0.002$. While Khalili-Damghani and Sadi-Nezhad (2013) assumed

all expert ratings are available for free, in the real world this is rarely the case. To make the test case more representative, I assumed that advisor evaluations are available on-request for a consulting fee. To capture that real-world decisions often have deadlines that limit how much information can be gathered, I set the maximum number of sequentially requested expert consultations to 5.

Methods of the experiment I recruited 301 participants for an online training experiment on Prolific. The average participant age was 29 years, and 148 participants identified as female. Participants were paid £3.50 for completing the experiment, plus an average bonus of £0.50. The median duration of the experiment was 22 minutes, resulting in a median pay of £10.9 per hour. The experiment was preregistered on AsPredicted and approved by the ethics commission of the Medical Faculty of the University of Tübingen under IRB protocol number 667/2018BO2.

Each participant was randomly assigned to one of three conditions: (1) the *No tutor* condition, in which participants did not receive any feedback and were free to discover efficient strategies on their own; (2) the *MGPS tutor* condition, in which participants practiced with the cognitive tutor that provided feedback on the resource-rationality score MGPS assigns to the selected planning actions; and (3) the *Dummy tutor* condition, an additional control condition in which participants practiced with a dummy tutor comparable to the MGPS tutor, albeit with randomized feedback on which planning actions are correct. All participants practiced their planning strategy in 10 training trials and were then evaluated across 10 test trials. For each trial, a new instance of the project selection environment was randomly generated by first sampling the project's criteria scores from the initial belief state and then generating each expert's ratings based on their reliability σ_e using a Gaussian distribution centered at the criteria score.

We evaluated the participants' performance using two measures: their *RR-score* and *click agreement* as introduced in previous chapters. *RR-score's* are normalized by subtracting the average reward of a random baseline algorithm and dividing by the participant scores' standard deviation.

Table 4.1: Results of the human training experiment. Per condition, the normalized mean resource-rationality score and the mean click agreement are reported. For both measures, I also report the 95% confidence interval under the Gaussian assumption (± 1.96 standard errors).

Condition	RR-score	Click Agreement
MGPS Tutor	0.3256 ± 0.0609	0.4271 ± 0.0201
No tutor	-0.0227 ± 0.0622	0.2521 ± 0.0171
Dummy tutor	0.0225 ± 0.0612	0.2664 ± 0.0159

The random baseline algorithm is defined as the policy that chooses meta-level actions at random until the maximum number of decision operations is reached. *Click agreement* measures, how well participants learned to follow the near-optimal strategy discovered by my method. Specifically, I computed for each participant, which proportion of their information requests matched the action taken by the approximately resource-rational strategy discovered by MGPS.

Experiment results Table 4.1 shows the results of the experiment. To determine whether the condition of participants had a significant effect on the RR-score and click agreement, I used an ANOVA analysis with Box approximation (Box, 1954). The ANOVA revealed a significant effect of condition on both RR-score ($F(1.99, 293.57) = 10.48, p < .0001$) and click agreement ($F(1.99, 291.48) = 15.5, p < .0001$). I further compared the performance of participants in the *MGPS tutor* condition to participants in the two control conditions with post hoc ANOVA-type statistics and used Cohen's d (Cohen, 2013) to evaluate the size of the effects. The post hoc tests revealed that participants in the *MGPS tutor* condition achieved a significantly higher RR-score than participants in the *No tutor* ($F(1) = 17.88, p < .0001, d = .35$) and *Dummy tutor* ($F(1) = 13.4$, $p = .0002, d = .31$) conditions. Additionally, participants in the *MGPS tutor* reached a higher click agreement with the pre-computed near-optimal strategy than participants in the *No tutor* ($F(1) = 25.08, p < .0001, d = .58$) and *Dummy tutor* ($F(1) = 19.3, p < .0001, d = .56$) conditions.

When evaluated on the same test trials and normalizing against the baseline reward and the standard deviation of the experiment results, MGPS achieves a mean reward of 1.17, demonstrating that MGPS discovers more resource-rational strategies than participants across all conditions. Although participants in the *MGPS tutor* condition performed significantly the better than participants in the other conditions, they did not learn to follow the strategy taught by the tutor exactly. Participants in the other conditions only discovered strategies with a similar *RR-score* to the random baseline strategy, with participants in the *No tutor* condition performing even worse than the random baseline strategy, and participants in the *Dummy tutor* condition outperforming the random baseline only by a small margin.

What strategy did the MGPS Tutor teach? To understand what participants in the MGPS Tutor condition were taught, I inspected the strategy discovered by MGPS. I found that this strategy systematically asks the most reliable experts (e_2 and e_6) to evaluate projects on the criterion with the highest decision weight (c_5). It starts by asking expert e_2 to rate criterion c_5 for a randomly selected project. If that rating is below the maximum score, the same expert is immediately asked to rate the same criterion for a different project. If expert e_2 gives project p_i the highest possible score on criterion c_5 ($max_{obs} = 5$), the strategy requests a second opinion about it from expert e_6. If expert e_6 also evaluates the criterion's value as 5, the decision process ends and project p_i is selected. If expert e_6 provides a rating below 5, expert e_2 is asked about a different project. This process is repeated until a project in which both experts e_2 and e_6 estimate $c_5 = 5$ is found, or the maximum number of decision operations (5) is reached. It is important to note that while this decision strategy is relatively easy to understand, describe, and execute, the task of discovering this specific strategy as more resource-rational than the vast number of possible other strategies is considerably more difficult.

What aspects of the strategy did participants learn? I further analyzed which aspects of MGPS' strategy participants learned from the cognitive tutor. Specifically, I investigated in which proportion of the test trials participants managed to (1) start planning with an optimal action and (2) correctly decide whether to continue investigating the same project or switch to requesting information about an alternative project.

MGPS' strategy starts by requesting information about criterion c_5 from one of the two most reliable experts. Participants in the *MGPS tutor* condition did so 56% of the time, whereas participants in the *No tutor* condition did so only 33% of the time. Looking into why many failed to request this information, I found that more participants learned to request information from the most reliable experts (73% for participants in the *MGPS tutor* condition and 43% for participants in the *No tutor* condition) than to request information about the most important evaluation criteria (64% for participants in the *MGPS tutor* condition and 43% for participants in the *No tutor* condition).

A second important aspect of MGPS's strategy is how it responds to the information revealed. Excluding trials in which participants' first expert request did not match an action MGPS identifies as optimal, I analyzed whether participants either correctly continued to evaluate the current project (in case of revealing an expert guess of 5), or correctly switched to evaluating an alternative project (when revealing an expert guess smaller than 5). Participants in the *MGPS tutor* condition (65% of test trials) and participants in the *No tutor* condition (65% of test trials) both learned to accurately switch projects when appropriate. Repeatedly evaluating the current project proved to be the most difficult component of the discovered strategy to learn, with participants in the only *MGPS tutor* condition following this aspect of the strategy in 11% of test trials and participants of the *No tutor* condition following the strategy in only 7% of test trials.

4.6 Performance evaluation

The results reported in the previous section show that MPGS can discover project selection strategies that are more effective than the strategies people discover on their own. But how does its performance compare to other strategy discovery algorithms? To answer this question, I evaluated MGPS on a computational benchmark. I again chose PO-UCT (Silver and Veness, 2010) for comparisons. PO-UCT utilizes Monte Carlo tree search to simulate the effects of different actions, resulting in more accurate results with increased computation time, making it a useful baseline for MGPS's computational efficiency and performance.

Method I evaluated the effectiveness of my method in the project selection task by comparing it against PO-UCT (Silver and Veness, 2010) with different numbers of steps. All methods were evaluated across the same 5000 randomly generated instances of the project selection environment.

The main performance measure was the resource-rationality score (*RR-Score* defined in Equation 1.1). To highlight the achieved improvements over a baseline algorithm that performs random meta-level actions, I normalized the reported *RR-scores*. Specifically, I applied a z-score transformation, subtracting the average reward of the random baseline algorithm (see Section 4.5.2) from the *RR-Scores* and dividing by the evaluated algorithm's *RR-Scores*' standard deviation. I analyze the differences in *RR-Scores* with an ANOVA and evaluate the size of statistical effects with Cohen's d (Cohen, 2013). Additionally, I compare the computational efficiency of the different methods, which is crucial for being able to provide real-time feedback in the cognitive tutor.

Results As shown in Table 4.2, MGPS outperformed all tested versions of PO-UCT and the random baseline strategy (as the *RR-scores* are normalized by subtracting the mean *RR-score* of the random baseline, the random baseline strategy itself has a normalized *RR-score* of 0). An ANOVA revealed significant differences in the *RR-scores* of the strategies discovered by the different methods

Table 4.2: Results of the performance evaluation. For each algorithm, I report the averge normalized resource-rationality score (*RR-Scores*) and the runtime per decision problem. For both measures, I also report the 95% confidence interval under the Gaussian assumption (± 1.96 standard errors).

Algorithm	RR-score	Runtime (s)
MGPS	0.9942 ± 0.0234	0.9079 ± 0.0052
PO-UCT (10 steps)	-0.4344 ± 0.0106	0.0175 ± 0.0004
PO-UCT (100 steps)	0.7309 ± 0.0302	0.1972 ± 0.0008
PO-UCT (1000 steps)	0.8681 ± 0.0256	2.3567 ± 0.0028
PO-UCT (5000 steps)	0.9054 ± 0.0232	10.8913 ± 0.0173

($F(4, 24995) = 2447, p < .0001$). Hukey-HSD post-hoc comparisons indicated that the strategies discovered by MGPS are significantly more resource-rational than the strategies discovered by PO-UCT with 10 steps ($p < .0001, d = 2.18$), 100 steps ($p < .0001, d = .27$), 1000 steps ($p < .0001$, $d = .14$), or 5000 steps ($p < .0001, d = .11$).

While MGPS achieves significantly higher *RR-scores* than all PO-UCT variants, the size of the effect decreases from a very large effect to a small effect when increasing PO-UCT's computational budget sufficiently. I therefore expect that PO-UCT with a much more than 5000 steps would ultimately achieve comparable *RR-scores* to MGPO, albeit at a much higher computational cost. Moreover, MGPS was substantially faster than PO-UCT with a computational budget of 1000 steps or more, which is when PO-UCT's performance starts to approach that of MGPS. With a computational budget of 100 steps or fewer, PO-UCT is faster than MGPS. However, such a small computational budget is not enough for PO-UCT to discover strategies with a *RR-score* anywhere near that of the strategy discovered by MGPS. Even when giving PO-UCT a computational budget 10 times higher than MGPS requires, PO-UCT does not achieve similar *RR-scores* to my method. Critically, the high amount of computation required for PO-UCT to achieve an approximately similar level of resource-rationality would render PO-UCT unusable for a cognitive tutor that computes feedback in real time.

4.7 Conclusion

People's decision-making is prone to systematic errors (Kahneman et al., 1982), and although people are happy to delegate some decisions, most CEOs are unlikely to let AI decide which projects their company should pursue. Moreover, people are reluctant to use normative project selection procedures because they tend to be more tedious (Schmidt and Freeland, 1992, Liu et al., 2017, Kornfeld and Kara, 2013). Motivated by people's insistence on making their own decisions with simple heuristics, I leveraged AI to discover and teach prescriptive decision strategies adapted to human cognition that perform substantially better than people's intuitive strategies but are nevertheless simple enough that people use them. To this end, I introduced a metareasoning method for leveraging AI to discover near-optimal decision strategies for human project selection. Modeling project selection through the lens of resource-rationality allowed me to formulate a mathematically precise criterion for the quality of decision strategies for human project selection. I further developed MGPS, an efficient automatic strategy discovery algorithm that automatically discovers efficient strategies for human project selection. MGPS discovered decision strategies that are much more resource-rational than the strategies humans discovered on their own and the strategies discovered by a general-purpose algorithm for solving POMDPs (PO-UCT). Using the decision strategies discovered by MGPS, I created a cognitive tutor that uses a shaping schedule and metacognitive feedback to teach the strategies to humans. In the training experiment, the cognitive tutor fostered significant improvements in participants' resource-rationality.

My results indicate that it is possible to use resource-rational analysis combined with automatic strategy discovery to improve human decision-making in a realistic project selection problem. As selecting projects is a common problem faced by both organizations and individuals, improving their decision strategies in this setting would have a direct positive impact. Specialized project selection tutors for specific types of project selection problems could be used to train current and future lead-

ers within companies and organizations. Moreover, project-selection tutors could be integrated into MBA programs to teach future decision-makers efficient decision strategies as part of their education. I am optimistic that the general methodology is also applicable to other real-world problems, offering a promising pathway to teach people efficient strategies for making better decisions in other areas as well. Besides project selection problems, I believe this approach could be used to improve real-world decision-making in areas such as grant-making and policy decisions.

5

Further real-world applications

In this chapter, I provide a preliminary model of a second real-world application for strategy discovery methods: social dilemmas. Specifically, I present a metareasoning model of decision-making in social dilemmas, but leave developing strategy discovery methods and cognitive tutors for this domain to future work.

5.1 Introduction

Social dilemmas are situations in which the interests of an individual are at odds with the collective interests of the group (Dawes, 1980). A classic example of a social dilemma is the tragedy of the commons (Hardin, 2018), in which each individual is incentivized to maximize their resource consumption, such as catching and selling as much fish as possible, but collectively the shared resource of fish in the ocean will be depleted if everyone acts unsustainably. Social dilemmas are commonly studied in game theoretic models, for example the prisoner's dilemma or the chicken dilemma (Van Lange et al., 2013), which are usually characterized by the fact that rational actions on the individual level don't lead to the best overall outcome. Recent work by Callaway et al. (2023) found that human decisions in such games can be explained by resource-rational heuristics, which trade off between expected utility and cognitive costs based on past experiences. This evidence makes me optimistic about the possibilities of applying a metareasoning model to analyze and improve human decisions in real-world social dilemmas.

In social dilemmas, cooperation leads to clearly preferred outcomes for the common good. Researchers have long tried to find ways to incentivize cooperation by adapting the structure of the choice environment or finding new game theoretic solutions (Kollock, 1998). While individuals in isolated choice situations seem to have little incentive for cooperation, motivational solutions to social dilemmas build on the fact that most people are not solely concerned by their own outcome but also assign a non-zero value to the outcomes achieved by other people (Kollock, 1998). For example, a recent online survey found that altruism and reciprocity are important motivators for people to cooperate in various real-world social dilemmas, such as whether to donate to public radio (Attari et al., 2014). Another potential driver of cooperation is one's reputation. Wang et al. (2012) investigate the effects of reputation in a Prisoner's dilemma setting by allowing simulated players to choose opponents based on their reputation, and find that the mechanism leads to increased cooperation.

In recent years, AI has been applied to discover new approaches to solving social dilemmas. Koster et al. (2022) used reinforcement learning to discover structural solutions to distributing public goods and found that the AI solution was both preferred by humans and lead to fairer outcomes. In subsequent work, a similar approach has been applied to an iterated social dilemma, where the reinforcement learning method discovered sustainable distribution strategies that promoted the collective good, demonstrating the potential of applying strategy discovery to social dilemmas (Koster et al., 2024).

Even if people are motivated to achieve an outcome that is not only good for themselves but also for other participants, their decisions in social dilemmas can still be negatively affected by human biases such as failing to consider the long-term consequences of their actions for themselves, as well as the consequences for other involved parties. Therefore, teaching people cognitive strategies for social dilemma situations that evaluate both long-term and short-term consequences, as well as evaluating how the outcome will affect not only themselves, but also other involved people, has the potential of allowing people to make decisions for the greater good.

In the remainder of this chapter, I introduce the metareasoning model of social dilemmas. The model focuses on two player interactions where each player chooses between cooperation and defection but includes an extended setting for more than two actions as well. Actions are evaluated by not only their short-term utility, but also their long-term utility, as well as the outcomes for other players or affected parties. Metareasoning actions include considering potential outcomes and their effects on every parties short and long-term utilities.

5.2 Object-level problem

The object-level problem can be framed as an MDP $(\mathcal{S}, \mathcal{A}, P, r)$. Possible states are $\mathcal{S} = \{s_0, s_{cc}, s_{cd}, s_{dc}, s_{dd}\}$, the starting state s_0 and four terminal states that represent possible outcomes of a two-player game

in which players AP_1 and AP_2 choose between the option to cooperate or to defect: $\mathcal{A} = \{c, d\}$. Player AP_1's choice is determined by the action in the MDP, while player AP_2 follows a mixed strategy with a fixed probability of p_c to choose cooperation, resulting in the following transition probabilities:

$$P(s_{cc}|s_0, c) = p_c \tag{5.1}$$

$$P(s_{cd}|s_0, c) = 1 - p_c \tag{5.2}$$

$$P(s_{dc}|s_0, d) = p_c \tag{5.3}$$

$$P(s_{dd}|s_0, d) = 1 - p_c \tag{5.4}$$

The reward depends not only on the two players but also a number of N additional passive stakeholders $\{PS_1, \ldots, PS_N\}$. Players and stakeholders form a set of involved entities $\mathcal{E} = \{AP_1, AP_2, PS_1, \ldots, PS_N\}$. For each participant $i \in \mathcal{E}$, a utility function $u_i(s)$ specifies their personal utilities for different outcomes. The utility is further split into the effects of short-term outcomes u_i^{st} and long-term outcomes u_i^{lt}. Additional weight parameters w_i specify how much each player's and each stakeholder's utility influences the final reward, representing how the decision-makers weigh their own as well as other participants' utilities.

$$u_i(s) = u_i^{st}(s) + u_i^{lt}(s) \tag{5.5}$$

$$r(s_0, c) = \sum_{i \in \mathcal{E}} p_c w_i u_i(s_{cc}) + (1 - p_c) w_i u_i(s_{cd}) \tag{5.6}$$

$$r(s_0, d) = \sum_{i \in \mathcal{E}} p_c w_i u_i(s_{dc}) + (1 - p_c) w_i u_i(s_{dd}) \tag{5.7}$$

The utility functions over short-term and long-term outcomes are probabilistic, and determined by sampling a specific outcome from a categorical distribution of short-term outcomes $o^{st} \sim P(O^{st})$ and long-term outcomes $o^{lt} \sim P(O^{lt})$. The sample probabilities of each individual outcome o_i depend on the state, resulting in the same set of outcomes but different probabilities for each terminal state of the MDP (with outcomes that can't occur in certain states being set to have a probability of 0).

5.3 Meta-level problem

The meta-level decision problem describes the mental operations a decision-maker in the object-level problem can use to make an optimal decision. The meta-level problem is formulated as a meta-MDP $(\mathcal{B}, \mathcal{C}, P, r_M)$, in which $\mathcal{B}$ specifies the belief-state and $\mathcal{C}$ specifies cognitive computations that update beliefs.

Belief State The belief state $b \in \mathcal{B}$ consists of two separate components, beliefs about the utility of different outcomes and beliefs over the other player's actions. Beliefs about utilities are modeled over the utility of outcome states $s \in \mathcal{S} \setminus \{s_0\}$ for a participant $i \in \mathcal{E}$ and split between the short-term and long-term outcomes $h \in \{st, lt\}$. Each belief $b_{h,i,s}$ is represented by a Normal-Gamma distribution $(\mu, \nu, \alpha, \beta)$, where μ represents the current belief about the expected utility a state s will have for a participant i over the timescale h, ν represents the number of past observations the belief is based on, and α and β are the parameters of a Gamma distribution that describes the belief about the variability of the outcome given a certain action and time horizon. Beliefs about the other player's probability to cooperate p_c are modeled using a beta distribution. Initial beliefs $b^{(0)}_{h,i,s}$ and $b^{(0)}_{p}$ are free parameters that can be used to represent prior information about a specific dilemma.

$$b_{h,i,s} = \left(\mu_{h,i,s}, \nu_{h,i,s}, , \alpha_{h,i,s}, , \beta_{h,i,s}\right) \tag{5.8}$$

$$b_p = \text{Beta}\left(\alpha_p, \beta_p\right) \tag{5.9}$$

Meta-level actions There are four types of meta-level actions (computations) $c \in \mathcal{C}$ available: considering short-term outcomes, considering long-term outcomes, considering the other player's strategy, and a termination action $\perp$ that stops the deliberation process.

Meta-level actions about short-term or long-term outcomes simulate possible outcomes $o_{h,i,s}$ and subsequently update the current belief state. Specifically, a computation $c_{h,i,s}$ simulates a possible outcome for state s and participant i on either a short-term timescale ($h = st$) or a long-term timescale ($h = lt$). The belief state $b_{h,i,s}$ is then updated on the imagined outcome o as follows (Murphy, 2007):

$$\mu' = \frac{\nu\mu + o}{\nu + 1} \tag{5.10}$$

$$\nu' = \nu + 1 \tag{5.11}$$

$$\alpha' = \alpha + \frac{1}{2} \tag{5.12}$$

$$\beta' = \beta + \frac{\nu}{\nu + 1}\frac{(o - \mu)^2}{2} \tag{5.13}$$

Imagined outcomes are sampled from adjusted outcome distributions $\hat{P}(O^b)$. To simulate that outcomes might be easier or harder to imagine than their actual probability of occurrence, an additional availability factor is used to scale the probabilities of different outcomes. For each outcome $o_i^b \in O^b$, an availability factor a_i^b is used to scale the probability of the specific outcome being recalled. After scaling each outcome, the probabilities are renormalized to arrive at the new probabil-

ity of sampling o_i^b under the adjusted outcome distribution $\hat{P}(O^b)$:

$$p'(o_i^b) = a_i^b p(o_i^b) \tag{5.14}$$

$$p''(o_i^b) = \frac{p'(o_i^b)}{\sum_{o^b \in O^b} p'(o^b)} \tag{5.15}$$

The meta-level action c_p investigates the other player's strategy by retrieving an observation $o_p \in 0, 1$ drawn from a Bernoulli distribution with $P(X = c) = p_c$. This action represents reasoning about how the other player might act by performing mental operations like remembering past interactions, recalling others' opinions of the other player, or thinking about their incentives. The belief is then updated based on the observation o_p:

$$b'_p = \text{Beta}\left(\alpha_p + o_p, \beta_p + 1 - o_p\right) \tag{5.16}$$

Meta-level rewards Both simulating outcomes and investigating the other player's actions incur a negative cost of planning: λ_o and λ_p. Terminating the planning process results in the expected reward of the object-level task based on the current belief state.

$$r_M(b, c_{b,i,s}) = \lambda_o \tag{5.17}$$

$$r_M(b, c_p) = \lambda_p \tag{5.18}$$

$$r_M(b, \perp) = \max_{a \in \mathcal{A}} \mathbb{E}\left[r(b, a)\right] \tag{5.19}$$

$$\begin{aligned}\mathbb{E}\left[r(b, c)\right] = &\sum_{i \in \mathcal{E}} \frac{\alpha_p}{\alpha_p + \beta_p} w_i(\mu_{st,i,cc} + \mu_{lt,i,cc}) + \\ &(1 - \frac{\alpha_p}{\alpha_p + \beta_p}) w_i(\mu_{st,i,cd} + \mu_{lt,i,cd})\end{aligned} \tag{5.20}$$

$$\begin{aligned}\mathbb{E}\left[r(b, d)\right] = &\sum_{i \in \mathcal{E}} \frac{\alpha_p}{\alpha_p + \beta_p} w_i(\mu_{st,i,dc} + \mu_{lt,i,dc}) + \\ &(1 - \frac{\alpha_p}{\alpha_p + \beta_p}) w_i(\mu_{st,i,dd} + \mu_{lt,i,dd})\end{aligned} \tag{5.21}$$

5.4 Extension: large action spaces

Object-level In this setting, players have a range of discrete actions $\mathcal{A} = \{a_1, ..., a_n\}$ that are ordered from most selfish (a_1) to most altruistic (a_n). The state space is extended to reflect all possible combinations of actions: $\mathcal{S} = \{s_0, s_{1,1}, ..., s_{1,n}, s_{2,1}, ..., s_{n,n}\}$. The second player follows a mixed strategy that is modeled by a Normal $\mathcal{N}\left(\mu_p, \sigma_p^2\right)$ distribution that describes how altruistic their actions usually are (μ_p) and how much they vary (σ_p). The second player's discrete action is obtained by sampling from the Normal distribution, rounding the sampled value to the nearest integer, and clamping the value between a_1 and a_n (see Equation 5.22).

$$a \sim \min(\max(\lfloor \mathcal{N}(\mu_p, \sigma_p^2) + 0.5 \rfloor, a_1, a_n) \tag{5.22}$$

Individual sample probabilities for each action can be obtained using the cumulative distribution function (CDF), resulting in a fixed mixed strategy (see Equation5.23). The reward for choosing an action $i \in \mathcal{A}$ is specified by summing over the possible effects on all participants and other stakeholders for each possible actions $j \in \mathcal{A}$ the other player might take (see Equation 5.24).

$$p(a_i|\mu_p, \sigma_p) = \begin{cases} \Phi\left(\frac{a_1+0.5-\mu_p}{\sigma_p}\right) & \text{for } a_i = a_1, \\ \Phi\left(\frac{a_i+0.5-\mu_p}{\sigma_p}\right) - \Phi\left(\frac{a_i-0.5-\mu_p}{\sigma_p}\right) & \text{for } a_1 < a_i < a_n, \\ 1 - \Phi\left(\frac{a_n-0.5-\mu_p}{\sigma_p}\right) & \text{for } a_i = a_n. \end{cases} \tag{5.23}$$

$$r(s_0, i) = \sum_{e \in \mathcal{E}} \sum_{j \in \mathcal{A}} p(a_j) w_e u_e(s_{i,j}) \tag{5.24}$$

Meta-level

$$b_p = \left(\mu_p, \nu_p, , \alpha_p, , \beta_p\right) \tag{5.25}$$

The belief over the other player's action probabilities b_p is modeled as a Normal-Gamma distribution. The meta-level action c_p investigates the other player's strategy by retrieving a discretized sample $o_p \in \mathcal{A}$ based on parameters (μ_p, λ_p) as described in Equation 5.22. The belief is then updated based on the observation o_p, as described in Equations 5.10 to 5.13.

Equation 5.26 describes the expected termination reward, which now depends on the belief over action probabilities for the other player $(\mu_p, \frac{\beta_p}{\alpha_p - 1})$ and the belief over rewards for all entities.

$$\mathbb{E}\left[r(b,i)\right] = \sum_{e \in \mathcal{E}} \sum_{j \in \mathcal{A}} p(a_j | \mu_p, \frac{\beta_p}{\alpha_p - 1}) w_e (b_{st,e,ij} + b_{lt,e,ij}) \tag{5.26}$$

Other possible extensions To extend the setting to a more general case with M active players $\{AP_1, \ldots, AP_M\}$ and K actions $\mathcal{A} = \{a_1, \ldots, a_k\}$. Analogous to the two-player setting, all players except for AP_1 follow a mixed strategy with a fixed probability distribution over K. The state space is extended to contain all possible combinations of different actions players can take and the reward function sums over all possible outcome states, weighted by their probability.

An additional extension could change the action sequence in the object-level MDP to be sequential, allowing the second player's strategy to depend on the chosen action of the first player. In this case, the second player would follow two separate mixed strategies with p_{cc} being the probability of cooperation if the first player cooperates, and p_{dc} being the probability of cooperation if the first player defects.

Lastly, outcomes can be determined at more fine-grained timescales as well, for example by adding medium-term outcomes. In the more general case, an arbitrary number of discrete time-steps could be added to the outcome calculation. This could be implemented into the model in a straightforward way by increasing the number of utility terms in Equation 5.5 and extending the belief states $b_{h,i,s}$ to include more timescales (i.e. $h \in \{st, mt, lt\}$).

5.5 Conclusion

In this chapter, I proposed a flexible metareasoning model for social dilemma tasks. The model takes the decision-maker's reciprocal and altruistic motivations into account by taking the impact of a decision on every affected person into account. The weight parameters w_i allow modeling different levels of altruistic motivation, ranging from fully selfish (by setting all weights other than one's own

to 0) to fully altruistic (by setting all weights to be equal). Additionally, the model explicitly takes the impact of actions on different time-scales into account, allowing to contrast the short-term benefits of self-centered decisions with potential long-term drawbacks like damaging one's reputation or depleting the resources of a public good.

Due to the model's flexibility in representing different motivations and outcomes, as well as the extended formulation with a wider range of possible actions, I believe the model to be applicable to a wide range of real-world social dilemmas. In the future, the model can be useful to investigate and understand which kind of strategies people use when faced with social dilemmas, find optimal solutions to specific dilemmas formulated within the model, and potentially improve people's decision-making in social dilemmas by teaching them strategies that better align with their long-term values and overcome human biases.

6

Discussion

In complex real-world situations, humans often make suboptimal decisions. Teaching people more resource-rational decision strategies that make better trade-offs between the expended cognitive resources and the utility of the decision's outcome has the potential of substantially improving peoples lives. A promising approach towards this goal is to automatically discover efficient planning strategies using strategy discovery methods (Callaway et al., 2018a) and teach them to people using intelligent cognitive tutors (Callaway et al., 2022a). However, multiple key limitations in existing

methods prevented them from being applied in the real world. First, both strategy discovery methods and cognitive tutors were not scalable enough for the size and complexity of real-world environments. Second, existing methods were incompatible with partially observable environments, a common characteristic of many real-world tasks. Lastly, there was no prior work on how to model real-world tasks as metareasoning problems. The key motivation of this thesis was to address these limitations and bring strategy discovery methods closer to being used to improve human decision-making in the real world.

The first challenge I identified was the limited scalability of existing strategy discovery methods and cognitive tutors. Solving the meta-MDP associated with metareasoning problems requires evaluating an exponentially increasing number of belief states with increasing environment size. This presents a severe computational bottleneck that prevents existing strategy discovery methods from being applied to larger environments required to model many real-world problems. To address this challenge, I introduced hierarchical-BMPS, a state-of-the-art strategy discovery algorithm that can solve larger meta-MDPs than previously possible by hierarchically decomposing the metareasoning problem into two separate steps: first selecting a goal to pursue and then planning how to best achieve the selected goal. I evaluated hierarchical-BMPS in simulation experiments that showed it outperforms a wide range of baseline algorithms, as well as the strategies people discover by exploring the environment themselves. Using a demonstration-based tutoring system, I also demonstrated that the strategies discovered by hierarchical-BMPS can be taught to people and significantly improve the resource-rationality of the strategies they learn compared to multiple control conditions.

The cognitive tutor proposed by Callaway et al. (2022a) is similarly limited in its scalability. Since the tutor relies on computing the Q-value of all available meta-actions, it can only be applied to small, easily solvable environments. Additionally, existing tutors let people choose meta-actions freely and provide corrective feedback only after the action is selected. This process can make learning new strategies in large environments both time-consuming and frustrating, since the chance of

coincidentally selecting the correct action during learning is much lower than it is for small environments. To overcome these limitations, I created a new metacognitive tutor that uses the myopic VOC to deliver binary feedback instead of relying on exact Q-values. To simplify the learning problem further, the tutor utilizes shaping schedules that gradually increase the complexity of multiple aspects of the planning problem during learning. The first aspect is the size of the environment, which gradually increases throughout the training session, starting with a much smaller and simplified version of the problem. The second aspect is the number of meta-actions participants can choose between, which starts with only two relevant choices and then gradually increases until participants. Lastly, as implemented in Section 4.5.1, the type of meta-action can also be broken down into individual components like selecting between advisors and selecting between criteria, allowing the tutor to teach individual parts of the overall strategy in an isolated manner. In the reported training experiments, the improvements to the cognitive tutor proved effective in teaching people more complex metareasoning strategies than previously possible.

The second key limitation with existing strategy discovery methods was their limited ability to model the complexity of real-world problems. To this end, I introduced a new metareasoning model and strategy discovery algorithm for partially observable environments. In this setting, meta-actions reveal only uncertain estimates of the object-level effects of actions. Partial observability substantially increases the complexity of the metareasoning task, since decision makers also have to take the uncertainty in their beliefs into account when selecting planning computations. To discover planning strategies in partially observable environments, I developed MGPO, a new metareasoning algorithm that myopically approximates the VOC by applying Bayesian belief updates. I tested MGPO in simulation experiments, where it proved faster and more resource-rational than two baseline algorithms. I further tested whether MGPO can be used to improve human decision-making by teaching its discovered strategies to people in an online experiment. Training with the MGPO-based tutor significantly improved the resource-rationality of people's strategies compared to two baseline

conditions.

Finally, I addressed the last limitation by modeling two more realistic tasks, project selection and social dilemmas, as metareasoning problems and applying strategy discovery to one of them. To apply strategy discovery to the project selection task, I developed MGPS, an improved strategy discovery method adapted to the unique characteristics of project selection. The concrete project selection environments were modeled from data of a real-world project selection task described by Khalili-Damghani and Sadi-Nezhad (2013). I evaluated MGPS' performance on the project selection task in a simulation experiment and a human training experiment. In the simulation experiment, MGPS achieved a higher resource-rationality score at a fraction of the runtime of the baseline algorithm, PO-UCT. In the training experiment, the strategies discovered by MGPS significantly improved human resource-rationality on project selection tasks when taught by the cognitive tutor. Additionally, I developed a mathematical framework for extending strategy discovery methods to a second real-world task, social dilemmas. The model defines resource-rational decision-making in social dilemma situations for varying levels of social preferences and taking both short and long-term consequences for all affected parties into account.

The work presented in this thesis relates closely to prior work on strategy discovery, resource-rationality, and intelligent tutoring systems. Decision strategies are framed within the paradigm of resource-rationality (Lieder and Griffiths, 2020a), which defines optimal decision-making as efficiently trading off the quality of a decision with the cost of the cognitive resources expended during the decision-making process. This framework is combined with a model of human planning, the Mouselab-MDP (Callaway et al., 2017), which externalizes the usually unobservable internal human planning process as a meta-level MDP (Hay et al., 2014). The developed strategy discovery methods extend prior work on strategy discovery, which has developed successful metareasoning algorithms on small environments (Russell and Wefald, 1991a, Lieder et al., 2017, Hay et al., 2014, Lin et al., 2015, Callaway et al., 2018a), to larger and more complex planning problems. The cognitive tutor

directly builds upon prior work on cognitive tutors by Callaway et al. (2022a), who developed a tutor that gives metacognitive feedback to people. This tutor has been improved upon by making it more computationally efficient through approximating the VOC instead of relying on the exact solution to the metareasoning problem, as well as utilizing shaping schedules that incrementally increase the difficulty of the learning task.

The empirical findings have several important implications for the debate about human rationality (Tetlock and Mellers, 2002, Stanovich and West, 2000, Gigerenzer and Selten, 2002, Lieder and Griffiths, 2020a). Across all human training experiments in this thesis, I consistently found that human resource-rationality was substantially lower than the resource-rationality of the strategies discovered by the strategy discovery algorithms. Additionally, teaching people the automatically discovered planning strategies consistently improved their resource-rationality score. These findings suggest that, unlike in small problems (Callaway et al., 2018b, 2020), the planning strategies people use in large and partially observable decision-problems might be far from resource-rational. Although some of the discrepancies could be due to unaccounted cognitive costs, people's limited resource-rationality cannot be completely explained by cognitive costs (Krueger et al., 2024). Therefore, people choosing their planning operations suboptimally likely plays an important role as well. These results add three important nuances to the resource-rational perspective on human rationality (Lieder and Griffiths, 2020a,b). First, it suggests that human cognition might systematically deviate from the principles of resource-rationality. Second, it suggests that the magnitude of these deviations depends on the complexity of the task. Third, it suggests that those sub-optimalities can be overcome by teaching people resource-rational strategies.

It was recently suggested that the computational efficiency of human cognition can be understood in terms of rational metareasoning (Griffiths et al., 2019). According to this view, the human mind strategically selects its computations to maximize the expected utility of its decisions minus the cost of computation. Performing rational metareasoning exactly would be computationally

intractable (Russell and Wefald, 1991a). However, it is believed that people may be able to approximate rational metareasoning efficiently (Lieder and Griffiths, 2017). One of the simplest approximations to rational metareasoning is the meta-greedy policy, which only looks one single computation ahead (Russell and Wefald, 1991a). In my experiments with partial observability, people's level of resource-rationality was substantially lower than that of the meta-greedy policy. This suggests that if people approximate rational metareasoning at all, then their approximation is probably even simpler and less accurate than the meta-greedy approximation. Recent research suggests that people learn cognitive strategies partly through simple, model-free reinforcement learning mechanisms (He et al., 2021, Krueger et al., 2017). If this is the case, then methods that have been developed to foster this simple form of learning, such as feedback (Callaway et al., 2022a) and shaping (Skinner, 1953, 1958), may be very appropriate for helping people learn how to make better decisions. Since our cognitive tutor was based on those principles, its effectiveness suggests that this might be the case.

A fundamental problem of utilizing strategy discovery for improving human decision-making is that it is unclear, how well people can transfer their metacognitive strategies to other similar problems. In cognitive science, real-world transfer of improvements from cognitive training is a debated topic with mixed evidence (Lintern and Boot, 2021, Hardy et al., 2015). Prior work on cognitive tutors found positive evidence for strategy transfer between small and similar planning problems (Callaway et al., 2022a), but transfer between simplified training environments and more complex real-world has not been tested. If such transfer were possible, human planning in the real world could potentially be improved by letting them practice useful and resource-rational strategies such as far-sighted planning in simplified settings. However, since the transfer to real-world applications is highly unclear, this thesis proposes a different approach to improving real-world decision-making: modeling relevant real-world problems directly and improving human decision-making in environments that resemble the real-world task as closely as possible by estimating relevant parameters from data. This approach minimizes the need for transfer and would ideally allow people to directly apply

the learned strategies.

Additional limitations concern the concrete strategy discovery methods developed in this thesis. Firstly, the methods require a precise model of the environment. In real-world settings, the decision environment is likely built from a combination of real-world data and the knowledge of domain experts. How well the strategies discovered in the model of the environment translate to real applications likely depends on an accurate model of the task. However, when the available data or domain knowledge is limited or too expensive, it can be difficult to construct an accurate metareasoning model. Future work could investigate and potentially address this issue by extending MGPS with a Bayesian inference approach to estimate uncertain parameters of the environment (Mehta et al., 2022). This extension would make MGPS robust to uncertainties in the underlying distributions of evaluation criteria or expert reliability, and make it possible to discover resource-rational strategies even when data to estimate environment parameters is limited. A worry with integrating uncertainty over the environment structure is that the resulting decision strategies could prove very complicated and difficult to learn for humans when combined with the larger and more complex problem settings required for real-world applications.

The strategy discovery algorithms I developed for partially observable environments, MGPO and MGPS, are limited by the fact that they are using myopic approximations of the VOC. This approximation has worked very well in the environments reported in this thesis, which makes me optimistic that it will be sufficient in many problem settings relevant to real-world applications. However, it could lead to unforeseen inaccuracies in other environments, especially when object-level rewards depend on multiple non-linearly combined features. To overcome this limitation, MGPO could be extended to plan multiple metareasoning steps into the future, although this has proved very computationally expensive in preliminary testing and wasn't pursued further since the environments presented in Chapter 3 and 4 didn't require it. Combining MGPO with the hierarchical decomposition of the metareasoning problem presents one promising method of increasing

its scalability. An alternative approach could be to utilize deep reinforcement learning for metareasoning, motivated by its successes in complicated planning environments like chess and Go (Silver et al., 2018). However, these approaches come with their own problems. Deep reinforcement learning models can be difficult and expensive to train, which would have to be repeated for every environment type they are applied to. Additionally, there is no guarantee that the learned models will act resource-rational in all situations, as they could have serious blind-spots that result in suboptimal actions in some situations (Wang et al., 2023).

While the cognitive intelligent tutors I developed effectively taught people the discovered strategies and overall improved their resource-rationality, people generally did not fully understand the taught strategies. This is indicated by click agreement scores of less than 50% across almost all training experiments. Partially, learning can likely be improved by increasing the amount of practice participants receive, which was limited by practical concerns about the complexity and length of crowdsourced online experiments that wouldn't be present in the same way when training on relevant real-world tasks. To truly maximize learning, I also expect further improvements to the cognitive tutor to be required, where the exact adjustments needed are likely to be at least in part domain specific. For example, the tutor for project selection simplifies the problem by teaching to select between advisors and project criteria separately. The main purpose of the training experiments was to demonstrate that effective teaching using cognitive tutors in larger and more complex settings is possible, and not necessarily to optimize the teaching methodology.

Since optimizing the cognitive tutor was not the core aim of this thesis, I am looking forward to future pedagogical work that improves and further evaluates the mechanisms by which the cognitive tutor teaches strategies. The most straightforward way to improve the existing tutor is to extend the training schedules, focusing on the parts of the optimal strategy participants had the most trouble learning. Another promising approach could be to extend research on making the discovered strategies interpretable (Skirzyński et al., 2021) to partially observable environments. This would allow it

to automatically create verbal descriptions of the discovered strategies. Alternatively, LLMs could be used to generate verbal feedback based on the computed VOC values that can explain why certain meta-actions are correct or incorrect, as well as making the training experience more varied and motivating.

Another core limitation of applying strategy discovery methods to real-world problems is the difficulty of identifying relevant tasks that fit well into the framework and for which it is possible to find sufficient data or experts with domain knowledge to create an accurate metareasoning model. Generally, strategy discovery methods should be useful on real-world tasks in which additional information can be used to refine a decision but comes at a significant cost, making it important to weigh which and how much information to acquire. This is arguably true for many significant real-world settings, for example deciding which charity to donate to for the highest positive impact, which form of education to pursue, which field to work in, or which applicant to hire. However, for many of these tasks, it is very difficult to construct accurate metareasoning models. For example, selecting which job to pursue is clearly a highly important decision with a high impact on both personal life satisfaction (Unanue et al., 2017) and potential positive impact (MacAskill, 2015). But creating a metareasoning model of career choice would require hard to collect data on how much a person benefits from additional information search based on their personal experience and skill set.

Encouraged by the promising results from successfully teaching humans in the naturalistic model of project selection, I am most excited about future work assessing the real-world impact of improving people's decision-making by evaluating their decisions directly in the real world. While my training experiments moved closer towards real-world tasks by modeling the project selection problem from real data, all experiments were conducted in artificial online settings. The most direct way to test this would be to evaluate project selection or decision-making in social dilemmas in a field test, for example by evaluating the project-selection tutor against professional decision-makers in a business setting. Project-selection tutors could also be integrated into MBA programs to teach

future decision-makers efficient decision strategies as part of their education. The project selection framework can be adapted to other similar decision problems in the future, for example a company deciding between company strategies or a funding agency deciding which proposal to fund. More generally, the methods proposed in this thesis should be applicable to new domains as well, which would require creating a metareasoning model as demonstrated with project selection and social dilemmas, and then utilize MGPO or hierarchical-BMPS to discover resource-rational strategies. Potential further real-world applications that could be modeled in this way are hiring or admission selection in which a decision-maker has to decide between different applicants, choice of career path or study programme, or teaching people when to seek advice from coworkers or the internet during their jobs.

In summary, I presented multiple strategy discovery algorithms and cognitive tutors that leverage human-centered AI to improve human planning and decision-making in large, complex, and real-world environments, and demonstrated its effectiveness through simulations and training experiments. The reported advances have applications in artificial intelligence, cognitive science, education, human-computer interaction, and improving both individual and organizational decision-making. The strategy discovery algorithms expand the size and complexity of metareasoning problems the approach can be applied to, including real-world tasks like project selection. Being able to discover resource-rational strategies for these problems makes it possible to scale up resource-rational analysis for understanding the cognitive mechanisms of decision-making (Lieder and Griffiths, 2020a), as well as improving human decision-making in real-world settings. The advances to cognitive tutors make it possible to teach highly complicated and realistic planning strategies to people. Overall, the methods developed in this thesis are an important step towards intelligent systems that surpass human resource-rationality, understanding how people make decisions, and leveraging artificial intelligence to improve human decision-making in the real world. Future work on cognitive tutors might make it feasible to cost-effectively boost the decision-making competence of millions of

people. It might thereby become possible to improve crucial decisions without resorting to nudging (Hertwig and Grüne-Yanoff, 2017) and without interfering with people's freedom to make their own decisions (Thaler and Sunstein, 2021). I am optimistic that the general methodology is applicable to many real-world problems, offering a promising pathway to teach people efficient strategies for making better decisions.

References

M. Abdel-Basset, A. Atef, and F. Smarandache. A hybrid neutrosophic multiple criteria group decision making approach for project selection. *Cognitive Systems Research*, 57:216–227, 2019.

W. Abrahamse, L. Steg, C. Vlek, and T. Rothengatter. A review of intervention studies aimed at household energy conservation. *Journal of environmental psychology*, 25(3):273–291, 2005.

M. A. Adams, M. Bruening, P. Ohri-Vachaspati, and J. C. Hurley. Location of school lunch salad bars and fruit and vegetable consumption in middle schools: A cross-sectional plate waste study. *Journal of the Academy of Nutrition and Dietetics*, 116(3):407–416, 2016.

V. Aleven, B. Mclaren, I. Roll, and K. Koedinger. Toward meta-cognitive tutoring: A model of help seeking with a cognitive tutor. *International Journal of Artificial Intelligence in Education*, 16 (2):101–128, 2006.

J. R. Anderson, C. F. Boyle, and B. J. Reiser. Intelligent tutoring systems. *Science*, 228(4698): 456–462, 1985.

T. Araujo, N. Helberger, S. Kruikemeier, and C. H. De Vreese. In ai we trust? perceptions about automated decision-making by artificial intelligence. *AI & society*, 35:611–623, 2020.

S. Z. Attari, D. H. Krantz, and E. U. Weber. Reasons for cooperation and defection in real-world social dilemmas. *Judgment and Decision Making*, 9(4):316–334, 2014.

Y. Benjamini and Y. Hochberg. Controlling the false discovery rate: a practical and powerful approach to multiple testing. *Journal of the Royal statistical society: series B (Methodological)*, 57 (1):289–300, 1995.

J. Beshears and H. Kosowsky. Nudging: Progress to date and future directions. *Organizational behavior and human decision processes*, 161:3–19, 2020.

S. Bonaccio and R. S. Dalal. Advice taking and decision-making: An integrative literature review, and implications for the organizational sciences. *Organizational behavior and human decision processes*, 101(2):127–151, 2006.

M. M. Botvinick. Hierarchical models of behavior and prefrontal function. *Trends in cognitive sciences*, 12(5):201–208, 2008.

G. E. Box. Some theorems on quadratic forms applied in the study of analysis of variance problems, i. effect of inequality of variance in the one-way classification. *The annals of mathematical statistics*, 25(2):290–302, 1954.

K. H. Brodersen, C. S. Ong, K. E. Stephan, and J. M. Buhmann. The balanced accuracy and its posterior distribution. In *2010 20th international conference on pattern recognition*, pages 3121–3124. IEEE, 2010.

T. Bucher, C. Collins, M. E. Rollo, T. A. McCaffrey, N. De Vlieger, D. Van der Bend, H. Truby, and F. J. Perez-Cueto. Nudging consumers towards healthier choices: a systematic review of positional influences on food choice. *British Journal of Nutrition*, 115(12):2252–2263, 2016.

F. Burstein, C. W Holsapple, and D. J. Power. Decision support systems: a historical overview. *Handbook on decision support systems 1: Basic themes*, pages 121–140, 2008.

F. Callaway, F. Lieder, P. M. Krueger, and T. L. Griffiths. Mouselab-MDP: A new paradigm for tracing how people plan. In *The 3rd Multidisciplinary Conference on Reinforcement Learning and Decision Making, Ann Arbor, MI*, 2017. URL https://osf.io/vmkrq/.

F. Callaway, S. Gul, P. M. Krueger, T. L. Griffiths, and F. Lieder. Learning to select computations. *Uncertainty in Artificial Intelligence*, 2018a.

F. Callaway, F. Lieder, P. Das, S. Gul, P. Krueger, and T. Griffiths. A resource-rational analysis of human planning. In C. Kalish, M. Rau, J. Zhu, and T. Rogers, editors, *CogSci 2018*, 2018b.

F. Callaway, B. van Opheusden, S. Gul, P. Das, P. Krueger, F. Lieder, and T. Griffiths. Human planning as optimal information seeking, 2020. Manuscript under review.

F. Callaway, Y. R. Jain, B. van Opheusden, P. Das, G. Iwama, S. Gul, P. M. Krueger, F. Becker, T. L. Griffiths, and F. Lieder. Leveraging artificial intelligence to improve people's planning strategies. *Proceedings of the National Academy of Sciences*, 119(12):e2117432119, 2022a.

F. Callaway, B. van Opheusden, S. Gul, P. Das, P. M. Krueger, T. L. Griffiths, and F. Lieder. Rational use of cognitive resources in human planning. *Nature Human Behaviour*, 6(8):1112–1125, 2022b.

F. Callaway, T. L. Griffiths, and G. K. Rehbinder. Rational heuristics for one-shot games, 2023.

A. F. Carazo, I. Contreras, T. Gómez, and F. Pérez. A project portfolio selection problem in a group decision-making context. *Journal of Industrial & Management Optimization*, 8(1):243, 2012. Publisher: American Institute of Mathematical Sciences.

C. S. Carver and M. F. Scheier. *On the self-regulation of behavior*. Cambridge University Press, 2001.

M. Chi and K. VanLehn. Meta-cognitive strategy instruction in intelligent tutoring systems: how, when, and why. *Journal of Educational Technology & Society*, 13(1):25–39, 2010.

J. Cohen. *Statistical power analysis for the behavioral sciences*. Academic press, 2013.

S. Coldrick, C. Lawson, P. Ivey, and C. Lockwood. A decision framework for r&d project selection. In *IEEE International Engineering Management Conference*, volume 1, page 413–418. IEEE, 2002.

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

A. T. Corbett, K. R. Koedinger, and J. R. Anderson. Intelligent tutoring systems. In *Handbook of human-computer interaction*, pages 849–874. Elsevier, 1997.

R. M. Dawes. Social dilemmas. *Annual review of psychology*, 31(1):169–193, 1980.

D. G. B. de Souza, E. A. dos Santos, N. Y. Soma, and C. E. S. da Silva. MCDM-Based R&D Project Selection: A Systematic Literature Review. *Sustainability*, 13(21):11626, 2021. Publisher: MDPI.

K. Ellis, M. Nye, Y. Pu, F. Sosa, J. Tenenbaum, and A. Solar-Lezama. Write, execute, assess: Program synthesis with a repl. *Advances in Neural Information Processing Systems*, 32, 2019.

A. Fawzi, M. Balog, A. Huang, T. Hubert, B. Romera-Paredes, M. Barekatain, A. Novikov, F. J. R Ruiz, J. Schrittwieser, G. Swirszcz, et al. Discovering faster matrix multiplication algorithms with reinforcement learning. *Nature*, 610(7930):47–53, 2022.

M. Fendley and S. Narayanan. Decision aiding to overcome biases in object identification. *Advances in Human-Computer Interaction*, 2012:7–7, 2012.

G. Gigerenzer and W. Gaissmaier. Heuristic decision making. *Annual review of psychology*, 62(1): 451–482, 2011.

G. Gigerenzer and R. Selten. *Bounded rationality: The adaptive toolbox*. MIT press, 2002.

G. Gigerenzer and P. M. Todd. *Simple heuristics that make us smart*. Oxford University Press, USA, 1999.

F. Gino. Do we listen to advice just because we paid for it? the impact of advice cost on its use. *Organizational Behavior and Human Decision Processes*, 107:234–245, 2008.

A. C. Graesser, M. W. Conley, and A. Olney. *Intelligent tutoring systems.* American Psychological Association, 2012.

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

E. Guerra and G. Mellado. A-book: A feedback-based adaptive system to enhance meta-cognitive skills during reading. *Frontiers in Human Neuroscience*, 11:98, 2017.

A. Guez, D. Silver, and P. Dayan. Efficient bayes-adaptive reinforcement learning using sample-based search. In F. Pereira, C. Burges, L. Bottou, and K. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012. URL https://proceedings.neurips.cc/paper/2012/file/35051070e572e47d2c26c241ab88307f-Paper.pdf.

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

G. Hardin. The tragedy of the commons. In *Classic Papers in Natural Resource Economics Revisited*, pages 145–156. Routledge, 2018.

J. L. Hardy, R. A. Nelson, M. E. Thomason, D. A. Sternberg, K. Katovich, F. Farzin, and M. Scanlon. Enhancing cognitive abilities with comprehensive training: A large, online, randomized, active-controlled trial. *PloS one*, 10(9):e0134467, 2015.

N. Hay, S. Russell, D. Tolpin, and S. E. Shimony. Selecting computations: Theory and applications. *arXiv preprint arXiv:1408.2048*, 2014.

R. He, Y. R. Jain, and F. Lieder. 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*, 2021.

L. Heindrich and F. Lieder. Leveraging automatic strategy discovery to teach people how to select better projects. *arXiv preprint arXiv:2406.04082*, 2024.

L. Heindrich, S. Consul, and F. Lieder. Leveraging ai to improve human planning in large partially observable environments. *arXiv preprint arXiv:2302.02785*, 2023.

A. Henriksen and A. Traynor. A practical r&d project-selection scoring tool. *IEEE Transactions on Engineering Management*, 46(2):158–170, 1999. doi: 10.1109/17.759144.

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

D. Hummel and A. Maedche. How effective is nudging? a quantitative review on the effect sizes and limits of empirical nudging studies. *Journal of Behavioral and Experimental Economics*, 80: 47–58, 2019.

Q. J. Huys, N. Eshel, E. O'Nions, L. Sheridan, P. Dayan, and J. P. Roiser. Bonsai trees in your head: how the Pavlovian system sculpts goal-directed choices by pruning decision trees. *PLoS computational biology*, 8(3), 2012.

Y. R. Jain, F. Callaway, T. L. Griffiths, P. Dayan, P. M. Krueger, and F. Lieder. A computational process-tracing method for measuring people's planning strategies and how they change over time. *Behavior Research Methods*, 2022.

L. P. Kaelbling and T. Lozano-Pérez. Hierarchical planning in the now. In *Workshops at the Twenty-Fourth AAAI Conference on Artificial Intelligence*, 2010.

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

D. Kahneman, J. L. Knetsch, and R. H. Thaler. Anomalies: The endowment effect, loss aversion, and status quo bias. *Journal of Economic perspectives*, 5(1):193–206, 1991.

A. Kemtur, Y. Jain, A. Mehta, F. Callaway, S. Consul, J. Stojcheski, and F. Lieder. Leveraging machine learning to automatically derive robust planning strategies from biased models of the environment. In *CogSci 2020*. CogSci, 2020.

K. Khalili-Damghani and S. Sadi-Nezhad. A hybrid fuzzy multiple criteria group decision making approach for sustainable project selection. *Applied Soft Computing*, 13(1):339–352, 2013.

K. R. Koedinger and A. Corbett. Cognitive tutors. *Smart machines in education*, pages 145–167, 2001.

K. R. Koedinger, J. R. Anderson, W. H. Hadley, and M. A. Mark. Intelligent tutoring goes to school in the big city. *International Journal of Artificial Intelligence in Education*, 8:30–43, 1997.

P. Kollock. Social dilemmas: The anatomy of cooperation. *Annual review of sociology*, 24(1): 183–214, 1998.

B. Kornfeld and S. Kara. Selection of Lean and Six Sigma projects in industry. *International Journal of Lean Six Sigma*, 2013. Publisher: Emerald Group Publishing Limited.

R. Koster, J. Balaguer, A. Tacchetti, A. Weinstein, T. Zhu, O. Hauser, D. Williams, L. Campbell-Gillingham, P. Thacker, M. Botvinick, et al. Human-centred mechanism design with democratic ai. *Nature Human Behaviour*, 6(10):1398–1407, 2022.

R. Koster, M. Pîslar, A. Tacchetti, J. Balaguer, L. Liu, R. Elie, O. P. Hauser, K. Tuyls, M. Botvinick, and C. Summerfield. Using deep reinforcement learning to promote sustainable human behaviour on a common pool resource problem. *arXiv preprint arXiv:2404.15059*, 2024.

P. M. Krueger, F. Lieder, and T. Griffiths. Enhancing metacognitive reinforcement learning using reward structures and feedback. In *Proceedings of the 39th Annual Conference of the Cognitive Science Society*, 2017.

P. M. Krueger, F. Callaway, S. Gul, T. L. Griffiths, and F. Lieder. Identifying resource-rational heuristics for risky choice. *Psychological Review*, 2024.

R. P. Larrick. Debiasing. *Blackwell handbook of judgment and decision making*, pages 316–338, 2004.

R. P. Larrick, J. N. Morgan, and R. E. Nisbett. Teaching the use of cost-benefit reasoning in everyday life. *Psychological Science*, 1(6):362–370, 1990.

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

F. Lieder and T. L. Griffiths. Resource-rational analysis: understanding human cognition as the optimal use of limited computational resources. *Behavioral and Brain Sciences*, 43, 2020a.

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

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

F. Lieder, F. Callaway, Y. Jain, P. Krueger, P. Das, S. Gul, and T. Griffiths. A cognitive tutor for helping people overcome present bias. In *RLDM 2019*, 2019a.

F. Lieder, O. X. Chen, P. M. Krueger, and T. L. Griffiths. Cognitive prostheses for goal achievement. *Nature human behaviour*, 3(10):1096–1106, 2019b.

F. Lieder, F. Callaway, Y. R. Jain, P. Das, G. Iwama, S. Gul, P. Krueger, and T. L. Griffiths. Leveraging artificial intelligence to improve people's planning strategies, 2020. Manuscript in revision.

C. H. Lin, A. Kolobov, E. Kamar, and E. Horvitz. Metareasoning for planning under uncertainty. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*, 2015.

G. Lintern and W. R. Boot. Cognitive training: Transfer beyond the laboratory? *Human Factors*, 63(3):531–547, 2021.

L. Litman, J. Robinson, and T. Abberbock. Turkprime. com: A versatile crowdsourcing data acquisition platform for the behavioral sciences. *Behavior research methods*, 49(2):433–442, 2017.

F. Liu, W.-d. Zhu, Y.-w. Chen, D.-l. Xu, and J.-b. Yang. Evaluation, ranking and selection of R&D projects by multiple experts: an evidential reasoning rule based approach. *Scientometrics*, 111(3): 1501–1519, 2017. Publisher: Springer.

S. Liu, K. C. See, K. Y. Ngiam, L. A. Celi, X. Sun, M. Feng, et al. Reinforcement learning for clinical decision support in critical care: comprehensive review. *Journal of medical Internet research*, 22 (7):e18477, 2020.

W. MacAskill. *Doing good better: Effective altruism and a radical new way to make a difference*. Guardian Faber Publishing, 2015.

D. J. Mankowitz, A. Michi, A. Zhernov, M. Gelmi, M. Selvi, C. Paduraru, E. Leurent, S. Iqbal, J.-B. Lespiau, A. Ahern, et al. Faster sorting algorithms discovered using deep reinforcement learning. *Nature*, 618(7964):257–263, 2023.

G. M. Marakas. *Decision support systems in the 21st century*, volume 134. Prentice Hall Upper Saddle River, 2003.

B. Marthi, S. J. Russell, and J. A. Wolfe. Angelic semantics for high-level actions. In *Seventeenth International Conference on Automated Planning and Scheduling*, pages 232–239, 2007.

A. Mehta, Y. R. Jain, A. Kemtur, J. Stojcheski, S. Consul, M. Tošić, and F. Lieder. Leveraging machine learning to automatically derive robust decision strategies from imperfect knowledge of the real world. *Computational Brain & Behavior*, 5(3):343–377, 2022.

G. A. Miller, E. Galanter, and K. H. Pribram. Plans and the structure of behavior., 1960.

V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

J. Mockus. *Bayesian approach to global optimization: theory and applications*, volume 37. Springer Science & Business Media, 2012.

V. Mohagheghi, S. M. Mousavi, J. Antuchevičienė, and M. Mojtahedi. Project portfolio selection problems: a review of models, uncertainty approaches, solution techniques, and case studies. *Technological and Economic Development of Economy*, 25(6):1380–1412, 2019.

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

E. Mousavinasab, N. Zarifsanaiey, S. R. Niakan Kalhori, M. Rakhshan, L. Keikha, and M. Ghazi Saeedi. Intelligent tutoring systems: a systematic review of characteristics, applications, and evaluation methods. *Interactive Learning Environments*, 29(1):142–163, 2021.

K. P. Murphy. Conjugate bayesian analysis of the gaussian distribution. *def*, 1(2σ2):16, 2007.

S. Nasiriany, V. Pong, S. Lin, and S. Levine. Planning with goal-conditioned policies. In *Advances in Neural Information Processing Systems*, pages 14843–14854, 2019.

K. Noguchi, Y. R. Gel, E. Brunner, and F. Konietschke. nparLD: An R software package for the nonparametric analysis of longitudinal data in factorial experiments. *Journal of Statistical Software*, 50(12):1–23, 2012. URL http://www.jstatsoft.org/v50/i12/.

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

K. Olsen, A. Roepstorff, and D. Bang. Knowing whom to learn from: individual differences in metacognition and weighting of social information. *PsyArXiv*, 2019.

T. H. Payne. Computer decision support systems. *Chest*, 118(2):47S–52S, 2000.

K. Pertsch, O. Rybkin, F. Ebert, C. Finn, D. Jayaraman, and S. Levine. Long-horizon visual planning with goal-conditioned hierarchical predictors. *arXiv preprint arXiv:2006.13205*, 2020.

A. Rapoport and A. M. Chammah. *Prisoner's dilemma: A study in conflict and cooperation*, volume 165. University of Michigan press, 1965.

I. Roll, R. S. Baker, V. Aleven, B. M. McLaren, and K. R. Koedinger. Modeling students' metacognitive errors in two intelligent tutoring systems. In *User Modeling 2005: 10th International Conference, UM 2005, Edinburgh, Scotland, UK, July 24-29, 2005. Proceedings 10*, pages 367–376. Springer, 2005.

D. Ronayne, D. Sgroi, et al. *Ignoring good advice*. University of Warwick, Centre for Competitive Advantage in the Global ..., 2019.

S. Russell. *Human compatible: AI and the problem of control*. Penguin Uk, 2019.

S. Russell and P. Norvig. Artificial intelligence: a modern approach, 2002.

S. Russell and E. Wefald. Principles of metareasoning. *Artificial intelligence*, 49(1–3):361–395, 1991a.

S. J. Russell and E. Wefald. *Do the right thing: studies in limited rationality*. MIT press, 1991b.

E. D. Sacerdoti. Planning in a hierarchy of abstraction spaces. *Artificial intelligence*, 5(2):115–135, 1974.

S. Sadi-Nezhad. A state-of-art survey on project selection using mcdm techniques. *Journal of Project Management*, 2(1):1–10, 2017.

M. Santos Silva. Nudging and other behaviourally based policies as enablers for environmental sustainability. *Laws*, 11(1):9, 2022.

A. C. Schapiro, T. T. Rogers, N. I. Cordova, N. B. Turk-Browne, and M. M. Botvinick. Neural representations of events arise from temporal community structure. *Nature neuroscience*, 16(4): 486, 2013.

A. T. Schmidt and B. Engelen. The ethics of nudging: An overview. *Philosophy Compass*, 15, 2020. URL https://api.semanticscholar.org/CorpusID:214272552.

R. L. Schmidt and J. R. Freeland. Recent progress in modeling R&D project-selection processes. *IEEE Transactions on Engineering Management*, 39(2):189–201, 1992. Publisher: IEEE.

E. Sezener and P. Dayan. Static and dynamic values of computation in mcts. In *Conference on Uncertainty in Artificial Intelligence*, pages 31–40. PMLR, 2020.

D. Silver and J. Veness. Monte-carlo planning in large pomdps. *Advances in neural information processing systems*, 23, 2010.

D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.

D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, et al. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362(6419):1140–1144, 2018.

H. A. Simon. Rational choice and the structure of the environment. *Psychological review*, 63(2): 129, 1956.

H. A. Simon. Bounded rationality. *Utility and probability*, pages 15–18, 1990.

B. Skinner. *Shaping and maintaining operant behavior*, pages 91–106. Free Press New York, 1953.

B. Skinner. Reinforcement today. *American Psychologist*, 13(3):94, 1958.

J. Skirzyński, F. Becker, and F. Lieder. Automatic discovery of interpretable planning strategies. *Machine Learning*, 110(9):2641–2683, 2021.

A. Solway, C. Diuk, N. Córdova, D. Yee, A. G. Barto, Y. Niv, and M. M. Botvinick. Optimal behavioral hierarchy. *PLoS computational biology*, 10(8), 2014.

K. E. Stanovich. *Decision making and rationality in the modern world*. Oxford University Press, USA, 2010.

K. E. Stanovich. *Rationality and the reflective mind*. Oxford University Press, 2011.

K. E. Stanovich and R. F. West. Individual differences in reasoning: Implications for the rationality debate? *Behavioral and brain sciences*, 23(5):645–665, 2000.

J. Svegliato and S. Zilberstein. Adaptive metareasoning for bounded rational agents. In *CAI-ECAI Workshop on Architectures and Evaluation for Generality, Autonomy and Progress in AI (AEGAP), Stockholm, Sweden*, 2018.

P. E. Tetlock and B. A. Mellers. The great rationality debate. *Psychological Science*, 13(1):94–99, 2002.

R. H. Thaler and C. R. Sunstein. *Nudge*. Yale University Press, 2021.

The GPyOpt authors. GPyOpt: A Bayesian optimization framework in Python. http://github.com/SheffieldML/GPyOpt, 2016.

P. M. Todd and G. E. Gigerenzer. *Ecological rationality: Intelligence in the world*. Oxford University Press, 2012.

M. S. Tomov, S. Yagati, A. Kumar, W. Yang, and S. J. Gershman. Discovery of hierarchical representations for efficient planning. *PLoS computational biology*, 16(4):e1007594, 2020.

A. Tversky and D. Kahneman. Judgment under uncertainty: Heuristics and biases: Biases in judgments reveal some heuristics of thinking under uncertainty. *science*, 185(4157):1124–1131, 1974.

W. Unanue, M. E. Gómez, D. Cortez, J. C. Oyanedel, and A. Mendiburo-Seguel. Revisiting the link between job satisfaction and life satisfaction: The role of basic psychological needs. *Frontiers in psychology*, 8:239579, 2017.

P. A. Van Lange, J. Joireman, C. D. Parks, and E. Van Dijk. The psychology of social dilemmas: A review. *Organizational Behavior and Human Decision Processes*, 120(2):125–141, 2013.

K. VanLehn. The relative effectiveness of human tutoring, intelligent tutoring systems, and other tutoring systems. *Educational psychologist*, 46(4):197–221, 2011.

R. Vecchio and C. Cavallo. Increasing healthy food choices through nudges: A systematic review. *Food Quality and Preference*, 78:103714, 2019.

O. Vinyals, I. Babuschkin, W. M. Czarnecki, M. Mathieu, A. Dudzik, J. Chung, D. H. Choi, R. Powell, T. Ewalds, P. Georgiev, et al. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, 575(7782):350–354, 2019.

E. Vul, N. Goodman, T. L. Griffiths, and J. B. Tenenbaum. One and done? optimal decisions from very few samples. *Cognitive science*, 38(4):599–637, 2014.

T. T. Wang, A. Gleave, T. Tseng, K. Pelrine, N. Belrose, J. Miller, M. D. Dennis, Y. Duan, V. Pogrebniak, S. Levine, et al. Adversarial policies beat superhuman go ais. In *International Conference on Machine Learning*, pages 35655–35739. PMLR, 2023.

Z. Wang, L. Wang, Z.-Y. Yin, and C.-Y. Xia. Inferring reputation promotes the evolution of cooperation in spatial social dilemma games. *PloS one*, 7(7):e40218, 2012.

J. Wolfe, B. Marthi, and S. Russell. Combined task and motion planning for mobile manipulation. In *Twentieth International Conference on Automated Planning and Scheduling*, 2010.

I. Yaniv and E. Kleinberger. Advice taking in decision making: Egocentric discounting and reputation formation. *Organizational behavior and human decision processes*, 83(2):260–281, 2000.