

**From the holdings of the Department of Special Collections , Princeton University Library,
Princeton, NJ**

These documents are either in the public domain, or can only be used for educational and research purposes (Fair use) as per U.S. Copyright law (text below). By accessing this file, all users agree that their use falls in the public domain or within fair use as defined by the copyright law. We do not charge any permission or usage fees for the publication of images of material in our collections, including those provided by Princeton University Library, via our website, catalog, or directly from staff. For more information regarding copyright, credit, and citation please review the information on the Copyright, Credit and Citation Guidelines page of our website.

<https://library.princeton.edu/special-collections/policies/copyright-credit-and-citation-guidelines>. If you have any questions, including how to order higher-quality images from the collection, contact Special Collections Public Services via our Ask Us! Form. <https://library.princeton.edu/special-collections/ask-us>

Doctoral Dissertations, 2011-2026

Neuroscience

2024

Structured Representations Underlying Efficient Decision-Making

Correa, Carlos Giovanni

Princeton, NJ : Princeton University

<http://arks.princeton.edu/ark:/88435/dsp01pv63g362s>

U.S. Copyright law: The copyright law of the United States (Title 17, §108, United States Code) governs the making of photocopies or other reproductions of copyrighted material. Under certain conditions specified in the law, libraries and archives are authorized to furnish a photocopy or other reproduction. One of these specified conditions is that the photocopy or reproduction is not to be used for any purpose other than private study, scholarship, or research. If a user makes a request for, or later uses, a photocopy or reproduction for purposes in excess of fair use that user may be liable for copyright infringement. This institution reserves the right to refuse to accept a copying order if, in its judgment, fulfillment of the order would involve a violation of copyright law. <https://www.copyright.gov/title17/title17.pdf>

Structured Representations Underlying Efficient Decision-Making

CARLOS GIOVANNI CORREA

A DISSERTATION
PRESENTED TO THE FACULTY
OF PRINCETON UNIVERSITY
IN CANDIDACY FOR THE DEGREE
OF DOCTOR OF PHILOSOPHY

RECOMMENDED FOR ACCEPTANCE
BY THE
PRINCETON NEUROSCIENCE INSTITUTE

ADVISERS: NATHANIEL D. DAW AND THOMAS L. GRIFFITHS

MAY 2024

© COPYRIGHT BY CARLOS GIOVANNI CORREA, 2024. ALL RIGHTS RESERVED.

ABSTRACT

Optimal sequential decision-making requires considering a vast number of potential action sequences, growing exponentially with every step into the future. The theoretical difficulty of decision-making is compounded by the resource limitations faced by humans and other animals, such as finite time and memory. While these limitations are often cited as the reason for suboptimal behavior, the emerging framework of *resource-rationality* adopts the perspective that seemingly-suboptimal behavior is actually adaptively tuned to these limitations and makes rational use of limited resources. This dissertation focuses on two broad approaches humans and other animals take to adaptively simplify their decisions. The first is *hierarchical representation*, intuitively appealing because appropriate choice of hierarchy can decompose complex tasks into simpler subtasks. We develop a resource-rational framework where subgoals are chosen based on how they can simplify the costly process of planning. We identify novel connections between our framework and alternative accounts in simulations, and find that the predictions of our framework are consistent with human behavior in a large-scale behavioral experiment. In a separate study, we run a process-tracing experiment where participants create hierarchically-structured programs and identify a heuristic bias towards reuse that guides hierarchical representations. The second broad approach we consider are *heuristic strategies* for reinforcement learning tasks. We propose a framework for strategy inference, where strategies are formulated as programs and evaluated based on their task performance in addition to their complexity. Focusing on simulations of bandit problems, we examine how strategies vary based on the weight given to complexity and their relationship to behavioral signatures in previous research.

Contents

ABSTRACT	3
1 INTRODUCTION	10
1.1 What makes decision-making difficult?	11
1.2 Planning	12
1.3 Learning	17
1.4 My contributions	21
2 HUMANS DECOMPOSE TASKS BY TRADING OFF UTILITY AND COMPUTATIONAL COST	24
2.1 Introduction	24
2.2 A formal framework for task decomposition	27
2.3 Comparing accounts of task decomposition	35
2.4 An empirical test of the framework	41
2.5 Experiment results	43
2.6 Discussion	52
3 EXPLORING THE HIERARCHICAL STRUCTURE OF HUMAN PLANS VIA PROGRAM GENERATION	58
3.1 Introduction	58
3.2 Background	61
3.3 Theories of program creation in Lightbot	64
3.4 Modeling framework	67
3.5 Experiment	74
3.6 Discussion	85
4 PROGRAM-BASED STRATEGY INDUCTION FOR REINFORCEMENT LEARNING	88
4.1 Introduction	88
4.2 Program-based strategy induction	90
4.3 Exploring strategies for a simple bandit task	94
4.4 Adaptive random exploration	96
4.5 Discrete decision states	99
4.6 Discussion	101
5 CONCLUSION	103
5.1 Hierarchy	103
5.2 Strategies	106

5.3	Future directions	107
APPENDIX A METHODS AND APPENDIX FOR CHAPTER 2		110
A.1	Methods	110
A.2	Extended data	121
A.3	The influence of graph complexity and structure on subgoal choice	125
A.4	Testing whether icons influence participant choice analysis	126
A.5	Predicting navigation response times	126
A.6	A formal relationship between QCut and RRTD-RW	128
A.7	Explaining the relationship between RRTD-RW and Degree Centrality	132
APPENDIX B APPENDIX FOR CHAPTER 3		135
B.1	Algorithms	135
B.2	Trace search	137
B.3	Program generation	137
REFERENCES		139

TO MY MOTHER AND FATHER.
I AM FOREVER GRATEFUL FOR YOUR UNFAILING LOVE AND SUPPORT.

Acknowledgments

I'm incredibly grateful to my advisors, Nathaniel Daw and Tom Griffiths, for their support. While I love many aspects of research, talking with you two about my projects has been a complete joy, consistently illuminating and energizing. Thank you for teaching me about decision-making, guiding me through my confusions, and encouraging me in my moments of doubt. I'm grateful for the close reading you two have given to this work, and how much you've each done to mentor me through the process. Tom: Thank you for your kindness and support during my writer's block, especially with Chapter 2. You've been a helpful guide in developing the habits, acceptance, and fortitude I've needed to write. It has meant the world to me. Nathaniel: I'm glad we were finally able to write Chapter 4, a project that I am so excited about but felt like a pipe dream for years. Thank you for seeing something in that project despite my halting progress, and being exceedingly generous with your time during the writing process.

I have so much gratitude for my collaborators during graduate school. Mark Ho has been a close friend, particularly during the pandemic. Thank you for so many generative conversations! Fred Callaway's clarity of thought and incisive feedback has been integral in ironing out many of the quirks of these projects. Sophia Sanborn was an invaluable guide in trying to wrangle the Lightbot project, from feedback on stimuli to data analysis and interpretation. Sreejan Kumar brought me on for a project that let me learn about program synthesis, which was integral to several subsequent projects. Thanks to all of you for helping me write and think about these projects!

I'm grateful to my thesis committee: Jon Cohen and Yael Niv. In moments I was overwhelmed by the minutiae of my projects, they helped me take a step back, appreciate what I'd done, and bring my projects closer to completion. Thanks to Tim Buschman, as well as Jon and Yael, for being on my exam committee. I'm also grateful to Ken Norman and Jonathan Pillow for their feedback on these projects during my general examination.

I feel so fortunate to have been surrounded by so many bright people in Tom's lab, Nathaniel's lab, and other labs in the department. I've appreciated the feedback and ideas on early versions of these projects, and more generally am grateful for the care, advice, and friendship. One of my favorite things about this line of work is talking to so many sharp people. Thank you all for filling these years with rich and invigorating conversations! Thanks to current and former Daw lab members: Yoel Sanchez Araujo, Flora Bouchacourt, Laura Bustamante, Lindsay Hunter, Ari Kahn, Gili Karni, Rachel Lee, Kevin Lloyd, Marcelo Mattar, Kate Nussenbaum, Payam Piray, Dylan Rich, Harrison Ritz, Yotam Sagiv, Sarah Jo Venditto, Oliver Vikbladh, Sam Zorowitz, and others. Thanks to current and former Griffiths lab members: Mayank Agrawal, Michael Chang, Ishita Dasgupta, Rachit Dubey, Matt Hardy, Robert Hawkins, Ham Huang, Ionatan Kuperwajs, Maya Malaviya, Tom McCoy, Logan Nelson, Evan Russek, Jake Snell, Ilia Sucholutsky, Ted Sumers, Bill Thompson, Bas van Opheusden, Bonan Zhao, Jian-Qiao Zhu, and others. I'm also grateful for the perspectives of Mingbo Cai, Sevan Harootonian, Orren Karniol-Tambour, Qihong Lu, Angela Radulescu, Simon Segert, Mingyu Song, Sina Tafazoli,

and others.

I have extreme gratitude for the administrative staff that keep the department running. I'm especially grateful to Alex Michaud and Ed Clayton for their warmth, patience, and guidance. Their hard work, careful attention, and thoughtfulness have been so helpful in navigating problems and bureaucratic obstacles. Thanks to Shaifali Patel and Jawaan Lowery for helping with my FPO logistics.

I'm grateful to Brian Wandell and Leanne Williams for being my mentors at Stanford. I'm so thankful they hired me and introduced me to life as a scientist. Thanks also to members of their respective labs—the VISTA Lab and Williams PanLab—for reading over my Ph.D. applications and helping me navigate the search for my research interests, with particular thanks to Andrea Goldstein-Piekarski, John Leikauf, Matthew Sacchet, and Brooke Staveland.

I must also thank those who helped me navigate the career transition from software engineering to neuroscience. What felt entirely unclear from the outset was navigable thanks to the helpful perspectives and advice of Ian Ballard, Tyler Bonnen, Kevin Brennan, Jason Laska, Brent Parsons, and Michael Stryker.

I feel indebted to my colleagues and mentors from the years before my scientific career. As a software engineer, I learned so much about careful engineering, effective communication, and time management from Karen Nguyen and David Stryker. Thank you two for being such patient mentors in my first years out of university—while I'm still a bit unruly at times, your lessons are a foundation I always return to. My undergraduate professors at the University of Texas at Austin helped a vague interest in programming grow into a lifelong fascination with computation, which I feel so fortunate to now apply in my research about cognition. Thanks to Alan Cline, William Cook, Don Fussell, Adam Klivans, Calvin Lin, Michael Walfish, and many others.

I'm grateful to my teachers from primary and secondary school. Thanks to Suzanne Hobbs, who supervised my independent study in Java programming. Thanks to my high school history teacher Andrew Martinez and English teacher Lorelei Stenseth for helping me grow as a critical thinker and supporting me in my struggles with writing. To Edward Avila, my band director over many years: A few months into 6th grade band, I can remember a time I was in tears because learning to play bassoon felt too difficult. You kindly encouraged me to keep trying, which led me to so many wonderful musical experiences. Thank you for showing me that frustration is a natural part of learning and daily practice adds up over time.

I'm grateful for the friends I made at Princeton. Thanks to the members of my cohort for the friendship and encouragement. Sandeep, I'm grateful for the time we spent together on walks, organizing visit day, and doing experiments in the teaching lab. Iris, I think neither of us will forget ELE535, thank you for the hours at the whiteboard that made it manageable. Dakota, thank you for my (still-living!) sourdough starter. And also Ivette, Jeremy, Orren, Rebekah, Yotam, Sero: I'm grateful that we could all be companions in grad school! I'm grateful to other friends in and out of the department, with special thanks to Graeme Baker, Rachel Bedder, Oded Bein, Tyler Boyd-Meredith, Alex Libby, Joel Martinez, Gabriel Moore, Wynne Stagnaro, Tim Thijm, Jamal Williams, and Jorge Yanar.

I feel so grateful to my closest friends for sticking around through so many seasons of life. Rachel: Thank you for your enthusiasm and encouragement! I'm so grateful for our years of antics as roommates out west, and glad we have continued to be so close. Andrew and Katherine: The way you two marvel at the world has helped me be more present and curious. Thank you. Peter: Your perseverance

has been so inspiring to me. It has also nourished me in many ways—thank you for book club and your incredible pizza. Thanks to Ria, Claire, Chandler, and Stephen for your caring friendship.

To Emma, my partner: Thank you for supporting me in navigating the ups and downs and listening to my complaints and ramblings about work. Most of all, I'm grateful for the life we share in our beautiful home—it has helped me find peace and stay grounded. Thank you for remembering to buy flowers, waking me with coffee when I've slept in, and putting up with the detritus from my hobbies.

To my family: Thank you for everything. I'm grateful to relatives that made me excited about science and engineering in my youth, particularly Hector and Gus. Julio and Kristina: A great joy of my life is growing together with you two. I'm proud of both of you and grateful that we are all so mutually supportive and loving.

Dad: You have taught me so much about precision, experimentation, and finding fulfillment in creation. Thank you for being the first engineer and scientist in my life. Mom: Thank you for encouraging me to dream big. Thank you for your unfailing commitment to my growth; your positivity and acceptance has helped me push through tough times. Most of all, your vision has always been an inspiration. Both of you two have taught me so much about diligence, which was absolutely necessary when writing this dissertation. I love you both.

Chapter 1

Introduction

Humans and animals are constantly faced with the task of making temporally-extended decisions: from identifying an escape from a looming predator to moving a piece in chess. How do organisms efficiently make these decisions? In the worst case, optimality requires considering candidate sequences of actions, which grow exponentially with every step into the future, incurring considerable computational cost. But instead of exhaustively considering possibilities, organisms often use structured representations like heuristics and hierarchies to support decision-making (Miller et al., 1960; Newell & Simon, 1972). While they can lead to suboptimal choices (Simon, 1955; Kahneman & Tversky, 1979; Botvinick et al., 2009), they offer a simpler alternative to the exponential costs of optimality. Understanding how organisms solve problems efficiently will require studying how they structure and make use of their representations.

In other areas of cognition, a productive approach has been to assume that behavior is well-adapted to the environment, so that identifying the problem faced by an organism and its optimal solution can give insights into behavior (Marr, 1982; Anderson, 1990; Geisler, 2011). However, it is well-known that people make suboptimal decisions, often in systematic ways that deviate from rational theories, like utility theory (Kahneman & Tversky, 1979). How can the apparent conflict between these two perspectives be reconciled: is behavior well-adapted, or a jumble of heuristics? An important link between these two approaches is that the computational difficulty of making decisions could lead organisms to take

simpler, heuristic approaches (Simon, 1955). In recent decades, this approach has been formalized in quantitative theories that predict that behavior aims to achieve high task performance while also minimizing resource use (Todd & Gigerenzer, 2007; Gershman et al., 2015; Lieder & Griffiths, 2020). This resource-rational approach, by formulating choice as a careful trade-off between maximizing utility and minimizing resource costs, can be used to argue that suboptimal behavior is well-adapted. Applying this framework makes it possible to explore how the computational costs of decision-making influence the choice and use of structured representations. In particular, the sensitivity to resource costs means this approach can be used to examine how behavior aligns with strategies and hierarchies that simplify decision-making.

In this thesis, I* will present research on how people and animals use hierarchical representations and strategies to simplify decision-making, by assuming the choice of structured representations is resource-rational. Hierarchy can simplify decisions by fragmenting problems into smaller parts. I will present two projects that examine how people choose hierarchies: first, a resource-rational theory of subgoal choice and analysis of a large-scale experiment and, second, a novel experiment that elicits explicit hierarchical representations. Strategies can simplify decisions by avoiding the consideration of actions, though this requires specifying how strategies are discovered. The final project I present examines how program-structured strategies, optimized to be both simple and effective, can give rise to previously observed behavioral signatures.

1.1 WHAT MAKES DECISION-MAKING DIFFICULT?

Decision-making requires choosing a sequence of actions to take. Naive search for actions is problematic because the number of sequences grows exponentially with the number of steps into the future under consideration, called a “curse of dimensionality” (Bellman, 1957). For example, if you were navigating in a city with a grid layout, you have a choice of four directions at every intersection. If you were

*I will use “I” to denote the author of this dissertation, and “we” when speaking as a member of a collaborative team that carried out some of the research.

to walk 10 blocks, there are $4^{10} \approx 10^6$ possible action sequences you can take. This issue is further exacerbated when the results of actions are uncertain, since all possible outcomes have to be accounted for at every step.

Below, I examine the difficulty of decision-making in two broad categories. The simpler case is solving a problem with full information about it. While this does not automatically lead to a solution, making a decision amounts to planning. The other case is solving a problem that is partly unknown. In this case, the problem becomes more complex because the decision-maker must draw from past experiences in order to learn enough about the task to solve it. In both cases, I give an overview of relevant research in computer science, psychology, and neuroscience, highlighting what makes these settings challenging to solve. I also discuss how organisms can use structure to more efficiently plan and learn. I examine how hierarchy can accelerate planning and simple strategies can provide an alternative to optimal learning. In contrast to approaches that adaptively plan and learn by tuning an algorithm (like pruning search trees in Huys et al., 2012 and tuning learning parameters in Behrens et al., 2007), I focus on how a resource-rational approach can be a guide in a larger space of algorithms.

1.2 PLANNING

Solving a problem with full information can be accomplished with a procedure that is trivial to describe but hard to compute: evaluate all courses of action and choose the best one. To make these ideas more formal, I model a problem as a Markov decision process (MDP; Puterman, 1994), which consists of a state space $\mathcal{S}$, action space $\mathcal{A}$, probability distribution over future states given the current state and action, $p(s_{t+1}|s_t, a_t)$, and the reward received from taking an action, $r(s_t, a_t)$. At every time step t , an agent samples an action a_t from their policy $\pi(a_t|s_t)$, the agent acquires a reward $r(s_t, a_t)$, and the next state of the environment is sampled from $p(s_{t+1}|s_t, a_t)$. The *value*, or expected sum of reward, of

following a policy π when starting at state s_t is

$$V^\pi(s_t) = \mathbb{E}_\pi \sum_{t'=t}^{\infty} r(s_{t'}, a_{t'}) \quad (1.1)$$

$$= \sum_{a_t} \pi(a_t | s_t) \left[r(s_t, a_t) + \sum_{s_{t+1}} p(s_{t+1} | s_t, a_t) \mathbb{E}_\pi \sum_{t'=t+1}^{\infty} r(s_{t'}, a_{t'}) \right] \quad (1.2)$$

$$= \sum_{a_t} \pi(a_t | s_t) \left[r(s_t, a_t) + \sum_{s_{t+1}} p(s_{t+1} | s_t, a_t) V^\pi(s_{t+1}) \right]. \quad (1.3)$$

The value can be expressed as an expectation over trajectories (Eq. 1.1), but can also be recursively defined (Eq. 1.3). In order to ensure the value is finite, I assume the MDP either has a fixed horizon (i.e. states track time and transition to an absorbing state after a fixed number of steps) or is a stochastic shortest path problem (i.e. policies either terminate in an absorbing state after finite time or get infinite negative reward; Bertsekas, 2007). The optimal policy π^* when starting from state s_0 is $\pi^* = \operatorname{argmax}_\pi V^\pi(s_0)$. The value of a policy can be used to construct an improved policy by acting greedily with respect to the value. Policy iteration is an algorithm that iteratively alternates steps of policy evaluation and improvement, which converges to the optimal policy (Sutton & Barto, 2018).

Notably, this recursive definition of a policy's value depends on a key property of MDPs—state transitions only depend on the current state and action so past states and actions can be ignored, called the *Markov property*. This means that the current state of the environment is sufficient for planning, which can vastly simplify decision-making. For example, when navigating, many action sequences can lead to the same location (1 block north then 1 block west leads to the same location as 1 block west then 1 block north). The path you took to reach a location would not influence how you subsequently navigate. Theoretically, the state-based approach to solving MDPs can be simpler than searching for action sequences, as long as action sequences often lead to the same states (as in navigation). Taking advantage of this simplification of decision-making was key to early formulations of sequential decision-making, like dynamic programming (Bellman, 1957) and models of problem solving (Newell & Simon, 1972). However, this approach can fail to simplify problems with large state spaces, if action sequences

rarely lead to the same states. For example, chess has a large number of possible moves, many of which are irreversible or highly constrained, making it rare to reach the same state twice when prospecting about future states.

1.2.1 HUMAN AND ANIMAL PLANNING

Despite the simplifications that a state-based approach to decision-making can provide, it is still difficult for humans and other animals. For example, recent research examining planning in tasks with a small number of states found that humans use a variety of strategies to avoid considering all possibilities, like avoiding action sequences that encounter a large, early loss (Huys et al., 2012), combining independent solutions to smaller subtasks (Huys et al., 2015), or truncating their depth of search with subsequent reliance on value estimates (Keramati et al., 2016). Humans also use these strategies in tasks with larger state spaces (van Opheusden et al., 2023). One explanation for the cognitive difficulty of planning could lie in the limitations of its neural implementation, an issue I explore next.

In rodents, neural activity in the hippocampus typically represents the current location, but when resting, hippocampal neurons can exhibit rapid sequences of activity consistent with non-local navigation trajectories (Diba & Buzsáki, 2007; Skaggs & McNaughton, 1996), a phenomena termed *hippocampal replay*. Some studies have found that replayed trajectories are predictive of subsequent navigation (Pfeiffer & Foster, 2013), suggesting that replayed trajectories can be interpreted as samples of state sequences used for the purpose of planning (Pezzulo et al., 2014; Buzsáki, 2015). In humans, replay has been detected in non-spatial tasks, using magnetoencephalography (MEG; Kurth-Nelson et al., 2016; Liu et al., 2019) and functional magnetic resonance imaging (fMRI; Schuck & Niv, 2019). While replay is faster than physical movement (10–20 times faster in rodents; Buzsáki, 2015), it is still an extremely limited resource compared to the algorithmic costs of planning. Replay also tends to happen during rest, which further restricts this resource. If replay is a basis for planning, then these severe constraints require that it be used adaptively.

1.2.2 EFFICIENT PLANNING WITH HIERARCHY

Breaking a problem down into smaller subproblems is an alternative to solving the whole problem. If these subproblems are jointly easier to solve, then their solutions can be combined to solve the large, original problem. Solving a problem in this way means progressing more rapidly into the depths of the search tree over actions, producing longer action sequences from the temporally abstract choices that combine subproblem solutions. The process is further optimized if subproblems are simpler to solve (like in a fast-growing search tree like chess) or if solutions can simply be reused. However, unhelpful task decompositions can make hierarchical planning more difficult, discussed further below. While the space of possible hierarchies is vast, I focus on two common approaches to constructing hierarchies (McGovern, 2002): using subgoals or repeating fixed action sequences (called *macro-actions* here). Both take larger steps in a search tree, in slightly different ways: planning to pass through a subgoal does not require a concrete plan for how to reach it, while macro-actions generate a stereotyped action sequence without concern for where it will lead.

A prominent approach to formalizing hierarchies for decision-making is the *options* framework (Sutton et al., 1999). An option is a temporally extended action, which is defined by 1) the states it can be initiated in, 2) a policy that is followed while the option is being executed, and 3) the states the option might terminate in. Contemporary research expanded on these ideas, developing approaches for policy learning and problem simplification via state abstraction (Dietterich, 2000), as well as more complex and program-structured behavior (Parr & Russell, 1997). However, these studies typically examined fixed hierarchies, leaving the discovery of hierarchies an open question. Many approaches have been proposed, that depend on commonly-visited states (Şimşek & Barto, 2009), spectral analyses of task structure (Şimşek et al., 2005), and minimizing search algorithm runtime (Jinnai et al., 2019).

For the sake of clarity, I briefly explain how subgoals and macro-actions can be expressed in the options framework. To define an option that reaches a subgoal, it suffices to specify that the option follows a policy that leads to the subgoal and then terminates at the subgoal. To define an option that pro-

duces a macro-action (or fixed sequence of actions) is slightly more complicated, since a macro-action only requires tracking the current position in the action sequence, so environmental state can be ignored. Making the option’s policy history-dependent (in order to track sequence position) is one way to implement this. While I do not further consider them, graph partitions are another common type of hierarchy that can be expressed by defining group-specific options that exit the group and can only be initiated within the group (as in Solway et al., 2014).

Despite its promise, hierarchy can hinder planning in a few ways. First, Botvinick et al. (2009) gives the example of options that are a bad fit for the task at hand, because an agent that uses them would be led further away from their goal. A related issue they note is the possibility of negative transfer, where previously-helpful options would restrict the organism to suboptimal behavior if the environment changes. A second concern is that adding too many options can increase the branching factor of the search tree so much that it obviates the benefit of hierarchy, as noted in Sutton et al. (1999). Besides reducing the number of options, one way to deal with this issue is to limit where options are initiated. For example, options based on graph partitions do so, as noted above. Another approach could automatically identify good initiation states based on state similarity (Castro & Precup, 2012). Because a similar issue arises when there are a large number of actions, inspiration can be drawn from approaches that deal with this. For example, studies have found that humans consider actions that are probable or promising (Mattar & Daw, 2018; Bear et al., 2020), a bias that can be a rational use of limited resources (Lieder et al., 2018). A related issue arises in the similar setting of *library learning*, an approach used to accelerate program synthesis by identifying a library of subprograms to reuse, where a large library is analogous to a large number of options. One approach to overcome this is to train a task- and context-specific neural network to prioritize among subprograms (Ellis et al., 2021).

Hierarchy has long been acknowledged to be an organizing principle in behavior (Miller et al., 1960; Newell & Simon, 1972). We review behavioral research on use of subgoals in Section 2.1 and macro-actions in Section 3.2. Research in the neural basis of hierarchy use in humans has focused on hierarchical learning (Badre & Frank, 2012; Diuk et al., 2013; Collins & Frank, 2016) and planning (Ribas-

Fernandes et al., 2011; Balaguer et al., 2016). In addition, research has noted multi-scale representations of space in the hippocampus in both rodents (Jung et al., 1994) and humans (Brunec et al., 2018), which could be a basis for planning at different scales of temporal abstraction. One theoretical study, which proposes that hippocampal place cells predict future occupancy, explores how predictive representations can be used for hierarchy discovery (Stachenfeld et al., 2017).

1.3 LEARNING

Planning assumes complete knowledge of the MDP (i.e. the transition function p and reward function r), which makes it possible to directly solve for the optimal policy. However, decision-making in an unknown MDP is more difficult because knowledge about the MDP must be gleaned from interactions with the environment. This results in a fundamental conflict between exploiting what has been learned (i.e. select the best option so far) and gathering more information (i.e. exploring options that might be uncertain but promising), termed the *explore/exploit dilemma* (Sutton & Barto, 2018). Within a fixed-length episode, it is clear that exploration is necessary early on (when much is unknown) and exploitation is rational later on (when new information would be of little help), but shifting between the two in a rational way is difficult. When learning, even simple problems become surprisingly difficult. For example, a *k-armed bandit problem* is a stateless task where there are k actions (called arms) that each yield action-specific, but random, rewards. If the distribution of rewards for each arm were known, the optimal choice would be simple: pick the arm with the greatest expected reward. However, since the rewards are unknown, it is necessary to sample the available options and learn based on observed rewards.

1.3.1 TWO WAYS TO LEARN

I will distinguish two approaches for learning, following Watkins (1989). The first is *optimal learning*, which maximizes the expected reward acquired over the entire course of learning (i.e. summed over all episodes). This requires considering all possible future outcomes and actions for all episodes, which grows exponentially with the number of episodes and episode length. The second approach is *learning*

an efficient strategy, by drawing from past experience to asymptotically learn an optimal policy, instead of maximizing all experienced rewards. In contrast to optimal learning, this approach only ensures that the optimal policy is eventually learned, without optimizing the rate of convergence or acquired reward along the way.

I briefly sketch out a solution for optimal learning, the first approach. While the agent is solving an MDP, the underlying transition and reward functions are unknown. So, optimal learning requires maintaining beliefs about the underlying MDP and performing a Bayesian update of beliefs with every new observation. These beliefs are a sufficient statistic of past experiences and can be used as a representation to solve the task by planning, but the problem remains intractable (Duff, 2002). Optimal learning can be modeled as a Bayes-adaptive MDP (Duff, 2002) or task inference (Humplik et al., 2019), though more complex cases where the task can change (like bandit problems with non-stationary rewards) correspond to the larger problem class of partially observable MDPs (Kaelbling et al., 1998). While there are problem-specific algorithms that are tractable (like index-based policies introduced by Gittins, 1979 for discounted, infinite-horizon bandit problems), in general these problems are often intractably difficult to solve.

Recent research has identified *meta-reinforcement learning* (meta-RL) as a tractable approach for approximate optimal learning (Duan et al., 2016; Wang et al., 2018). Meta-RL specializes the broader concept of meta-learning (Thrun & Pratt, 1998), which learns a learning algorithm through two nested learners: an inner learner that simply solves the task at hand and an outer meta-learner that improves the inner learner, guided by the inner learner’s task performance. Meta-RL typically uses a recurrent neural network (RNN) as an inner learner, with recurrent dynamics that can incorporate past experience. After meta-learning by way of gradient descent, this method can produce learners that solve the explore/exploit dilemma, dynamically tune learning parameters, and use inference-based strategies (Wang et al., 2018). Importantly, meta-RL can be seen as a sampling-based method for approximating optimal learning, as noted in theoretical (Ortega et al., 2019) and simulation-based research (Mikulik et al., 2020).

I turn to the second approach for learning, where an optimal policy can eventually be learned. Many algorithms focus on sample-based estimation of the value of a policy, using either samples of entire trajectories (like in Eq. 1.1) or, for greater statistical efficiency, using samples of state transitions (relying on existing estimates for successor states, resembling Eq. 1.3). Choice is guided by value estimates, but usually incorporates noise to ensure exploration. By relying on samples, this strategy does not need knowledge about the task to be known or inferred. As an example, the SARSA algorithm (Sutton & Barto, 2018) estimates the expected future reward, $Q(s, a)$, resulting from taking action a in state s . Having taken action a in state s , received reward $r(s, a, s')$, and then subsequently taken action a' in state s' , estimates are updated with a prediction error, δ , between the current estimate of the future reward and a new sample of the reward,

$$\delta = r(s, a, s') + Q(s', a') - Q(s, a) \quad (1.4)$$

$$Q(s, a) \leftarrow Q(s, a) + \alpha \delta. \quad (1.5)$$

Instead of the average of all previous samples, this update computes an exponentially-weighted average with greater weight on more recent samples, which supports learning in non-stationary environments. The learning rate $\alpha \in [0, 1]$ interpolates between the existing estimate and new sample, with greater α placing greater weight on recent samples up to $\alpha = 1$ which simply memorizes the most recent sample. Under appropriate conditions, SARSA converges to optimal behavior. But, as noted above, these algorithms do not in general perform optimal learning across all episodes.

1.3.2 HUMAN AND ANIMAL LEARNING

Interestingly, models based on prediction errors have a long history in the study of animal learning. One prominent account by Rescorla & Wagner (1972) used prediction errors to account for many puzzling phenomena in animal learning, which were well-explained by assuming that surprising outcomes drive learning. The activity of dopaminergic neurons has also been found to be correlated to prediction

errors (Montague et al., 1996; Niv, 2009), providing algorithmic evidence in support of error-driven learning. Variants of these models have been more closely related to the circuitry of the basal ganglia (Niv, 2009; Collins & Frank, 2014) and more recently adapted to account for heterogeneity in large-scale recordings of dopaminergic activity (Lee et al., 2024; Dabney et al., 2020).

Despite the strong evidence that human and animal learning is based on prediction errors, behavior often exhibits signatures of better-adapted strategies. One example is dynamic adjustment of learning algorithm parameters, like more decision noise when there is more time to learn (Wilson et al., 2014) or higher learning rates when the environment is volatile (Behrens et al., 2007). Other studies suggest varied estimation strategies that depend on different memory systems, like drawing from specific episodic memories in estimation (Duncan & Shohamy, 2016) or using working memory to rapidly learn associations (Collins & Frank, 2012). There is also evidence of greater deviations from error-driven learning, in order to support increasingly optimized behavior (Collins, 2018). For example in sequential tasks with well-learned transition structure, an organism can achieve greater reward by updating estimates at nonlocal states (which resembles planning) rather than simply doing so at the current state (Daw et al., 2005; Mattar & Daw, 2018), signatures of which have been found in behavior and neural data of humans and rodents (Daw et al., 2011; Miller et al., 2017; Liu et al., 2021). In tasks with more complex structure, there is evidence of well-adapted strategies, like sharing value estimates between related actions (Niv et al., 2015), inference of reversals in task rewards (Hampton et al., 2006; Bartolo & Averbeck, 2020), and more complex inference about hidden task states (Schuck et al., 2016; Starkweather et al., 2017; Bartolo & Averbeck, 2021). Collectively, these findings suggest that organisms go beyond simple estimation strategies based on error-driven learning, in order to optimize their behavior.

1.3.3 EFFICIENT LEARNING WITH STRATEGIES

Instead of directly solving optimal learning, I examine how strategies that generate actions based on past experiences can present simpler alternatives to decision-making. In a task like a two-armed bandit, one simple strategy is win-stay, lose-shift (WSLS; Robbins, 1952), which selects an action that wins

and switches to the other after a loss. In contrast to hierarchy, strategies do not make search easier—instead, they are an alternative algorithm to planning that is simpler to compute, and can be viewed as a compact representation of history-dependent behavior.

Research has found that human strategy use is resource-rational, displaying a sensitivity to cognitive resource use (Payne et al., 1988; Rieskamp & Otto, 2006; Lieder & Griffiths, 2017). Recent studies have used this insight for strategy discovery by augmenting sequential decision problems with cognitive operations and costs (called metalevel MDPs) and have found strategies that resemble human planning (Callaway et al., 2022) and risky choice (Lieder et al., 2017). Unfortunately, metalevel MDPs are often challenging to solve directly and result in policies that are unstructured, requiring manual effort to interpret (though an automatic approach was proposed in Skirzyński et al., 2021). Meta-RL is another approach to strategy discovery (further discussed in Section 4.1), which some research has augmented with resource costs, like penalties on network complexity (Binz & Schulz, 2022; Moskovitz et al., 2023). It’s also worth noting that bounds on the space of policies (i.e. a limit on the number of units in an RNN) can make the optimal policy impossible to express and can be viewed as a kind of resource limitation.

1.4 MY CONTRIBUTIONS

In this dissertation, I explore how structured representations are used to mitigate the difficulty of decision-making. In particular, my approach is to formulate the choice of structures as guided by resource-rationality, ensuring that task performance is maximized while resource use is minimized. In two studies, I investigate how people simplify planning by using hierarchy and, in a final simulation-based study, examine how strategies selected for their simplicity recapitulate previously observed behavioral signatures of learning.

In Chapter 2 I will present research where we propose and evaluate a normative framework for task decomposition based on the simple idea that people decompose tasks to reduce the overall cost of plan-

ning while maintaining task performance. Analyzing 11,117 distinct graph-structured planning tasks, we find that our framework justifies several existing heuristics for task decomposition and makes predictions that can be distinguished from two alternative normative accounts. I report a behavioral study of task decomposition ($N = 806$) that uses 30 randomly sampled graphs, a larger and more diverse set than that of any previous behavioral study on this topic. We find that human responses are more consistent with our framework for task decomposition than alternative normative accounts and are most consistent with a heuristic—betweenness centrality—that is justified by our approach.

Motivated by the observation that theories of hierarchy are often difficult to dissociate, in Chapter 3 I present a study where we explore how people form hierarchically-structured plans, using an experimental paradigm that makes hierarchical representations observable: participants create programs that produce sequences of actions in a language with explicit hierarchical structure. This task lets us test two well-established principles of human behavior: utility maximization (i.e. using fewer actions) and minimum description length (MDL; i.e. having a shorter program). We find that humans are sensitive to both metrics, but that both accounts fail to predict a qualitative feature of human-created programs, namely that people prefer programs with *reuse* over and above the predictions of MDL. We formalize this preference for reuse by extending the MDL account into a generative model over programs, modeling hierarchy choice as the induction of a grammar over actions. Our account can explain the preference for reuse and provides the best prediction of human behavior, going beyond simple accounts of compressibility to highlight a principle that guides hierarchical planning.

Changing focus to resource-rational learning, in Chapter 4 I present research motivated by understanding how organisms might learn strategies that are both simple and effective. Despite recent advances in strategy discovery using tools like recurrent networks that generalize the classic models, the resulting strategies are often onerous to interpret, making connections to cognition difficult to establish. We use Bayesian program induction to discover strategies implemented by programs, letting the simplicity of strategies trade off against their effectiveness. Focusing on bandit tasks, we find strategies that are difficult or unexpected with classical incremental learning, like asymmetric learning from rewarded and

unrewarded trials, adaptive horizon-dependent random exploration, and discrete state switching.

Chapter 2

Humans decompose tasks by trading off utility and computational cost *

2.1 INTRODUCTION

Human thought and action are hierarchically structured: We rarely tackle everyday problems in their entirety and instead routinely decompose problems into more manageable subproblems. For example, you might break down the high-level goal of “cook dinner” into a series of intermediate subgoals such as “choose a recipe,” “get the ingredients from the store,” and “prepare food according to the recipe.” Task decomposition—identifying subproblems and reasoning about them—lies at the heart of human general intelligence. It allows people to tractably solve problems that occur at many different timescales, ranging from everyday tasks such as cooking a meal to more ambitious projects such as completing a Ph.D.

At least two questions arise in the context of human task decomposition. First, how do people use

*This chapter is based on the following published article:

Correa, C. G., Ho, M. K., Callaway, F., Daw, N. D., & Griffiths, T. L. (2023a). Humans decompose tasks by trading off utility and computational cost. *PLOS Computational Biology*, 19(6), 1–31

Which itself was drawn from a conference proceedings:

Correa, C. G., Ho, M. K., Callaway, F., & Griffiths, T. L. (2020). Resource-rational task decomposition to minimize planning costs. In *Proceedings of the 42nd Annual Conference of the Cognitive Science Society*

decompositions? Second, how do people decompose tasks to begin with? Existing research provides answers to these two questions, but does so largely by considering each one in isolation. For example, we know that, when given hierarchical structure, people readily use it to bootstrap learning (Botvinick et al., 2009; Eckstein & Collins, 2020) and to organize planning (Balaguer et al., 2016; Cushman & Morris, 2015). Separately, studies show how hierarchical structure emerges from graph-theoretic properties of tasks (e.g., “bottleneck” states in Stachenfeld et al., 2017), latent causal structure in the environment (Collins & Frank, 2013; Tomov et al., 2020), or efficient encoding of optimal behaviors (Solway et al., 2014). These accounts provide insights into the function and mechanisms of hierarchically structured, action-guiding representations, but, again, they largely consider *the use* and *the creation* of such representations separately.

In this chapter we bridge this gap, developing an integrated account of how using a decomposition interacts with the task decomposition process itself. Our proposal is organized around a deceptively simple idea: Task decompositions are learned to facilitate efficient planning (Fig. 2.1). Based on this intuition, we develop a normative framework that specifies how an idealized agent should choose a hierarchical structure for a domain, given the need to balance task performance with the costs of planning. We quantify planning costs straightforwardly as the run-time of a planning algorithm, which means that our framework predicts that task decomposition is the result of interactions between task structure and the algorithm used to plan. Because we quantitatively examine how cognitive costs are balanced with task performance in the style of a *resource-rational analysis* (Lewis et al., 2014; Gershman et al., 2015; Griffiths et al., 2015), we refer to our framework as *resource-rational task decomposition*.

Although much prior research is motivated by the idea that hierarchical task decomposition has the potential to reduce planning costs (Sacerdoti, 1974; Korf, 1985b; Sutton et al., 1999; Şimşek et al., 2005; Ramkumar et al., 2016; Solway et al., 2014; Huys et al., 2015; Jinnai et al., 2019), our framework differs from some prominent accounts because we directly incorporate planning costs into the criteria used to choose a task decomposition. By contrast, existing normative accounts typically formulate task decomposition as *structure inference* with the goal of inferring the hierarchical structure of the

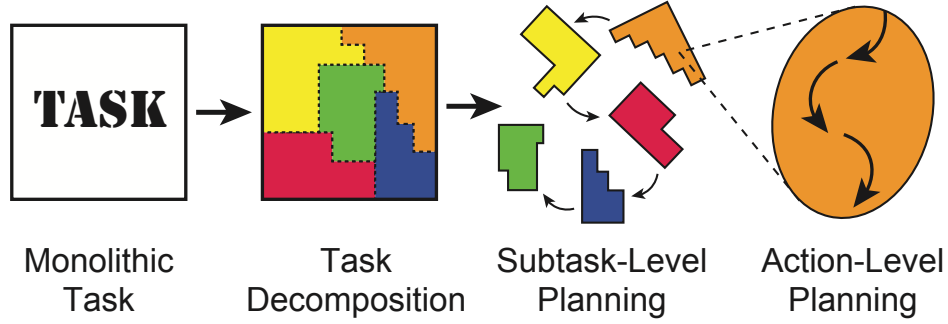


Figure 2.1: Our framework for task decomposition accounts for the computational cost of planning towards subgoals—task decompositions should jointly optimize task performance and the computational cost of search. We formalize this in three nested layers of optimization: Action-Level Planning solves for a plan to accomplish a subgoal, which has a computational cost. Subgoal-Level Planning constructs subgoal sequences that maximize reward and minimize computational cost. Task Decomposition selects subgoals based on their value in Subgoal-Level Planning. This figure was adapted from a figure published in [Correa et al. \(2020\)](#); License: CC BY 4.0.

environment ([Collins & Frank, 2013](#); [Tomov et al., 2020](#)) or sequential behavior ([Solway et al., 2014](#); [Maisto et al., 2015](#)), which only indirectly connect to the computations involved in planning or their efficiency. Instead, our formal framework extends research that performs task decomposition based on algorithm-specific planning costs—some algorithms previously studied are value iteration ([Jinnai et al., 2019](#)), random walk search ([McNamee et al., 2016](#)), and random sampling of optimal behavior ([Solway et al., 2014](#)). Generalizing beyond a fixed algorithm, our framework explicitly considers how planning efficiency shapes hierarchical representations, which we use to demonstrate how resource-rational task decompositions change with varied search algorithms. Additionally, normative accounts often have limited ability to explain human behavior without the specification of algorithmic details necessary to be efficient or psychologically plausible. Because our account focuses on efficient use of planning, it is even more critical to spell out these details. We conduct an initial exploration of these issues by examining the capacity of existing heuristics to serve as efficient algorithmic approximations to our normative account.

Using this framework, we conduct a systematic comparison of resource-rational task decomposition with four alternative formal models previously reported in the literature ([Şimşek et al., 2005](#); [Şimşek & Barto, 2009](#); [Solway et al., 2014](#); [Tomov et al., 2020](#)) using 11,117 different graph-structured planning

tasks. One key insight from this analysis is that our framework can justify several previously proposed *heuristics* for task decomposition based on graph-theoretic properties (e.g., those that capture the idea of “bottleneck states”; Şimşek & Barto, 2009; Şimşek et al., 2005). Our framework thus provides a normative justification for these heuristics within a broader framework of resource-rational decision-making. Critically, these connections between our framework and heuristics also demonstrate the existence of efficient approximations to our formal framework. We also show that our framework produces predictions that are distinguishable from previous normative models of task decomposition.

To empirically evaluate this framework, we report results from a pre-registered experiment ($N = 806$) that uses 30 distinct graph-structured tasks sampled from 1,676 graphs, the subset of the 11,117 graphs that are compatible with our experimental design. To our knowledge, this set of graph-structured tasks is larger and more diverse than that of any other experiment previously reported in the literature on how people decompose tasks. As such, it enables us to draw more general conclusions about task decomposition than previous studies. Across this large stimulus set, we find that our framework provides the best explanation for participant responses among normative accounts, which supports the thesis that people’s hierarchical decomposition of tasks reflects a rational allocation of limited computational resources in service of effective planning and acting.

2.2 A FORMAL FRAMEWORK FOR TASK DECOMPOSITION

How should an agent decompose a task? When purely optimizing behavior on a task (e.g., taking a shortest path in a graph), decomposing a task is only worthwhile in some larger context, such as making learning or computation more efficient. The computational efficiency of planning is a critical concern – as one attempts to plan into an increasingly distant future, over a larger state space, or under conditions of greater uncertainty, computation quickly becomes intractable, a challenge termed the *curse of dimensionality* (Bellman, 1957). In some cases, task decomposition can ameliorate this curse by splitting a task into more manageable subtasks. The difficulty comes in choosing among task decomposi-

tions since a bad choice can make the task at hand even more difficult (Botvinick et al., 2009). In this work, we formulate a framework for task decomposition where *planning costs* directly factor into people’s choices—quite literally, our framework decomposes tasks into subtasks based on the run-time and utility of the plan that results from planning algorithms that solve the subtasks.

To demonstrate the role planning costs play in how people break down tasks, consider the following scenario: After leaving work for the day, you plan to go to the post office to send a letter. Since you rarely navigate directly from your workplace to the post office, you’ll have to do some planning. You could determine some efficient way to get from work to the post office, but an alternative is to first get somewhere that is easy to navigate to and also along the way. Maybe the café you sometimes stop by before work—If it’s easier to plan a route from the café to the post office, then you’ve simplified your problem by breaking it down into subtasks. This example suggests that the way people break tasks down (e.g., navigating to the café first) is a trade-off between efficiency (e.g., taking a quick route) and the cost of planning with that task decomposition (e.g., planning via the café is simpler).

We formalize our framework using three nested levels of planning and learning (Fig. 2.1). At the lowest level is *action-level planning*, where concrete actions are chosen that solve a subtask (e.g., what direction do I walk to get to the café). The next level is *subtask-level planning*, where a sequence of subtasks is chosen (e.g., first navigating to the café and then to the post office). Finally, the highest level is *task decomposition*, where a set of subtasks that break up the environment are selected (e.g., setting the café as a possible subgoal across multiple tasks).

A central feature of our framework is the interdependence of choices made at each level: The optimal task decomposition depends on the computations occurring in the subtask-level planner, which depends on the computations that occur in the action-level planner. In particular, we are interested in how different decompositions can be evaluated as better or worse based on the cost of computing good action-level plans for a series of subtasks chosen by the subtask-level planner. In the next few sections, we discuss these different levels and how they relate to one another.

2.2.1 ACTION-LEVEL PLANNING

Action-level planning computes the optimal actions that one should take to reach a subgoal. Here, we focus on deterministic, shortest-path problems. Formally, action-level planning occurs over a task $(\mathcal{S}, T, s_0, z)$ defined by a set of states, $\mathcal{S}$; an initial state, $s_0 \in \mathcal{S}$; a subgoal state, $z \in \mathcal{S}$; and valid transitions between states, $T \subseteq \mathcal{S} \times \mathcal{S}$, so that s can transition to s' when $(s, s') \in T$. The neighbors of s are the states s' that it can transition to, $\mathcal{N}(s) = \{s' \mid (s, s') \in T\}$. We refer to the structure of a task $(\mathcal{S}, T)$ as the task environment or the task graph.

Given an initial state, s_0 , and a subgoal, z , action-level planning seeks to find a sequence of states that begins at s_0 and ends at z , which we denote as a plan $\pi = \langle s_0, s_1, \dots, z \rangle$. An action-level plan is computed by a *planning algorithm*, which is a stochastic function that takes in the initial state, valid transitions, and subgoal state and takes a certain amount of time to run, t . Thus, we can think of a planning algorithm Alg as inducing a distribution over plans and run-times given a start and end state, $P_{\text{Alg}}(\pi, t \mid s, z)$. In this work, we considered four different planning algorithms (Russell & Norvig, 2021; Ghallab et al., 2016), which we summarize below.

We start with a simple algorithm that hardly seems like one—the *random walk (RW) algorithm*. The algorithm starts at the initial state s_0 and repeatedly transitions to a uniformly sampled neighbor of the current state until it reaches the subgoal state z . Because it does not keep track of previously visited states to inform state transitions, this algorithm can revisit states many times and can result in both path lengths and run-times that are unbounded.

Depth-first search (DFS) augments RW by keeping track of the states along its current plan—this helps minimize repeated state visits. Because DFS does this, it will sometimes reach a dead-end where it is unable to extend the current plan, so it backtracks to an earlier state to consider an alternative choice among the other neighbors. DFS ensures that resulting plans avoid revisiting states, but still might be suboptimal and repeatedly consider the same states during the search process.

Iterative Deepening Depth-first Search (IDDFS; Korf, 1985a) consists of depth-limited DFS run to

increasing depths until the goal is found; while based on DFS, IDDFS returns optimal paths because it systematically increases the depth limit. IDDFS is conceptually similar to “progressive deepening,” a search strategy proposed by de Groot in seminal studies of chess players (De Groot, 1965; Newell & Simon, 1972).

Breadth-first search (BFS) ensures optimal paths by systematically exploring states in order of increasing distance from the start state s_0 . The algorithm does so by considering all neighbors of the start state (which are one step away), then all of their unvisited neighbors (which are two steps away), and so on, successively repeating this process until the goal is encountered. Through this systematic process, BFS is able to guarantee optimal solutions and ensure states will only ever be considered once, making the algorithm run-time linear in the number of states.

While only noted in passing above, each of these algorithms makes subtle trade-offs between run-times, memory usage, and optimality. Focusing on BFS and IDDFS, the two optimal algorithms, we briefly examine these trade-offs. BFS visits states at most once, but requires remembering every previously visited state; by contrast, IDDFS will revisit states many times (i.e. greater run-time compared to BFS) but only has to track the current candidate plan (i.e. smaller memory use compared to BFS). In effect, IDDFS increases its run-time to avoid the cost of greater memory use. We briefly return to this point below when examining algorithm run-times. Having introduced various search algorithms, we now turn to the role algorithms play in subtask-level planning, where algorithm plans and run-times jointly influence the choice of hierarchical plan.

2.2.2 SUBTASK-LEVEL PLANNING

Here, we assume a simplified model of hierarchical planning that involves only a single level above action-level planning, which we call *subtask-level planning*. Formally, subtask-level planning occurs over a set of subgoals, $\mathcal{Z} \subset \mathcal{S}$. Given a set of subgoals, subtask-level planning consists of choosing the best sequence of subgoals that accomplish a larger aim of reaching a goal state $g \in \mathcal{S}$. Each subgoal is then provided to the action-level planner, and the resulting action-level plans are combined into a

complete plan to reach the goal state.

The objective of the subtask-level planner is to identify the sequence of subgoals that brings the agent to the goal state while maximizing task rewards and minimizing computational costs. Here, we focus on a domain in which the task is simply to reach the goal state in as few steps as possible. Formally, the task reward associated with executing a plan is then simply the negative number of states in that plan: $R(\pi) = -|\pi|$. The computational cost that we consider is *cumulative expected run-time*. Thus, we define a subgoal-level reward function when planning to a single subgoal z from a state s using a planning algorithm that induces a distribution over plans and run-times $P_{\text{Alg}}(\pi, t \mid s, z)$ as:

$$R_{\text{Alg}}(s, z) = \sum_{\pi, t} P_{\text{Alg}}(\pi, t \mid s, z) [R(\pi) - t]. \quad (2.1)$$

This formulation is analogous to other resource-rational models that jointly optimize task rewards and run-time (Lieder et al., 2014; Callaway et al., 2022) but applied to the problem of task decomposition.

Eq. 2.1 defines the rewards for planning towards a single subgoal, but subtask-planning requires chaining plans together to form a larger plan that efficiently solves the task. This sequential optimization problem can be compactly expressed as a set of recursively defined Bellman equations (Bellman, 1957). Formally, given a task goal g , a set of subgoals $\mathcal{Z}$, and an algorithm Alg , the optimal subtask-level planning utility for all non-goal states s is then:

$$V_{\mathcal{Z}}^g(s) = \max_{z \in \mathcal{Z} \cup \{g\}} \{R_{\text{Alg}}(s, z) + V_{\mathcal{Z}}^g(z)\} \quad (2.2)$$

To ensure this recursive equation terminates, the utility of the goal state g is $V_{\mathcal{Z}}^g(g) = 0$. The fixed point of Eq. 2.2 can be used to identify the optimal subtask-level policy (Puterman, 1994). We permit the selection of the goal g as a subgoal to ensure that it is possible for the subtask-level planner to solve the task.

2.2.3 TASK DECOMPOSITION

Having defined action-level planning and subtask-level planning over subgoals, we can now turn to our original motivating question: How should people decompose tasks? In this context, this reduces to the problem of selecting the best set of subgoals $\mathcal{Z}$ to plan over. Importantly, we assume that people rely on a common set of subgoals for all the different possible tasks that they might have to accomplish in a given environment. Thus, the value of a task decomposition, $\mathcal{Z}$, is given by the value of the subtask-level plans averaged over the task distribution of an environment, $p(s_0, g)$. That is,

$$V(\mathcal{Z}) = \sum_{s_0, g} p(s_0, g) V_{\mathcal{Z}}^g(s_0). \quad (2.3)$$

The optimal set of subgoals $\mathcal{Z}^*$ for planning maximize this value, so $\mathcal{Z}^* = \operatorname{argmax}_{\mathcal{Z}} V(\mathcal{Z})$.

To summarize, the value of a task decomposition (Eq. 2.3) depends on how a subtask-level planner plans over the decomposed task (Eq. 2.2), which is shaped by the resulting plans and run-time of action-level planning (Eq. 2.1). This model thus captures how several key factors shape task decomposition: the structure of the environment, the distribution of tasks given by an environment, and the algorithm used to plan at the action level.

To provide an intuition for our framework, we explore its predictions in a simple task in Fig. 2.2. The environment is a grid with a single task that requires navigating from the green state to the orange state. Each column in the figure corresponds to a different search algorithm, showing how search costs change without subgoals (Top) and with a subgoal (Bottom; subgoal in blue). While the task seems like it would be extraordinarily trivial to a person—like walking from one side of a room to another—a critical attribute of these search algorithms is they have an entirely unstructured representation of the environment, giving them only very local visibility at a state. A more analogous task for a person might be navigating in a place with low visibility, such as a forest or a city in a blackout. Even in this simple task, some search algorithms (BFS and IDDFS) can be used more efficiently when the problem is split at its midpoint (Fig. 2.2d). The random walk is a notable counterexample, where using a subgoal results in

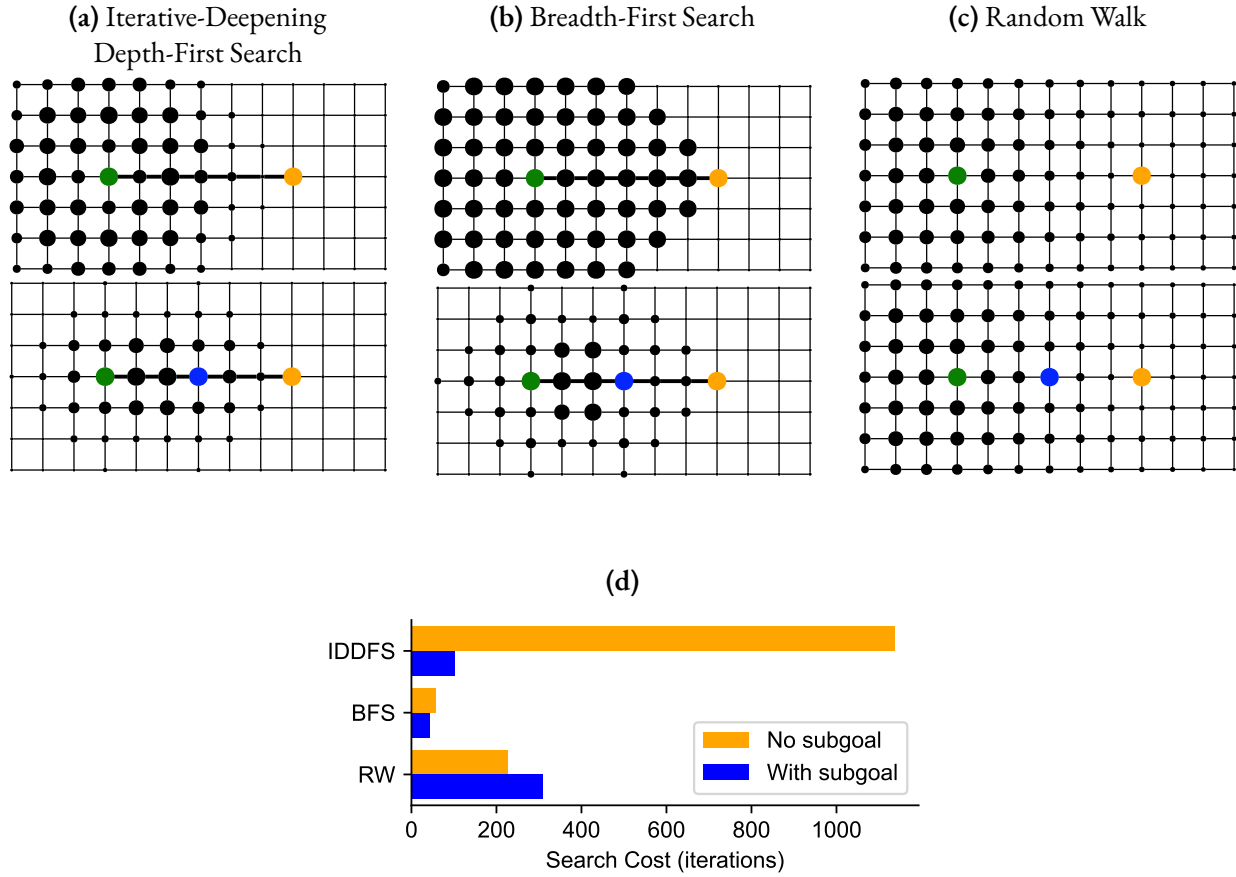


Figure 2.2: Choosing task decompositions that make planning more efficient. (a-c) State-specific search costs of the algorithms. The depicted task requires navigating on a grid from the start state (green) to the goal state (orange) with the fewest steps. Each column corresponds to a different algorithm and demonstrates two scenarios—*Top*: Search cost without subgoals, *Bottom*: Search cost when using the path midpoint as a subgoal (blue). We define the search cost as the number of iterations required for the search algorithm to find a solution. Larger states were considered more often during the search algorithm, resulting in greater search costs. (d) Plot of search cost for Iterative-Deepening Depth-First Search (IDDFS), Breadth-First Search (BFS), and a Random Walk (RW) in the task depicted in (a-c), with and without subgoals. Use of subgoals results in decreased search cost for BFS and IDDFS, but not for RW.

less efficient search. This result may initially seem puzzling, but occurs because a random walk is likely to get to the goal without passing through the subgoal. The gap in run-time between IDDFS and BFS might make IDDFS seem inefficient—however, as noted in the algorithm descriptions above, IDDFS makes a trade-off of increased run-time in order to decrease memory usage. While outside the scope of the current chapter, our formulation can be extended to incorporate other resource costs like memory

usage in order to study how they influence task decomposition. This example clearly demonstrates a few characteristics of our framework—that the choice of hierarchy critically depends on the algorithm used for search, and that hierarchy can have a normative benefit (since it reduces computational costs) even in the absence of learning or generalization.

The formal presentation of our framework considers subgoal choice with intentionally restricted algorithms: brute-force search methods that exclude problem-specific heuristics to accelerate planning. However, the examples in this section (navigating to the post office via the café, navigating in a place with low visibility) likely rely on algorithms that incorporate heuristics, particularly related to spatial navigation. While outside the scope of this chapter, our framework can flexibly incorporate any search algorithm that can define an algorithmic cost, including those that make use of heuristics. For example, in a previous theoretical study, we applied an early version of our framework to task decomposition in the Tower of Hanoi by using A* Search (Ghallab et al., 2016) with an edit distance heuristic (Correa et al., 2020). Our framework also considers a constrained set of task decompositions that consist of individual subgoals. In comparison, the influential *options framework* (Sutton et al., 1999) defines a more general set of hierarchical task decompositions, in which, for instance, subtasks can be defined by the subgoal of reaching any one out of a set of states and subtask completion can be non-deterministic. However, finding such subtasks remains a challenging problem in machine learning, so by focusing on the simpler problem of selecting a single subgoal we are able to make a significant amount of progress in understanding a key component of how humans plan. While not yet fully explored, our formalism can be extended to encompass broader types of hierarchy (and also varied search algorithms). For instance, another theoretical study adapted our framework to support more varied kinds of hierarchical structure by incorporating abstract spatial subgoals in a block construction task (Binder et al., 2021).

2.3 COMPARING ACCOUNTS OF TASK DECOMPOSITION

A number of existing theories have been proposed for how people decompose tasks. These accounts can be divided into two broad categories: *heuristics* for decomposition based on graph-theoretic properties of tasks and *normative* accounts based on the functional role of a decomposition. Our account is normative, so comparison with alternative normative accounts highlights the unique functional consequences of our framework. By comparing our framework to heuristics, we can characterize their predictions relative to normative theories as well as rationalize their use as heuristics for task decomposition. All models are listed and briefly described in Table 2.1.

To begin with, one way to compare accounts is to relate them formally. We do so by relating resource-rational task decomposition with a random walk (RRTD-RW) to QCut (Şimşek et al., 2005; Menache et al., 2002) and Degree Centrality in Appendix A. We prove a relationship between RRTD-RW and QCut that connects the two methods through spectral analysis and examine how the relationship between RRTD-RW and Degree Centrality varies based on spectral graph properties.

Another method we can use to compare theories is to compute their subgoal predictions on a fixed set of environments and compare them—qualitatively or quantitatively. This approach has been used in existing studies, but nearly always through qualitative comparison using a small number of hand-picked environments. For example, several published models perform qualitative comparisons to the graphs studied in Solway et al. (2014). These environments contain states that most models agree should be subgoals (Tomov et al., 2020; McNamee et al., 2016; Donnarumma et al., 2016; Correa et al., 2020)—these states are typically one or few that connect otherwise disconnected parts of the environment, making them “bottleneck states.” Environments with these kinds of bottleneck states robustly elicit hierarchically-structured behavior in experiments, but make it difficult to distinguish among theoretical accounts because they are in strong agreement (see top row of Fig. 2.3).

To perform a large-scale and unbiased comparison of these algorithms, we chose from a structurally rich set of environments: the set of all possible 11,117 simple, undirected, eight-node, connected graphs.

Table 2.1: Descriptions of Normative Algorithms and Heuristics

Normative Algorithm	Description
RRTD-IDDFS	Resource-Rational Task Decomposition (RRTD) using Iterative-Deepening Depth-First Search (IDDFS) as a search algorithm
RRTD-BFS	RRTD using Breadth-First Search (BFS) as a search algorithm
RRTD-RW	RRTD using a Random Walk (RW) as a search algorithm
Solway et al. (2014)	Identifies partitions of the task into sub-tasks that minimize the description length of optimal solutions, given that subtask solutions are reused across tasks.
Tomov et al. (2020)	Performs inference over partitions of the task graph into regions based on a prior over hierarchical graphs. Incorporates a preference for tasks to start and end in the same region, and for states in the same region to have similar rewards.
Heuristic	Description
QCut (Şimşek et al., 2005; Menache et al., 2002)	Partitions the task graph through spectral decomposition of the graph.
Degree Centrality	Chooses subgoals based on Degree Centrality, which is the number of transitions into or out of a state s . For tasks where all state transitions are reversible, Degree Centrality is the number of neighbors $ \mathcal{N}(s) $.
Betweenness Centrality (Şimşek & Barto, 2009)	Chooses subgoals based on Betweenness Centrality, which is how often a state s appears on shortest paths, averaged over all possible start and goal states. Takes into account cases with multiple shortest paths.

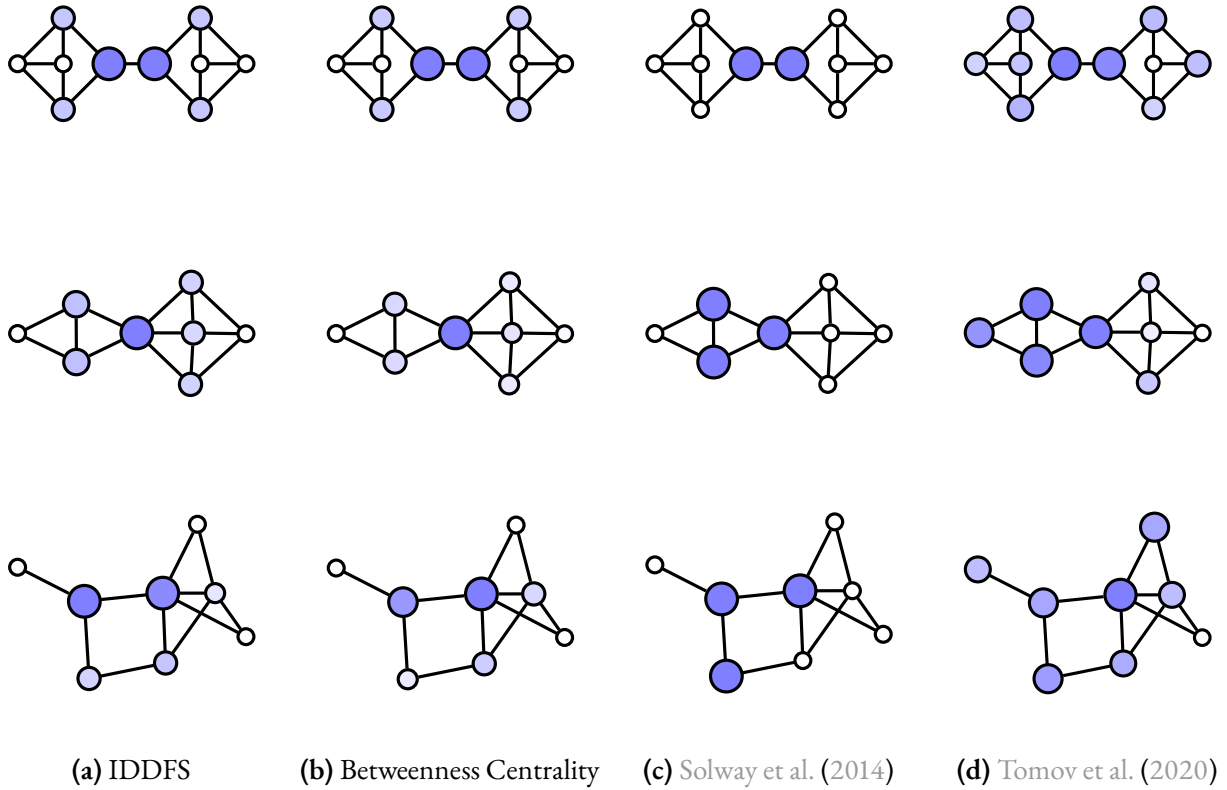


Figure 2.3: Comparing predictions of the (a) RRTD-IDDFS, (b) Betweenness Centrality, (c) Solway et al. (2014), and (d) Tomov et al. (2020) models. Tasks are plotted with states as circles and their transitions as lines connecting states. State color and size is proportional to model prediction when using the state as a subgoal. The model prediction for an RRTD model corresponds to the value of task decomposition using that subgoal, and for other models is specified in Section A.1.5. (Top) The 10-node, regular graph from Solway et al. (2014). (Middle, Bottom) Two eight-node graphs selected from the 11,117 included in our analysis.

We compare subgoal choice for several heuristic theories, several variants of our framework, and variants of the normative accounts proposed by Solway et al. (2014) and Tomov et al. (2020) in Fig. 2.4 (the theories are described in Table 2.1). Each cell of Fig. 2.4 is the correlation between the subgoal predictions of a pair of theories, averaged across all environments. For simplicity, we assume the task distribution is a uniform distribution over pairs of distinct start and goal states. For the RRTD-based models, the model prediction for a state is the corresponding value of task decomposition when the state is the only possible subgoal.

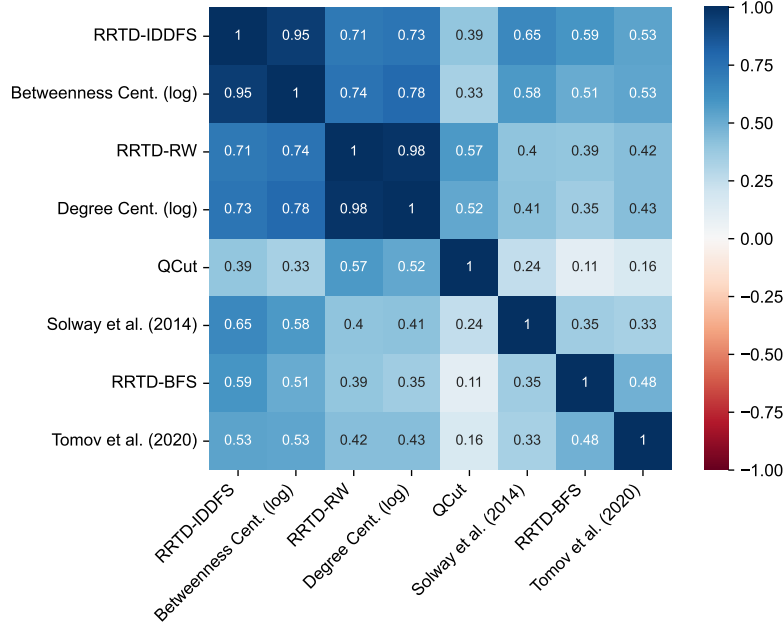


Figure 2.4: Correlation matrix comparing model predictions. For each graph, correlations between two models are computed on the per-state subgoal values, then averaged across the 11,117 simple, connected, undirected, eight-node graphs. We discard correlations when either of the two models predicts a uniform distribution over subgoals because the correlation is not defined in those instances.

We find a few notable clusters of theories—one demonstrates that RRTD-IDDFS is well-correlated with subgoal sampling based on Betweenness Centrality—this suggests the potential of a formal connection between the two algorithms, though the authors are not aware of an existing proof connecting them. A second prominent cluster shows RRTD-RW and Degree Centrality are highly correlated and that both are moderately correlated with QCut, consistent with our formal analysis. This relationship between RRTD-RW and Degree Centrality is qualitatively consistent with a published result that relates Degree Centrality to the task decomposition that minimizes a search cost related to RW (McNamee et al., 2016). The remaining algorithms—RRTD-BFS, Solway et al. (2014), and Tomov et al. (2020)—are singleton clusters, suggesting qualitative differences from the other algorithms.

To better understand these large-scale quantitative patterns, we qualitatively examine some of the normative algorithms and one heuristic. We focus on the subgoal predictions for three graphs, shown in Fig. 2.3. In the top row is a 10-node, regular graph that has been previously studied (Solway et al.,

2014; Tomov et al., 2020). In the middle row is a similar graph with critical differences: the graph is asymmetric about the graph bottleneck, and the bottleneck of the graph now corresponds to a single state instead of two connected states. In the bottom row is a graph notably distinct from graphs typically studied because it lacks an obvious hierarchical structure. All algorithms make similar predictions for the graph in the top row—an example of the difficulty in using typically studied graphs to distinguish among algorithms. Now, we look at the algorithms in more detail.

We first examine RRTD-IDDFS (Fig. 2.3a) and Betweenness Centrality (Fig. 2.3b), noting their strong agreement—this is consistent with the large-scale correlation analysis in the previous section. These two algorithms prefer the same subgoals in both graphs. At middle, they prefer the bottleneck state. At bottom, they prefer states that are close to many states—in particular, the two most-preferred states can reach any other state in at most two steps.

Now, we turn to the other normative accounts: Solway et al. (2014) (Fig. 2.3c) and Tomov et al. (2020) (Fig. 2.3d). Both rely on partition-based representations of hierarchical structure where states are partitioned into different groups. Mapping from partitions onto subgoal choices requires a step of translation. In particular, when considering a path that crosses from one group into another there are two natural subgoals that correspond to the boundary between the groups: either the last state in the first group or the first state in the second group. In the context of a task distribution, there are many possible ways to map partitions onto subgoal choices, without clear consensus between the two partition-based accounts that we consider. While these analysis choices have little impact on symmetric graphs (e.g., top row of Fig. 2.3), they are important for asymmetric graphs like the one in the middle row, which has a bottleneck state instead of a bottleneck edge. For simplicity, our implementation of Solway et al. (2014) uses the optimal hierarchy, placing uniform weight over all states at the boundaries between groups of the partition, as can be seen in Fig. 2.3c.

The algorithm from Tomov et al. (2020) introduces other subtleties. It poses task decomposition as inference of hierarchical structure, with two main criteria: 1) that there are neither too few nor too many groups (accomplished via a Chinese Restaurant Process) and 2) that connections within groups

are dense while connections between groups are sparse. The latter leads to issues when connection counts do not reflect hierarchical structure, as shown in Fig. 2.3d. At middle, the algorithm prefers partitions that minimize the number of cross-group connections, even when the bottleneck state is not on the boundary between groups. At bottom, the lack of hierarchical structure that can be detected by edge counts leads the algorithm to make diffuse predictions among many possible subgoals.

To close this section, we briefly discuss how our resource-rational account might be plausibly implemented. We outline two broad approaches: people might directly search for task decompositions that maximize Eq. 2.3, or they might attempt to approximate the objective through tractable heuristics. First, finding the optimal task decomposition in a brute-force manner is more computationally expensive than simply solving the task. One alternative is to *learn* the value of task decompositions, relying on the shared structure between tasks and subgoals to ensure learning efficiency—for example, in the domain of strategy selection, one study uses shared structure to ensure efficient estimation which is incorporated by decision-theoretic methods to deal with the uncertainty in these estimates (Lieder & Griffiths, 2017). The second approach might approximately optimize the objective by using a more tractable heuristic—the results in this section suggest two examples, where Betweenness Centrality can approximate RRTD-IDDFS and Degree Centrality can approximate RRTD-RW. While Degree Centrality is straightforward to compute, Betweenness Centrality is still computationally costly because it requires finding optimal paths for all tasks. Importantly, Betweenness Centrality has a probabilistic formulation, so it can be estimated with analytic error bounds (Borassi & Natale, 2019). In this formulation, states that are more central appear more often in paths sampled from an appropriate distribution (i.e. sample a task, then sample an optimal path uniformly at random). This suggests a trivial memory-based strategy that tracks the occupancy of states visited along paths—when the paths are appropriately sampled, the expected occupancy should be related to Betweenness Centrality. Another approach is to approximate Betweenness Centrality, like in one planning-specific method that analyzes small regions of the environment separately, then pools this information to choose subgoals (Şimşek & Barto, 2009).

The large-scale comparison of subgoal predictions in Fig. 2.4 demonstrates connections between ex-

isting heuristics and our framework for subgoal choice based on search efficiency. These connections suggest a rationale for the efficacy of these heuristics, which may stand in as tractable approximations of our resource-rational framework. Our qualitative comparison in Fig. 2.3 highlights some of the differing predictions among the accounts. But how do these different accounts relate to how people decompose tasks? We turn to this question in the next section.

2.4 AN EMPIRICAL TEST OF THE FRAMEWORK

To measure people’s task decompositions, we ask research participants to report their subgoal use after experience navigating in an environment. A previously published experiment by Solway et al. (2014) tested task decomposition using three graph navigation tasks. We developed a similar paradigm but used a set of 30 environments sampled randomly from 1,676 graphs, a subset of the 11,117 used in our large-scale model comparison above. The criteria for this subset were selected to ensure compatibility with the experiment and are detailed in the methods. This set of environments is larger and more diverse than those used in previous studies (Solway et al., 2014; Huys et al., 2012) which allows us to draw broader and more generalizable inferences about the task decomposition process.

Inspired by prior studies (Solway et al., 2014), we conducted an experiment with two phases: participants were first familiarized with an environment by performing a series of *navigation trials* (Fig. 2.5a), then answered a series of questions about their subgoal choices (Figs. 2.5c, 2.5d).

Participants gained exposure to the environment by performing 30 navigation trials requiring navigation to a goal state from some initial state. These *long trials* were randomly selected while ensuring the initial and goal states were not directly connected (Fig. 2.5a). Participants moved from the current state to a neighboring state using numeric keys. The trial ended when the goal state was reached. In simulations and pilot studies, states with high visit rates coincided with the predictions of RRTD-IDDFS and Betweenness Centrality. This made it difficult to dissociate model predictions from an alternative memory-based strategy where frequently visited states are selected as subgoals. As noted in the previous

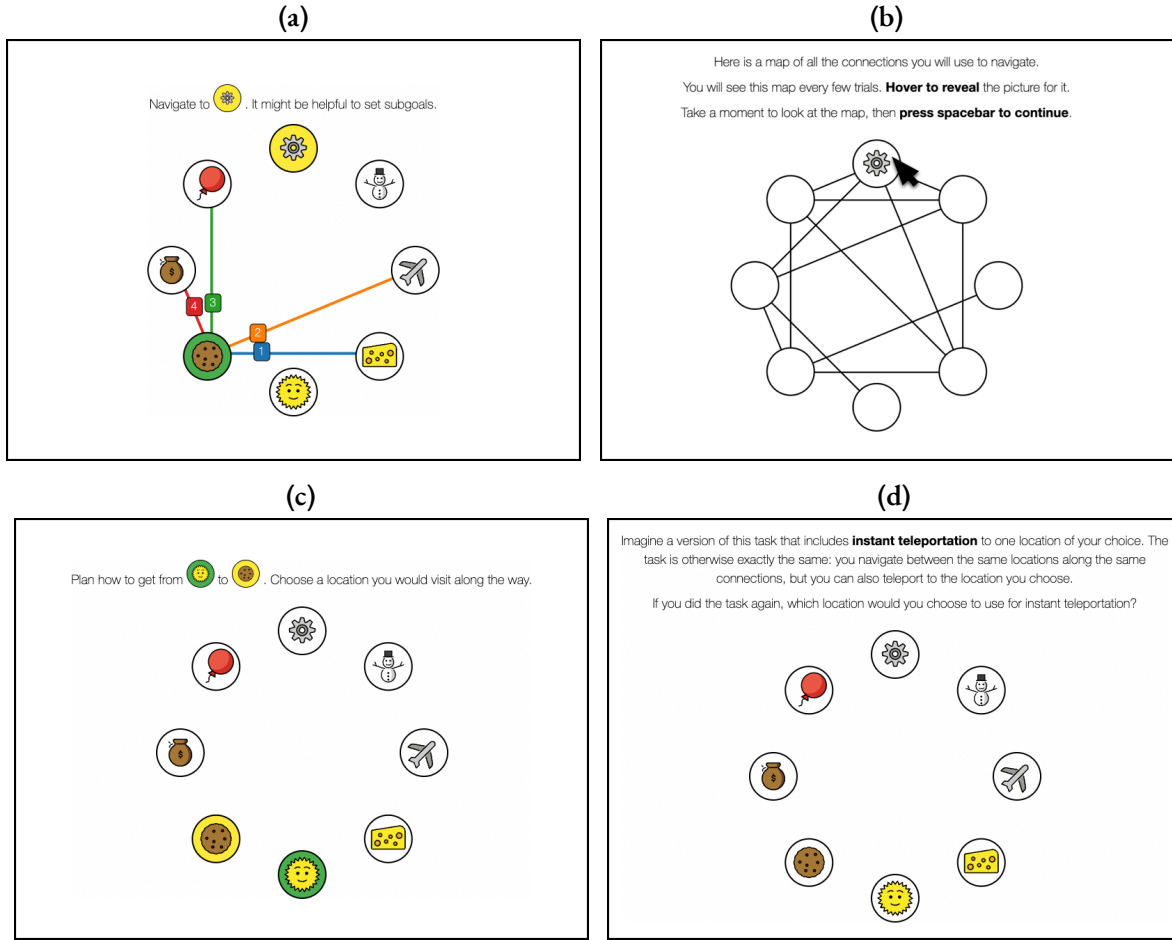


Figure 2.5: Screenshots from the experimental interface. The depicted graph is the same as the top left graph of Fig. A.1. (a) An example *navigation trial*. The current state has a green background and the goal state has a yellow background. Only the edges connected to the current state are shown. (b) The interface used to show all graph edges between navigation trials. There is no indication of past or future trials on this screen. State icons are only shown when the cursor is placed on them. (c) An example *implicit subgoal probe*. (d) The final post-task assessment with the *teleportation question*. All icons were designed by OpenMoji and are reproduced here with permission.

section, this memory-based strategy is related to sampling-based estimation of Betweenness Centrality. To address this confound, we modified the experimental task distribution so that long trials were interleaved with *filler trials* requiring navigation to a state directly connected to the start state. These filler trials were adaptively selected to increase visits to states besides the most frequently visited one; in pilot studies and simulations, this was sufficient to dissociate visit rate and model predictions.

A critical methodological difficulty is visually representing the environment in a way that enables rapid learning but does not introduce confounds. To prevent participants from relying on heuristics such as the Euclidean distance between states, states were assigned random locations in a circular layout. The absence of useful heuristics ensures the problems participants face are more comparable to that solved by brute-force search algorithms, which also lack a heuristic. To encourage model-based reasoning instead of visual search, connections between states were only shown for the current state. So that participants could still easily learn the connections, participants were periodically shown the graph with all connections between trials (Fig. 2.5b).

To query participant subgoal choice, we used both direct and indirect probes to comprehensively and reliably measure subgoal choice, including novel as well as previously studied prompts (Solway et al., 2014). In the context of 10 navigation trials, we first prompted participants “Plan how to get from A to B. Choose a location you would visit along the way,” the *implicit subgoal probe* (Fig. 2.5c). Then, in the context of the same trials after shuffling, we asked participants “When navigating from A to B, what location would you set as a subgoal? (If none, click on the goal),” the *explicit subgoal probe*. In order to ensure familiarity with the concept of a subgoal, participants were introduced to the concept of a “subgoal” in the context of a cross-country road trip during the experiment tutorial. In a final post-task assessment, we asked participants “If you did the task again, which location would you choose to use for instant teleportation?”, the *teleportation question* (Fig. 2.5d). We asked this question outside the context of any particular navigation trial.

2.5 EXPERIMENT RESULTS

We recruited English-speaking participants in the United States on the Prolific recruiting platform, pre-screening to exclude participants of previous experimental pilots and those with approval ratings below 95%. Of the 952 participants that completed the experiment, 806 (85%) satisfied the pre-registered exclusion criteria requiring efficient performance on the navigation trials. If a participant took 75% more

actions than the optimal path (averaged across the last half of long trials), their data was excluded. The number of participants per graph varied after exclusion criteria were applied, without significant differences per graph (before exclusion: range 27–34, after exclusion: range 21–30, two-factor χ^2 test comparing included to excluded, $p = .985$). Participants took an average of 17.17 minutes ($SD = 8.02$) to complete the experiment.

Even though the experimental interface obfuscated task structure by showing the task states in a random circular layout, participants became more effective from the first to the second half of training: long trials were solved more quickly (from 10.30s ($SD = 29.74$) to 7.60s ($SD = 11.19$)), with more efficient solutions (from 36% to 20% more actions than the optimal path; completely optimal solutions increased from 70% to 79%; solutions that included a repeated state decreased from 14% to 9%), and with decreased use of the map (on-screen duration decreased from 9.01s ($SD = 20.97$) to 2.98s ($SD = 12.66$); number of hovered states decreased from 5.43 ($SD = 8.70$) to 1.38 ($SD = 3.99$); duration of state hovering decreased from 2.52s ($SD = 7.23$) to 0.60s ($SD = 9.04$)). In order to rule out confounding effects due to differences in the complexity and structure of tasks that participants solved, we related participant behavior on the probes to measures associated with each graph and found no significant relationships in Appendix A.

Our findings are organized into two sections: First, an analysis of the subgoal probes, demonstrating their internal consistency and relationship to behavior. Then, model-based prediction of subgoal probe choice, as well as a subset of choice behavior.

2.5.1 SUBGOAL PROBES ARE INTERNALLY CONSISTENT AND PREDICT BEHAVIOR

A crucial methodological concern is the validity of the probes for subgoal choice—in existing studies, various types of probes have been used, but not compared systematically. Choice on the explicit and implicit subgoal probes had high within-probe consistency across participants while choice on the teleoperation question had low within-probe consistency across participants, based on the average correlation between per-participant choice rates and per-graph choice rates (Explicit Probe $r = 0.71$, Implicit

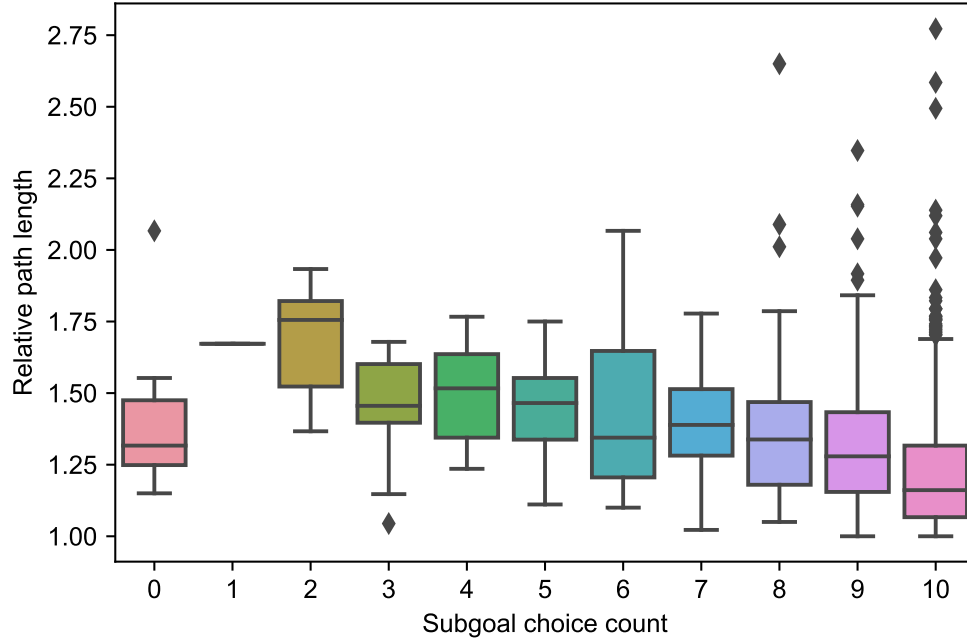


Figure 2.6: Participants that choose a subgoal instead of the goal more often in Explicit Probe trials have shorter average path length, relative to the optimal path length ($r = -0.29, p < .001$).

Probe $r = 0.63$, Teleportation Question $r = 0.37$). We also evaluated consistency between probes by comparing the per-graph, per-state choice rates. The Explicit and Implicit Probes were well-correlated ($r = 0.98, p < .001$), though the relationship between the Teleportation Question and the remaining probes was weaker (Teleportation Question and Explicit Probe: $r = 0.58, p < .001$, Teleportation Question and Implicit Probe: $r = 0.58, p < .001$). While the Teleportation Question exhibits relatively low self-consistency and cross-probe consistency, it is difficult to compare to the consistency of the other probes since both the Explicit and Implicit probes were sampled for 10 different tasks per participant, while the Teleportation Question was only sampled once per participant.

Beyond simply assessing consistency, it is also crucial to link the probes to participant behavior during navigation, ensuring there is a link between decision-making and our indirect assessment via probes. On the Explicit Probe trials, participants were given the option of choosing the goal instead of a subgoal. On average, participants who chose a subgoal more frequently took shorter paths in the navigation trials ($r = -0.29, p < .001$; Fig. 2.6), which suggests that use of subgoals promotes efficiency.

We also briefly examine the relationship between subgoal choice count on Explicit Probe trials and response times during navigation. We were unable to find evidence of a correlation between the subgoal choice count of a participant and their average log-transformed navigation trial duration ($r = 0.01$, $p = .858$). In order to understand how subgoal use influences response times, future studies should examine trial-level measures in appropriate experimental designs, an issue we remark on in the discussion.

To further link the probes to behavior, we examine instances where participants performed the same task (matched by start and goal state) in the navigation trials and the probe trials. This allows us to ask whether participants took paths that passed through the states they later identified as subgoals. To simplify the interpretation of the analysis, we focus on navigation trials where the participant’s path was optimal and there were multiple optimal paths between the start and goal. Evaluated over these pairs of matched navigation and probe trials, we found that participants’ choices on probe trials were consistent with their choices among optimal paths (Explicit Probe: 75.4%, Implicit Probe: 70.6%) more often than would be expected by random choice among optimal paths (Explicit Probe: 70.5%, $p < .001$; Implicit Probe: 65.2%, $p < .001$; Monte Carlo test). Probe trial choice is also a significant predictor of choice among optimal paths when analyzed using multinomial regression (Explicit Probe: $\chi^2(1) = 54.7$, $p < .001$, Implicit Probe: $\chi^2(1) = 62.5$, $p < .001$).

In sum, these results suggest the subgoal probes are well-correlated, though to a lesser degree for the Teleportation Question. They also suggest a strong connection between the probes and planning behavior.

2.5.2 COMPARING SUBGOAL CHOICE TO THEORIES

Having established that participants learned the task, as well as the validity of their probe responses, we now turn to our central claim, namely that subgoal choice is driven by the computational costs of hierarchical planning. Letting participants’ responses to the subgoal probes stand as reasonable proxies of subgoal choice, we relate the predictions of normative accounts and heuristics to participant probe choice across the three probes.

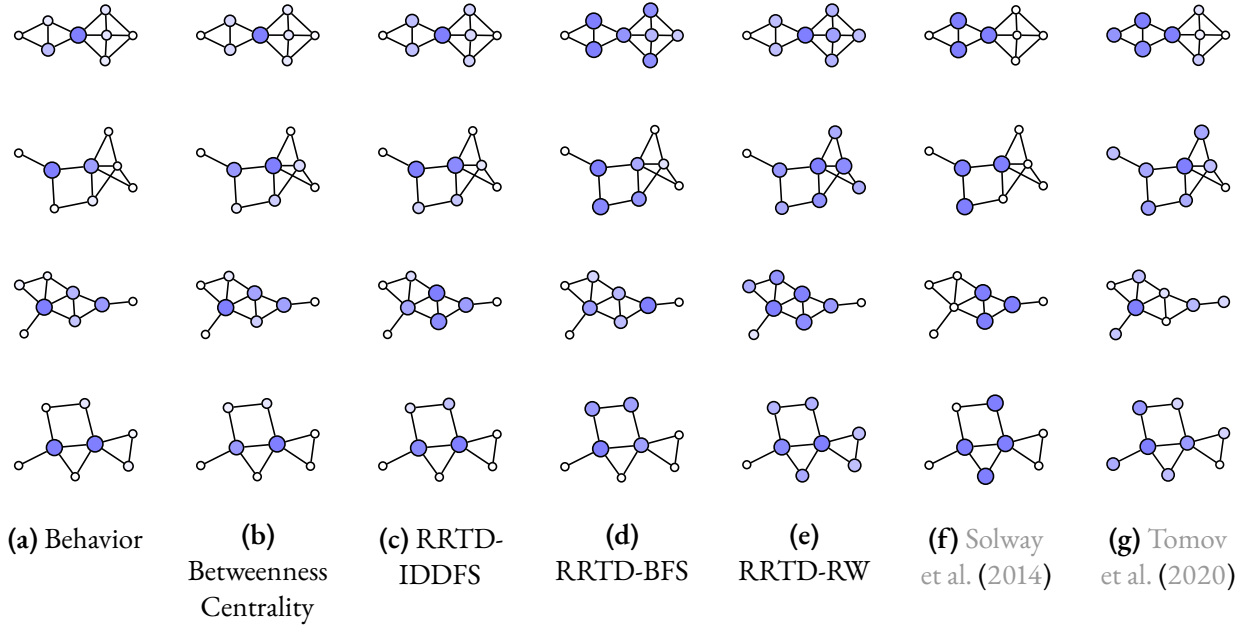


Figure 2.7: Visualization of behavior and model predictions on four eight-node graphs selected from the 30 graphs used for the experiment. (a) State color and size is proportional to subgoal choice, summed across participants and probe types. Each participant responded to a total of 21 subgoal probes and the number of participants per graph, from the top graph to the bottom graph, was 28, 26, 25, and 26. Visualized models are (b) Betweenness Centrality, (c) RRTD-IDDFS, (d) RRTD-BFS, (e) RRTD-RW, (f) Solway et al. (2014), and (g) Tomov et al. (2020). For model predictions, state color and size is proportional to model prediction when using the state as a subgoal.

We start by qualitatively examining participant subgoal choice in Fig. 2.7, extending the qualitative analysis given above with two additional graphs, more model predictions (Fig. 2.7b-2.7g), and behavioral data averaged across tasks, probes, and participants (Fig. 2.7a). Participant data for all graphs is in Fig. A.2 in Appendix A. We first note that Betweenness Centrality and RRTD-IDDFS are consistent in the graphs (Fig. 2.7b-2.7c), and are both relatively consistent with participant probe choice—as described above, states that are close to many other states are preferred. For brevity, we skip over RRTD-BFS and RRTD-RW in this description. The predictions of Solway et al. (2014) are less consistent with participant probe choices (Fig. 2.7f)—as previously described, the predictions are unintuitive because of the difficulty in mapping between partitions and subgoals, particularly when graph bottlenecks correspond to states instead of edges. The predictions of Tomov et al. (2020) are also less consistent with

participant probe choices (Fig. 2.7g)—as previously described, the predictions do not correspond with intuitive subgoals because the model relies on between-group edges being sparser than within-group edges.

We now quantitatively compare model predictions of participant subgoal choice.

For each model and probe type, we predict participant choices using hierarchical multinomial regression, where standardized model predictions are included as a factor with a fixed and per-participant random effect. Since the regression analyses have the same effect structure and the underlying theories being compared have no free parameters, we compare the relative ability of factors to predict probe choice through their log likelihood (LL) in Fig. 2.8. We also report the results of likelihood-ratio tests to the null hypothesis of a uniformly random choice model in Table 2.2. As in the analysis above, we assume the task distribution is uniformly-distributed over all pairs of distinct states. Among normative theories, we found the RRTD-IDDFS model best explained behavior as judged by LL. For the Explicit and Implicit Probes, the next best models in sequence were RRTD-BFS, then both Solway et al. (2014) and Tomov et al. (2020) with similar performance, and finally RRTD-RW. For the Teleportation Question, the best models after RRTD-IDDFS were Solway et al. (2014), Tomov et al. (2020), RRTD-BFS, and finally RRTD-RW. This suggests that, among the normative theories, those based on search costs are most explanatory of subgoal choices.

We additionally compare model predictions to participant state occupancy during navigation trials in order to assess whether people are relying on simple, memory-based strategies to respond to the probes, as described above. We find that participant behavior is better explained by all normative theories for the Explicit and Implicit Probes (with the exception of RRTD-RW), but only RRTD-IDDFS and Solway et al. (2014) for the Teleportation Question.

Among the heuristic theories, we found Betweenness Centrality best explained behavior as judged by LL. For all probes, Degree Centrality was next best, followed by QCut. As above, we compared model predictions to participant state occupancy and found that participant behavior is better explained by Betweenness Centrality for the Explicit and Implicit Probes, and both Betweenness and Degree Cen-

Table 2.2: Estimated coefficients with standard errors from hierarchical multinomial regression predicting sub-goal choice. Likelihood-ratio test statistics compare regression models to the null hypothesis of sampling subgoals uniformly at random.

Normative Algorithm	Explicit Probe	Implicit Probe	Teleportation Question
RRTD-IDDFS	$\beta = 1.78$ $SE = 0.04$ $\chi^2(2) = 5183.1$ $p < .001$	$\beta = 1.63$ $SE = 0.04$ $\chi^2(2) = 3941.2$ $p < .001$	$\beta = 0.73$ $SE = 0.06$ $\chi^2(1) = 155.7$ $p < .001$
RRTD-BFS	$\beta = 4.98$ $SE = 0.10$ $\chi^2(2) = 4412.8$ $p < .001$	$\beta = 4.94$ $SE = 0.11$ $\chi^2(2) = 3402.8$ $p < .001$	$\beta = 1.45$ $SE = 0.20$ $\chi^2(1) = 55.1$ $p < .001$
RRTD-RW	$\beta = 0.37$ $SE = 0.02$ $\chi^2(2) = 686.7$ $p < .001$	$\beta = 0.92$ $SE = 0.03$ $\chi^2(2) = 1822.1$ $p < .001$	$\beta = 0.29$ $SE = 0.05$ $\chi^2(1) = 39.9$ $p < .001$
Solway et al. (2014)	$\beta = 0.75$ $SE = 0.02$ $\chi^2(2) = 3929.7$ $p < .001$	$\beta = 0.69$ $SE = 0.02$ $\chi^2(2) = 3072.7$ $p < .001$	$\beta = 0.37$ $SE = 0.03$ $\chi^2(1) = 114.3$ $p < .001$
Tomov et al. (2020)	$\beta = 1.09$ $SE = 0.03$ $\chi^2(2) = 3678.7$ $p < .001$	$\beta = 0.97$ $SE = 0.02$ $\chi^2(2) = 3056.4$ $p < .001$	$\beta = 0.41$ $SE = 0.04$ $\chi^2(1) = 102.4$ $p < .001$
Heuristic	Explicit Probe	Implicit Probe	Teleportation Question
QCut	$\beta = -0.14$ $SE = 0.01$ $\chi^2(2) = 1504.3$ $p < .001$	$\beta = -0.19$ $SE = 0.01$ $\chi^2(2) = 1236.3$ $p < .001$	$\beta = 0.04$ $SE = 0.04$ $\chi^2(1) = 1.4$ $p = .238$
Degree Cent. (log)	$\beta = 0.73$ $SE = 0.02$ $\chi^2(2) = 2534.3$ $p < .001$	$\beta = 0.64$ $SE = 0.02$ $\chi^2(2) = 1923.3$ $p < .001$	$\beta = 0.45$ $SE = 0.04$ $\chi^2(1) = 116.0$ $p < .001$
Betweenness Cent. (log)	$\beta = 0.86$ $SE = 0.02$ $\chi^2(2) = 5598.2$ $p < .001$	$\beta = 0.82$ $SE = 0.02$ $\chi^2(2) = 4666.1$ $p < .001$	$\beta = 0.58$ $SE = 0.03$ $\chi^2(1) = 307.3$ $p < .001$
State Occupancy (log)	$\beta = 1.00$ $SE = 0.03$ $\chi^2(2) = 3222.8$ $p < .001$	$\beta = 0.89$ $SE = 0.03$ $\chi^2(2) = 2673.3$ $p < .001$	$\beta = 0.39$ $SE = 0.04$ $\chi^2(1) = 103.3$ $p < .001$

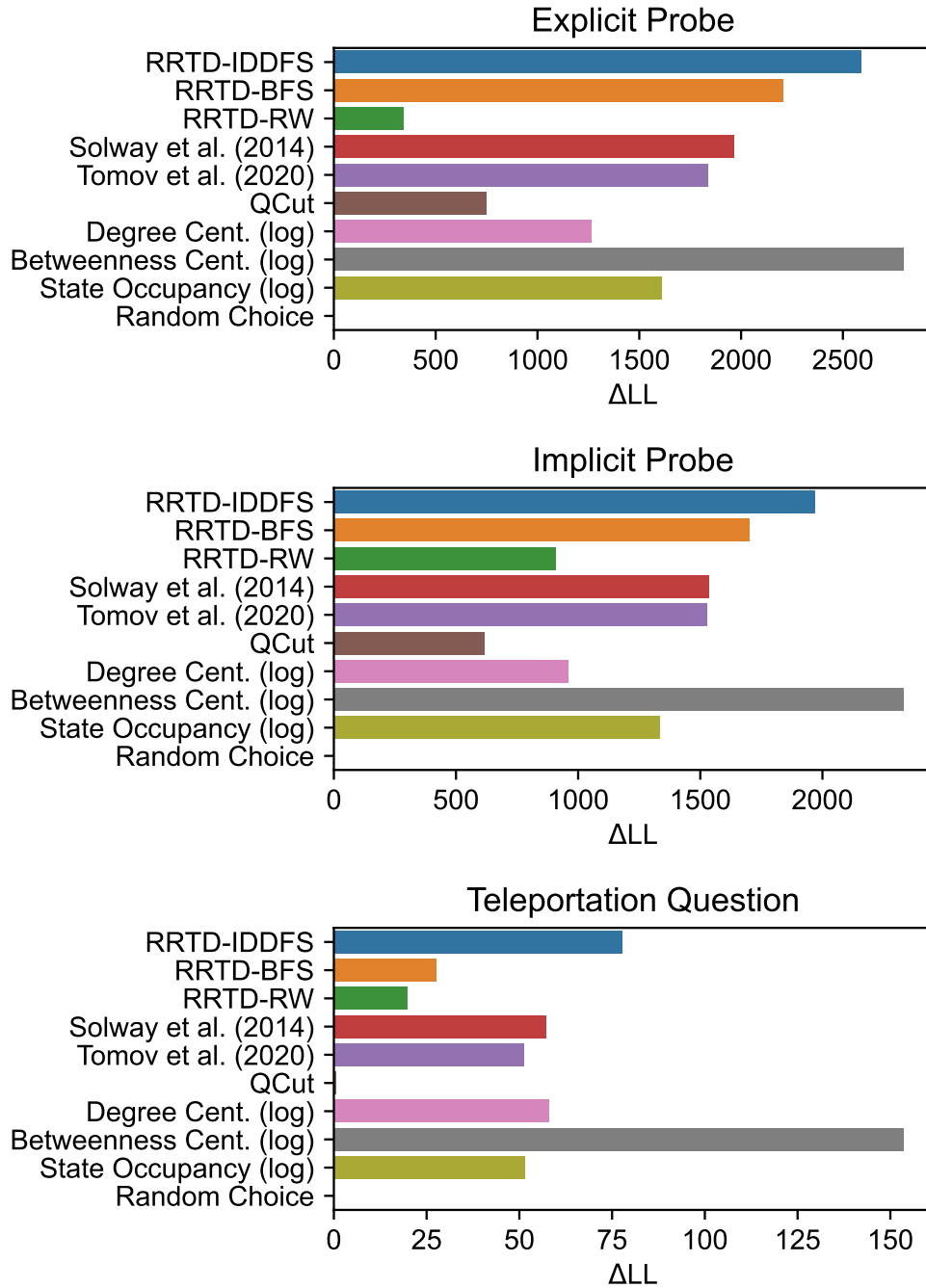


Figure 2.8: Comparison of statistical analysis using mixed-effects multinomial regression to predict subgoal choice behavior for each subgoal probe. Log likelihood (LL) is relative to the minimum model LL for each probe. Larger values indicate better predictions of behavior.

trality for the Teleportation Question.

These results are consistent with the empirical connection between RRTD-IDDFS and Betweenness Centrality found in the large-scale simulation above. Betweenness Centrality further improves on the behavioral fit of RRTD-IDDFS, suggesting that our participants may be using a metric like Betweenness Centrality to approximate resource-rational task decomposition. In contrast, we found that state occupancy was a worse fit to participant behavior than either RRTD-IDDFS or Betweenness Centrality, suggesting that the introduction of filler trials was sufficient to rule out a trivial strategy based on state occupancy. We return to these points in the discussion.

Since the experiment was designed so that only local connections were visible during navigation trials but conducted via an online platform, in the closing survey we asked participants if they used a reference to the task structure besides the interface (“Did you draw or take a picture of the map? If you did, how often did you look at it?”). Participant responses were as follows: 603 participants selected “Did not draw/take picture,” 65 selected “Rarely looked,” 90 selected “Sometimes looked,” and 48 selected “Often looked.” In order to ensure the above results were not impacted, we ran the same analysis in the subset of participants ($N = 603$) that selected “Did not draw/take picture” and found qualitatively similar results (Fig. A.3 and Table A.1 in Appendix A).

In another analysis in Appendix A, we tested whether icon identity influenced these results by incorporating the icon used for state presentation into the null choice model. We found minimal influence—like the above results, the addition of subgoal predictions was statistically significant for each model and comparisons based on log likelihood were qualitatively similar.

In a final analysis, we predict participant navigation in the instances where their path was one of several optimal paths. We analyze participant choice as a simple two-stage process: a subgoal is sampled with log probability proportional to model predictions (weighted by a parameter β_1), then an optimal path is sampled with log probability proportional to a free parameter β_2 if it contains the subgoal and 0 otherwise. For each theory of subgoal choice, we optimized this two-stage choice model to maximize the likelihood assigned to observed choices. Because there were a relatively small number of trials

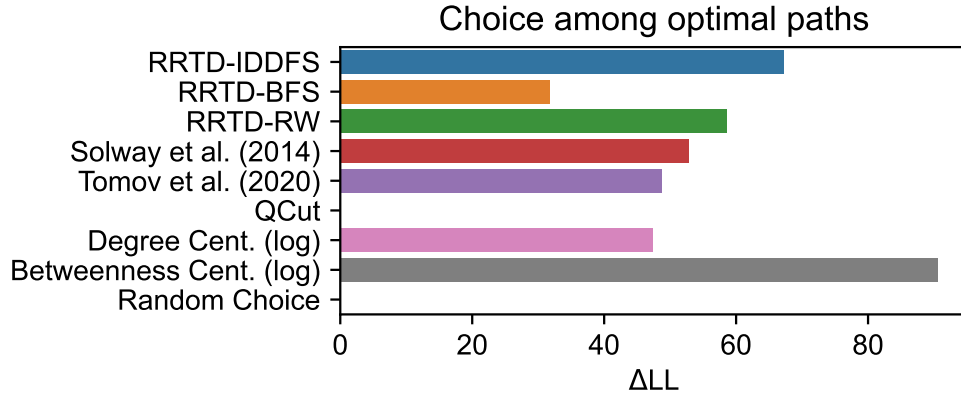


Figure 2.9: Comparison of two-stage choice models to predict participant choice behavior among optimal paths. Log likelihood (LL) is relative to the minimum model LL. Larger values indicate better predictions of behavior.

per participant, we did not fit random effects for participants. The model results are shown in Fig. 2.9. These results are again consistent with those previously observed—among normative theories RRTD-IDDFS is best, among heuristic theories Betweenness Centrality is best, and Betweenness Centrality is overall the most explanatory. In a supplementary analysis, we also found evidence that participant responses during optimal navigation were slower at both self-reported subgoals and those predicted by models, as detailed in Appendix A.

2.6 DISCUSSION

In this work, we have proposed a resource-rational framework for task decomposition where tasks are broken down into subtasks based on planning costs. Our first contribution is a novel formal account of this idea based on a resource-rational analysis (Lieder & Griffiths, 2020). Specifically, our proposal involves three levels of nested optimization: *Task decomposition* identifies a set of subgoals for a given task, *subtask-level planning* chooses sequences of subgoals to reach a goal, and *action-level planning* chooses sequences of concrete actions to reach a subgoal. Optimal task decomposition thus depends on both the structure of the environment and the computational resource usage specific to the planning algorithm. We quantitatively compared the predictions of our framework to four heuristic and norma-

tive theories proposed in the literature across 11,117 graph-structured tasks. These analyses show that our framework provides different predictions from other normative accounts and aligns with heuristics. We argue that this provides a rationalization of these heuristics for task decomposition in terms of resource-rational planning that accounts for computational costs.

To test our framework, we ran a pre-registered, large-scale study using 30 graph-structured environments and 806 participants. This study includes a more diverse set of tasks than that of any previously reported study in the literature, allowing us to draw more general conclusions about how people form task decompositions. The results of this study reveal that, among normative models, people’s responses most closely align with the predictions of our model. This provides support for our theory that people are engaged in a process of resource-rational task decomposition. Among heuristics for task decomposition, one heuristic is a better fit to behavior than our framework. Because the heuristic makes similar predictions as our framework, this might indicate that people use the heuristic as a tractable approximation to our framework.

Our account, while normative, is not particularly interpretable. Identifying the qualitative patterns that guide human subgoal choice and relating them to the patterns resulting from our framework’s sensitivity to search costs will be necessary for an interpretable account of human subgoal choice. Critically necessary are experimental paradigms that provide rich, but minimally confounded behavior—our experiments extend those in the literature, but our results depend heavily on the self-reported subgoals of research participants. While we have already demonstrated relationships between self-reported subgoals and behavior, making more extensive comparisons to behavior is important for future research. For example, although we found a systematic relationship between participants’ responses to the subgoal probes and their previous navigation decisions, these two measurements were taken minutes apart. We chose to separate these two measurements so as to avoid possible measurement effects in which explicitly asking about subgoals may lead people to navigate through the state they identified. However, this likely weakens the observed relationship between the probe responses and the navigation decisions. An additional difference during navigation trials is that participants are still learning the task structure

and are shown connections from the current state. By contrast, during the probe trials, participants are unable to see any connections and must rely solely on what they have learned. These subtle differences may reduce the relationship between these two trial types. They are also different from the formal framework which assumes perfect knowledge about task structure—because this could influence task decomposition, we note a relevant extension below. The effect of this difference could be studied by introducing a separate experimental phase where participants are trained on the task structure directly, outside the context of navigation—the first experiment in Solway et al. (2014) contains a similar phase, but had limited navigation trials. Developing experimental techniques to measure subgoal choices in the process of navigation without biasing the planning process is an important direction for future work.

We highlight three limitations of the navigation trials studied in this experiment. The first is that participants are encouraged to plan hierarchically (“It might be helpful to set subgoals” in Fig. 2.5a). While this seems likely to have minimal impact on *which* subgoals participants choose, the main focus of this chapter, it may impact *whether* participants plan hierarchically in the first place. Future studies intending to assess how people choose to plan hierarchically should consider avoiding prompts like this.

Second, though we report analyses of participant response times, these analyses were not pre-registered and our experiment was not designed to assess response times. These findings should be reevaluated in experimental designs appropriate to assess response times. For example, we found that participants were slower to respond at their subgoals, suggesting they were planning at those states. However, our normative framework makes no prediction about *when* planning occurs. The framework only defines a resource-rational value for subgoals that can be used to simplify planning whether it happens before action or after reaching a subgoal. Future experiments could investigate this further through manipulations to influence when planning is employed, like a timed phase for up-front planning or an incentive for fast plan execution.

A third limitation is that the navigation trials provide limited insight into the algorithms people are using to plan. While participants only see local connections during navigation, analogous to the local

visibility search algorithms have, there are a number of reasons why their navigation behavior would be difficult to relate to the choice of search algorithm. The main issue is that planning steps are generally covert, and do not correspond in any simple way with steps of overt behavior given by the plan that is ultimately produced. Such covert planning is better suited, in future work, to being studied through process-tracing experiments (Callaway et al., 2022; Ho et al., 2022), think-aloud protocols, or by investigating neural signatures related to planning and learning (Liu et al., 2021). It is possible that some aspects of planning are externalized in the current experiment, via exploration on navigation trials, but this is at best incomplete. For instance, participant behavior improves over the course of navigation trials, suggesting they are performing mental search instead of search via navigation. Also, the experiment restricts single-step movement to neighboring states, whereas by contrast, algorithms like BFS might plan over states in an order where subsequent states are not neighbors. Identifying the search algorithm participants use is critical for future studies since our framework predicts that task decomposition is driven by the search algorithm used.

As mentioned in the text, our framework and experiment explicitly focus on the constrained setting of brute-force search. However, other theoretical studies have extended this framework to incorporate heuristic search (Correa et al., 2020; Binder et al., 2021) and abstract subgoals (Binder et al., 2021). Future research should continue to explore extensions of this framework to more robustly test the predictions of resource-rational task decomposition. For example, our framework could be used to make predictions about subgoal choice in spatial navigation tasks by incorporating spatial distance heuristics and using heuristic search algorithms like A* search or Iterative-Deepening A* search (Ghallab et al., 2016). Another direction could explore other resource costs like memory use, motivated by the relatively low memory use of IDDFS discussed above. At present, our framework assumes planning for tasks occurs independently, avoiding the reuse of previous solutions to subtasks. Despite this absence, our framework still predicts a normative benefit for problem decomposition. However, research has found that people learn hierarchically, exhibiting neural signatures consistent with those predicted by hierarchical reinforcement learning theory (Ribas-Fernandes et al., 2011). Further research could relax the indepen-

dence between task solutions by explicitly reusing solutions (as in Solway et al., 2014) or turning to formulations based on reinforcement learning (Harb et al., 2018), particularly those designed for a distribution of goal-directed tasks with shared structure as studied in this chapter (Nasiriany et al., 2019). A particularly interesting direction could incorporate model learning (as in the Dyna architecture in Sutton & Barto, 2018) across tasks in order to explain the influence of task learning on task decomposition. Further extension of this framework could build on resource-rational models developed in other domains, like Markov Decision Processes (Jinnai et al., 2019) and feedback control (Prystawski et al., 2022), and draw inspiration from approaches used to learn action hierarchies in high-dimensional tasks (Harb et al., 2018; Nasiriany et al., 2019).

Our framework is intended to be an idealized treatment of the problem of task decomposition, but an essential next step for this line of research is to understand how people tractably approximate the expensive computations needed to determine search costs, which we discuss above. Our results already make some progress in this direction. Specifically, we found that participant responses were best explained by the Betweenness Centrality model. This model's predictions are highly correlated with the normative RRTD-IDDFS model but require far less computation to produce. This suggests that people may be using Betweenness Centrality as a heuristic to approximate the task decomposition that minimizes planning cost. However, Betweenness Centrality is also expensive to compute since it requires finding optimal paths between all pairs of states—something our participants are not likely doing. Since we were able to rule out one trivial estimation strategy (state occupancy) through the inclusion of filler trials, the issue of tractable estimation is an open question for future research to explore by proposing other estimation strategies and experimental manipulations to dissociate their predictions. Identifying even more efficient approximations to resource-rational task decomposition will be essential for a process-level account of human behavior, as well as for advancing a theory of subgoal discovery for problems with larger state spaces.

The human capacity for hierarchically structured thought has proven difficult to formally characterize, despite its intuitive nature and long history of study (Newell & Simon, 1972; Sacerdoti, 1974). In

this study we propose a resource-rational framework that motivates and explains the use of hierarchical structure in decision-making: People are modeled as having subgoals that reduce the computational overhead of action-level planning. Our framework departs from and complements other normative proposals in the literature. Most published accounts pose task decomposition as an *inference problem*: People are modeled as inferring a generative model of the environment (Collins & Frank, 2013; Tomov et al., 2020) or as compressing optimal behavior (Solway et al., 2014; Maisto et al., 2015). We quantitatively relate our framework to existing proposals in a simulation study; In addition, we conduct a large-scale behavioral experiment and find that our framework is effective at predicting human subgoal choice. The work presented here is consistent with other recent efforts within cognitive science to understand how people engage in computationally efficient decision-making (Griffiths et al., 2015; Lewis et al., 2014; Gershman et al., 2015; Lieder & Griffiths, 2020). It is also complementary to recent work in artificial intelligence that explores the interaction between planning and task representations (Jinnai et al., 2019; Harb et al., 2018). Our hope is that future work on human planning and problem-solving will continue to investigate the relationships between computation, representation, and resource-rational decision-making.

Chapter 3

Exploring the hierarchical structure of human plans via program generation *

3.1 INTRODUCTION

Human behavior has rich hierarchical structure, in which actions are grouped into abstract, higher-level actions that are used to accomplish tasks (Miller et al., 1960; Newell & Simon, 1972; Klir & Simon, 1991). However, these internal representations are not observable, so they are typically inferred from behavioral signatures of hierarchy (Rosenbaum et al., 1983). Consider the problem of *making tomato sauce for dinner*. The task can be made successively more concrete as *prepare the tomatoes* then *prepare one tomato* then *slice the tomato*. Some of these actions have highly repetitive structure, like preparing a tomato by repeatedly slicing it. While action hierarchies arise naturally from practiced behavior, they can still provide a benefit in novel settings by providing a compact representation for behavior and by radically decreasing the complexity of search for a solution.

How do people generate hierarchical plans and action hierarchies, particularly in novel settings?

Since the hierarchical structure of behavior is not directly observable, standard approaches infer this

*This chapter is based on the following preprint:

Correa, C. G., Sanborn, S., Ho, M. K., Callaway, F., Daw, N. D., & Griffiths, T. L. (2023b). Exploring the hierarchical structure of human plans via program generation. *arXiv preprint arXiv:2311.18644*.

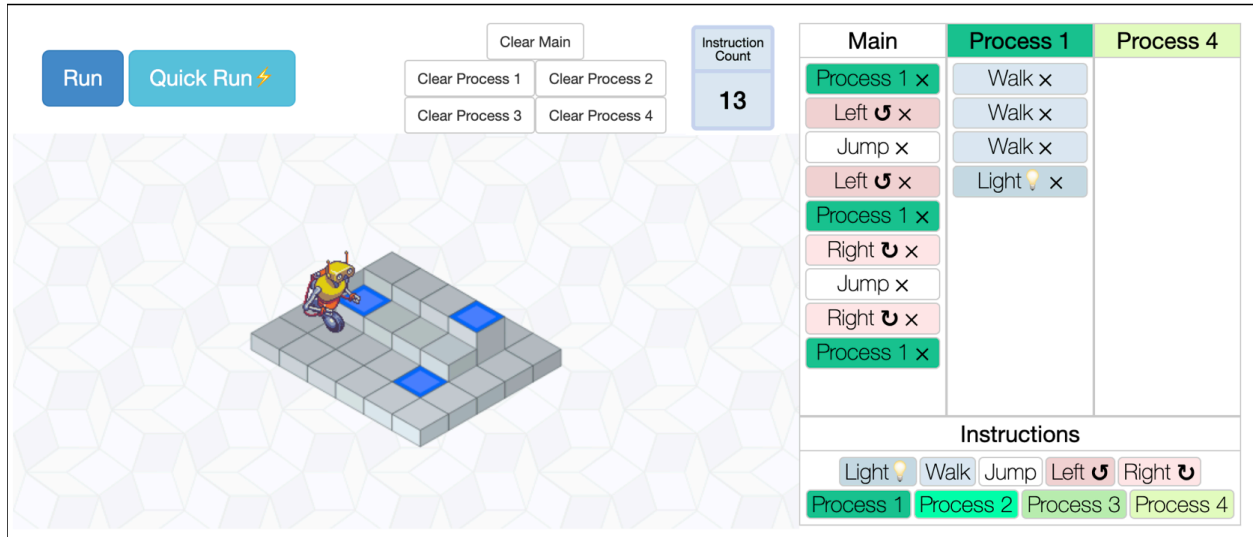


Figure 3.1: The interface for Lightbot, a process-tracing experiment for complex, hierarchical plans. Participants create programs by dragging instructions from the lower right to define a program. The program is executed by the robot from the initial subroutine (“Main”), and the task is completed when all blue squares in the environment are activated. An example program that solves the task is shown. The program includes a single subroutine (“Process 1”) with four actions (Walk, Walk, Walk, Activate Light). Participants can use up to four subroutines (referred to as processes in the experiment), but for brevity only two are pictured. The *Run* button executes the program with an animation of the robot taking each action. The program is executed without animation with the *Quick Run* button. The program length is displayed as an *Instruction Count*, and the white buttons at the top are used to clear all instructions from a subroutine for ease of program editing.

structure from behavioral signatures of hierarchy. For example, in the sequence-learning literature, hierarchies are inferred based on elevated response times or error rates when switching between abstract actions (Rosenbaum et al., 1983; Verwey, 1996; Acuna et al., 2014). Similar methods are applied when investigating hierarchical planning, though additional behavioral attributes can be measured such as the particular choice of action sequence or self-reports about behavioral hierarchy (Solway et al., 2014; Huys et al., 2015; Tomov et al., 2020; Correa et al., 2023a).

In order to make hierarchical plans observable, we use a process-tracing paradigm where participants are tasked with creating hierarchically-structured plan-like *programs* in an environment called Lightbot.[†] In the task, participants create a program to guide a robot that starts in a specific location (Fig. 3.1) and must visit all unlit lights (blue squares) to activate them. The programs must contain simple actions (e.g., Walk, Activate Light, Turn Right), but can also have hierarchical struc-

ture by reusing sequences of actions as *subroutines*. These programs serve as an explicit representation of a hierarchical plan, where abstract actions correspond to subroutines. In previous studies, this experimental paradigm has been used to demonstrate that people prefer compressible programs (Sanborn et al., 2018).

This process-tracing paradigm facilitates the comparison of different accounts of hierarchy selection. For example, given the preference for compressibility in this domain (Sanborn et al., 2018), a natural theory is that hierarchy selection is based on the principle of minimum description length (MDL; Chater, 1999). MDL and related information-theoretic frameworks have been previously used to explain hierarchical representations in sequence learning (Planton et al., 2021) and planning (Solway et al., 2014; Maisto et al., 2015; Lai et al., 2022). Another natural theory is that of utility maximization, e.g. solving the problem in the fewest steps. In this work, we find evidence that people favor both shorter programs and those with fewer actions. However, we additionally identify examples in which people prefer reusing subroutines above and beyond the predictions of either account. We formalize this preference as a prior belief that subroutine use should be biased towards subroutines that were used often in the past. This prior is central to our theory of human planning as *grammar induction*, an approach drawn from linguistics where it has been used to model word learning as joint inference of a grammar over words and sentences using those words (Goldwater et al., 2009; Johnson et al., 2006). Inspired by this analogy, we use this approach to model planning as joint inference of a grammar over abstract actions and a plan using those actions. Here the preference for reuse arises from a prior over the abstract (subroutine-level) actions (Johnson et al., 2006).

In this chapter, we study how people select hierarchical plans by directly analyzing the programs they create in our process-tracing experiment. First, we develop our framework of plan inference using grammar induction and detail our generative model over hierarchical plans, showing that our theory is a natural approach to predict qualitative aspects of behavior missed by simpler accounts. Then, we analyze

[†]Lightbot (<https://lightbot.com>) is an educational game developed by Danny Yaroslavski, released as a Flash game in 2008 and a mobile application in 2013.

behavioral data from an online experiment using our process-tracing paradigm. We examine qualitative examples and perform model comparison, finding that participant programs are best predicted by our grammar induction model. Analyzing other behavioral signatures unique to our experiment, we also find evidence that supports the idea that people prefer hierarchies that simplify planning.

3.2 BACKGROUND

Our approach to modeling complex, hierarchical planning draws on several distinct literatures: sequence learning, reinforcement learning, and Bayesian program induction. In this section, we briefly summarize the key ideas from these literatures.

3.2.1 SEQUENCE LEARNING

A long-standing question is how sequences are learned by humans and other animals. An influential finding is that sequence memorization is vastly improved when information is grouped into larger *chunks* (Miller, 1956). Methodologically, mental representations of hierarchical structure are often inferred from behavior on the basis of elevated response times (Verwey, 1996), errors (Rosenbaum et al., 1983; Acuna et al., 2014), or decreased speed in motor trajectories (Ramkumar et al., 2016). Recent normative theories of sequence learning explain chunk selection through the optimization of a speed-accuracy trade-off (Dezfouli & Balleine, 2012; Wu et al., 2023), MDL (Planton et al., 2021), and idealized search costs (Ramkumar et al., 2016). There are also theories of action segmentation that model reuse by using a similar framework (an Adaptor Grammar; Johnson et al., 2006) as our approach (Buchsbaum et al., 2015). Another broad approach to modeling sequence learning is statistical learning (Perruchet & Pacton, 2006)—for example, modeling action chunking using a hierarchical non-parametric model of sequence statistics (Éltető et al., 2022; Éltető & Dayan, 2023).

3.2.2 HIERARCHICAL REINFORCEMENT LEARNING

Hierarchical reinforcement learning adapts the framework of reinforcement learning to a setting with hierarchically-structured policies. In one prominent framework, this is formalized by augmenting a task with abstract actions, called “options”, which each consist of a behavioral policy and conditions for initiation and termination (Sutton et al., 1999; Stolle & Precup, 2002; Botvinick et al., 2009). Much of the research on option discovery has focused on *subgoal*-based options, with approaches including partitioning a task into subtasks (Tomov et al., 2020; Solway et al., 2014; McNamee et al., 2016) or identifying “bottleneck” states within the state space (i.e. states that are commonly passed on the way to the ultimate goal; Şimşek & Barto, 2009).

Many approaches have been taken to understand how people choose behavioral hierarchies, including some based on task partitioning (Solway et al., 2014; Tomov et al., 2020), policy compression (Solway et al., 2014; Maisto et al., 2015; Lai et al., 2022), and minimizing planning costs (Huys et al., 2015; Correa et al., 2023a). The approach we introduce in this chapter based on an Adaptor Grammar (Johnson et al., 2006) is similar to a model used to examine planning behavior in prior work (Huys et al., 2015). Studies have investigated error-based learning of hierarchical policies, finding evidence in both neural data (Ribas-Fernandes et al., 2011) and behavioral patterns (Eckstein & Collins, 2020).

3.2.3 BAYESIAN PROGRAM INDUCTION

Bayesian program induction is an approach to inferring programs π consistent with some observed data d (Goodman et al., 2008b; Piantadosi et al., 2012; Lake et al., 2015; Rule et al., 2020). Concretely, by specifying a *prior* probability distribution over programs $p(\pi)$, and the *likelihood* of the observed data under that program $p(d|\pi)$, Bayes’ theorem can be used to define the *posterior* probability of programs

conditioned on the data

$$p(\pi|d) \propto p(d|\pi)p(\pi).$$

This approach has been fruitfully used to model many aspects of cognition, including concept learning (Goodman et al., 2008b; Fränken et al., 2022), number word learning (Piantadosi et al., 2012), writing characters with motor actions (Lake et al., 2015), and sequential decision-making (Maisto et al., 2015; Wingate et al., 2011). While the computational demands of inference usually restrict its application to simple domains, modern machine learning methods like neural networks have made it possible to scale these methods to more difficult domains, while still producing interpretable, human-like representations (Ellis et al., 2021; Poesia & Goodman, 2023). Another crucial strategy for making inference tractable is library learning (Ellis et al., 2021; Poesia & Goodman, 2023; Zhao et al., 2023), where program fragments are reused in order to accelerate inference. Our grammar induction model builds upon this idea.

3.2.4 MINIMUM DESCRIPTION LENGTH

Minimum description length (MDL) is an information-theoretic approach that focuses on minimizing the description length of mental representations and is formally related to Bayesian inference (Chater, 1999). In the setting of sequence learning, an MDL representation is one that provides a minimal, hierarchical compression of the sequence. This approach has been applied broadly in studying sequence learning. For example, Planton et al. (2021) propose that compressed, hierarchical representations of sequences support learning. Maisto et al. (2015) incorporate a prior over hierarchical policies related to the description length of plans under those hierarchies. Lai et al. (2022) extend a theory of policy compression to account for the information-theoretic savings from ignoring perceptual information while executing an action chunk. Solway et al. (2014) note their theory identifies hierarchies that min-

imize the description length of optimal behavior, even though it is primarily formulated as a Bayesian account.

Having introduced these literatures, we now turn to our approach. We draw on ideas from sequence learning and hierarchical reinforcement learning to study the hierarchical structure underlying action sequences, and propose a modeling framework based on principles from program induction and MDL.

3.3 THEORIES OF PROGRAM CREATION IN LIGHTBOT

Our approach to studying complex, hierarchical plans has two key components. First, we use a process-tracing experimental paradigm in order to make explicit the internal hierarchical representations participants use for planning. Instead of asking participants to simply solve the problem, we ask participants to create a hierarchically-structured program that will be interpreted by an agent in the experiment. Second, we model these plans with a framework based on models of grammar learning from computational linguistics (Goldwater et al., 2009; Johnson et al., 2006). In this section, we introduce the experiment and motivate our choice of modeling framework.

Our process-tracing paradigm is based on the popular Lightbot game. The experiment interface and an example program are shown in Fig. 3.1. In this task, research participants control a robot in a gridworld-like environment by supplying a program for it to execute, with the goal of having the robot light all the blue squares. The program is composed by dragging and dropping instructions in the interface shown in Fig. 3.1. Participants create programs that include five primitive actions (`Walk`, `Jump`, `Turn Left`, `Turn Right`, `Activate Light`) as well as calls to subroutines that can be defined and reused. Subroutines are defined in the same way, so they also consist of actions and subroutine calls, which makes it possible to submit programs that correspond to hierarchically-structured plans. When participants submit a program for execution, the robot starts from the initial subroutine and runs instructions in order, so that primitive actions immediately take effect and subroutine calls execute the respective subroutine. Thus, executing a program results in a deterministic sequence of

actions, which we refer to as the *execution trace*. We formally define the task, programs, and their execution in the next section (see Section 3.4).

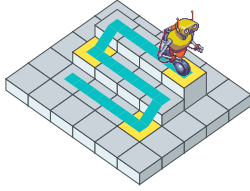
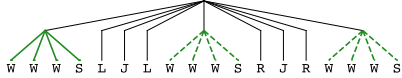
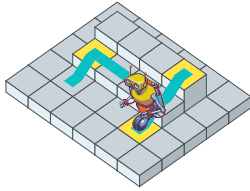
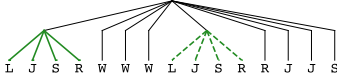
Trace	Program	Step Count	Program Length
		18	13
		15	13

Figure 3.2: Two example programs that solve the task are shown in Fig. 3.1. The top row contains the same program as Fig. 3.1, demonstrating the execution trace, or sequence of resulting actions, in blue. Table rows correspond to distinct programs. Table columns show a program execution trace, a tree representation of the program, the step count (number of actions) resulting from program execution, and the program length. The tree representation of a program shows how the sequence of actions relates to the program structure. Actions are executed in sequence, from left to right. Subroutines are shown with green lines that connect the subroutine call to its constituent instructions. The first use of a subroutine has solid lines, while subsequent uses have dashed lines. In the experiment, participants only see an animation of the robot, not the execution trace in blue. Legend: S: Activate Light, W: Walk, J: Jump, R: Turn Right, L: Turn Left.

In the example in Fig. 3.1, the robot is shown in the start state. The example program in Fig. 3.1 results in an S-shaped execution trace and accomplishes the goal of activating the lights, as shown in the top row of Fig. 3.2. Notably, the program has a subroutine that contains the instructions Walk, Walk, Walk, Activate Light. If the program were written without a subroutine, the program length would be 18, equivalent to the execution trace length. However, by using this subroutine, the program is much shorter, using only 13 instructions. In order to make program structure clearer in this chapter, we display programs as trees, as in the second column of Fig. 3.2, described further in the figure caption.

We formalize the process of planning as Bayesian program induction over hierarchical plans, so the

posterior probability of a plan depends on the likelihood that it solves a task as well as the belief assigned to it *a priori*. By framing planning as an inference problem, we can examine different assumptions about program structure in the form of prior distributions and see how they influence the posterior distribution of programs that solve the task. This is a notable departure from standard formulations of planning, where plans are simply unstructured sequences of actions that solve a task, often biased to be the shortest plan. A key component of the posterior is the likelihood that we assign to a program. Because participants are required to create programs that complete a task, we simply define this likelihood to require that a program reaches the desired goal state.

A prior over programs is the other key component needed to define a posterior. Our modeling approach compares different priors over programs, which we introduce here. The simplest prior ignores program structure, preferring those that result in shorter execution traces. This model corresponds to a softmax choice rule (Luce, 1959) over plans based on the number of resulting actions, similar to standard modeling approaches in reinforcement learning. An alternative prior might prefer shorter programs, a reasonable approach given that program length directly corresponds to the number of motor actions participants take when writing programs. We quantify program length as the *number of instructions* used in the written description of the program, as opposed to the *number of actions* the program yields in the execution trace. We treat program length as description length, and consider this an MDL prior.

Both of these priors are simple and sensible approaches to modeling plan inference in Lightbot, but are inconsistent with some intuitive examples. For example, in Fig. 3.2 the top row features a simple and intuitive example program that contains a single subroutine (Walk, Walk, Walk, Activate Light). This subroutine is used three times and mirrors the repeated environmental structure. The bottom row, by contrast, contains a less intuitive program that uses a subroutine only twice. Despite the more natural feel of the top row program over the bottom row (confirmed by being preferred by experimental participants in the results below), it would not be preferred by either of the priors outlined above.

We introduce a theory to explain this by having past use of a subroutine inform future use—in other words, subroutine choice is biased to reuse subroutines. So, the top program in Fig. 3.2 is more intuitive because it has greater repeated use of the subroutine. Our theory of grammar induction is a prior over Lightbot programs, including subroutines, where past use of subroutines directly informs the probability assigned to their future use. By contrast, the MDL prior assigns the same probability to programs of equal length, equally preferring flat sequences of actions and highly nested programs. By accounting for this bias towards reuse, our prior can explain qualitative patterns in the use of subroutines in human behavioral data, as we explore further below.

3.4 MODELING FRAMEWORK

In this section, we formally define our proposed framework. We first discuss the Lightbot domain and programs, introducing notation and other details. Then, we formalize our framework for hierarchical plan induction, along with the simple approaches based on minimizing trace length and MDL. In addition, we discuss the algorithm we use to perform inference. Finally, we introduce another method for defining a prior over hierarchical plans, which uses a framework from computational linguistics (Johnson et al., 2006) to augment a grammar to support reuse. This framework has been applied to study linguistic phenomena like word segmentation (Goldwater et al., 2009) and inference of discourse structure (Luong et al., 2013), and has also been extended to study forms of generalization that require greater abstraction, like past tense and derivational morphemes in English (O’Donnell, 2015) and in non-linguistic topics like causal concept bootstrapping (Zhao et al., 2023).

3.4.1 TASK FORMALISM

We formalize Lightbot as an undiscounted, deterministic Markov Decision Process (MDP), specified by a state set $\mathcal{S}$, initial state $s_0 \in \mathcal{S}$, action set $\mathcal{A}$, transition function $T : \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{S}$, and reward function $R : \mathcal{S} \times \mathcal{S} \rightarrow \mathbb{R}$. A Lightbot task is defined by a 2-dimensional grid of squares, which each has an associated integer height and type (light square or not). The state set $\mathcal{S}$ has elements that track the en-

environment state (each light is either active or unactive) and the agent state (a location and orientation). The action set $\mathcal{A}$ consists of the five primitive actions (Walk, Jump, Turn Left, Turn Right, Activate Light). The goal states $\mathcal{G} \subset \mathcal{S}$ are those where all light squares are activated.

The transition function T is straightforward: Turn Left and Turn Right change the agent orientation, by rotating either counterclockwise or clockwise, respectively; Activate Light activates a light if present at the agent location. A valid use of Walk and Jump both move the agent forward, changing the location based on the orientation. A valid use of Walk requires the current and next location to have matching heights, while a valid use of Jump requires the next location to either have a height greater by one unit or smaller by any amount. The agent does not move for invalid uses of either action.

The goal in Lightbot is to activate all light squares, so upon reaching a goal state, $g \in \mathcal{G}$, from a non-goal state, $s \notin \mathcal{G}$, the agent is rewarded, $R(s, g) = 1$. Otherwise, the agent receives no reward, $R(s, s') = 0$. The goal states $g \in \mathcal{G}$ are absorbing, so for all actions a , $T(g, a) = g$.

3.4.2 PROGRAM FORMALISM

We now introduce notation for the programs that participants submit in our process-tracing paradigm. We define a *program* π as a tuple of subroutines $(\rho^0, \rho^1, \dots)$, where each subroutine ρ^i consists of a sequence of instructions $(\rho_0^i, \rho_1^i, \dots)$ and has length $|\rho^i|$. Each *instruction* ρ_j^i is either an action in the MDP, so that $\rho_j^i \in \mathcal{A}$, or a *subroutine call* $\rho_j^i \in \{\rho^1, \rho^2, \dots\}$. The program length is the length of all subroutines, $|\pi| = \sum_i |\rho^i|$.

To execute a program, we start at an initial state $s_0 \in \mathcal{S}$ and begin executing the initial subroutine ρ^0 . Because ρ^0 is the initial subroutine, we exclude it from the list of valid subroutine calls. Executing a subroutine ρ^i in turn requires sequentially executing its constituent instructions ρ_j^i . When the current instruction is an action, so $\rho_j^i \in \mathcal{A}$, the action is taken, resulting in a state transition, so $s_{t+1} = T(s_t, a_t = \rho_j^i)$ and $t = t + 1$, and a corresponding reward, $R(s_t, s_{t+1})$. Otherwise, the instruction ρ_j^i is a subroutine call, which does not directly result in a state transition but goes on to execute the

referenced subroutine. When execution of the referenced subroutine is completed, execution returns to the original subroutine that made the call and continues on to the next instruction. The execution of a program can be visualized as a tree (Fig. 3.2), where leaf nodes correspond to executed actions, in order from left to right, and internal nodes correspond to subroutine calls.

Programs are executed until they reach a goal state, which results in an execution trace $\tau(\pi) = (a_0, a_1, \dots, a_{T-1})$ and state trajectory $(s_0, s_1, \dots, s_T)$, where the final state is a goal, $s_T \in \mathcal{G}$. This constraint ensures that programs that reach a goal always halt, even if the program is recursive. The value of a program is $V(\pi) = \sum_t R(s_t, s_{t+1})$. Since there is no reward until the goal is reached, $V(\pi) = 1$ when a program reaches the goal and $V(\pi) = 0$ otherwise. An algorithm for program evaluation is included in the appendix (see Alg. 1).

3.4.3 INFERRING PROGRAMS

We formulate the choice of programs as the result of Bayesian program induction, adapted to this sequential decision setting by recasting planning as inference (Levine, 2018; Wingate et al., 2011; Toussaint & Storkey, 2006). Concretely, we define a posterior distribution over programs that combines a prior distribution over programs $p(\pi)$ with a likelihood that indicates whether a program solves the task.

PROGRAM INDUCTION

We first introduce a binary random variable Ω so that $\Omega = 1$ means a program solves the task. We condition on programs that solve the task in order to define a posterior distribution over programs, $p(\pi \mid \Omega = 1)$. Applying Bayes' theorem, we can define this posterior in terms of a likelihood that reflects whether a program solves the task, $p(\Omega = 1 \mid \pi)$, and a prior over programs, $p(\pi)$,

$$\begin{aligned} p(\pi \mid \Omega = 1) &= Z^{-1} p(\Omega = 1 \mid \pi) p(\pi) \\ &\propto p(\Omega = 1 \mid \pi) p(\pi) \end{aligned}$$

where Z is a normalizing constant $Z = \sum_{\pi'} p(\Omega = 1 \mid \pi')p(\pi')$.

We define the likelihood to identify programs that solve the task, or equivalently, reach the goal. Since the value of a program is an indicator of reaching the goal, we can simply define our likelihood as

$$p(\Omega = 1 \mid \pi) = V(\pi) = \begin{cases} 1 & \text{if } \pi \text{ reaches the goal} \\ 0 & \text{otherwise} \end{cases}.$$

A key benefit of this inference-based formulation is that it can integrate prior beliefs about the structure of programs alongside the requirement that a program solves the task.

APPROXIMATE INFERENCE

Computing the normalizing constant Z requires the intractable task of enumerating all programs. As a tractable alternative, we approximate the normalization constant Z using a finite corpus of programs. To generate these programs, we first search for a corpus of execution traces $(a_0, a_1, \dots)$ that solve a given task. Then, we generate programs consistent with the execution traces using various heuristics. We describe trace search and program generation in the supplemental material.

Given a collection of programs Π , we can compute an approximation to the normalization constant Z_Π by summing over the programs in Π , formally

$$Z_\Pi = \sum_{\pi' \in \Pi} p(\Omega = 1 \mid \pi')p(\pi').$$

This lets us define an approximate posterior

$$p(\pi \mid \Omega = 1) \cong Z_\Pi^{-1} p(\Omega = 1 \mid \pi)p(\pi). \quad (3.1)$$

We note that Z_Π is a lower bound to Z , $Z_\Pi < Z$, since Z is a sum of positive terms and Z_Π is a sum of a subset of those terms.

3.4.4 BASELINE MODELS

Having outlined our framework for Bayesian program induction, we formalize some simple priors over programs. As a baseline, we consider a model that minimizes trace length. This prior ignores program structure and penalizes the resulting trace length

$$p(\pi) \propto \exp \{ -|\tau(\pi)| \} .$$

Another baseline we consider is a prior that minimizes the description length of a program, which we refer to as the MDL prior. We formalize the description length (DL) as the program length, so $DL(\pi) = |\pi|$. The description length is simple and idealized, assuming that all instructions have the same description length. We use this to define a prior, so that shorter lengths are preferred:

$$p(\pi) \propto \exp \{ -DL(\pi) \} .$$

Some prominent accounts of task decomposition have incorporated description length into their objectives to influence choice over subgoals. In one account, minimizing description length guides the choice among task decompositions, used to compress fixed, optimal policies (Solway et al., 2014). In another account, the description length of plans guides subgoal choice in the form of a prior over subgoals in an inference-based hierarchical planner (Maisto et al., 2015).

As noted above, the simple approaches introduced in this section struggle to explain intuitive solutions to Lightbot problems, as in Fig. 3.2. The critical missing piece is that these approaches are insensitive to differences in hierarchical structure observed when holding program length or trace length constant. We fill this explanatory gap with the model we introduce next.

3.4.5 GRAMMAR INDUCTION

In this section, we recast hierarchical planning as grammar induction. Here, we propose a generative model in which new subroutines augment a grammar over actions. The key idea behind our model is that past use of subroutines should inform the probability assigned to future use of subroutines. We formalize this idea using Adaptor Grammars (Johnson et al., 2006).

To start, we define a prior over a single subroutine ρ^i consisting only of low-level actions, so that $\rho_j^i \in \mathcal{A}$ for all j —that is, the subroutine does not contain other subroutines. Assuming a uniform distribution over possible actions $p(\rho_j^i = a) = \frac{1}{|\mathcal{A}|}$, the overall probability of an action sequence is

$$p(\rho^i) = p_{\text{end}} p(\rho_0^i = a) \prod_{j=1} (1 - p_{\text{end}}) p(\rho_j^i = a),$$

with p_{end} the probability of terminating the sequence.

We augment this prior to generate subroutine calls in addition to actions by (1) sampling a subroutine call with probability p_{call} or an action with probability $1 - p_{\text{call}}$ and then (2) either generating a new subroutine for the call or reusing one that was previously defined. Permitting the reuse of subroutines means that the resulting grammar is no longer *context-free*, since the probability of a subroutine varies based on previous subroutine calls. To formalize this kind of long-range dependency, we instead use an *Adaptor Grammar* (Johnson et al., 2006; Huys et al., 2015), which augments a standard context-free grammar by allowing the grammar’s productions to be dependent on the history of previous productions. This provides a formal basis for a generative model of reuse, where previous productions can become more probable in the future. We use a Chinese Restaurant Process (CRP; Aldous, 1985) to adapt subroutine generation for our grammar over Lightbot programs. We briefly highlight some key aspects of this account of subroutine use: new subroutines are generated, subroutines can be reused, and neither the number of subroutines nor the subroutines themselves are predefined. Next, we describe how subroutine calls are sampled in detail.

The CRP is a probability distribution over clusters of data that incorporates a new element by either

assigning it to (1) a new cluster or (2) an existing cluster, biased towards larger clusters. We likewise sample subroutine calls, as either (1) a new subroutine or (2) an existing subroutine, biased towards often-used subroutines. Formally, having already drawn n subroutine calls to m distinct subroutines with a total of n_k calls to subroutine k , we define a distribution over the next subroutine call z_{n+1} , given the history of drawn subroutine calls $z_{1:n}$

$$p(z_{n+1} \mid z_{1:n}) = \begin{cases} \frac{\alpha}{n+\alpha} & \text{if } z_{n+1} = m + 1 \\ \frac{n_k}{n+\alpha} & \text{if } 1 \leq z_{n+1} = k \leq m \end{cases} \quad (3.2)$$

The first case deals with a new subroutine call, which requires that a new subroutine be generated. In the second case, an existing subroutine is selected proportionally to how often its been used in the past, n_k . When an existing subroutine is selected, it is reused and generation of that subroutine is avoided. The CRP has a single parameter, $\alpha \geq 0$, which is usually referred to as the concentration parameter since it controls the relative probability of creating a subroutine—as $\alpha \rightarrow 0$ reuse of an existing subroutine becomes more likely and as $\alpha \rightarrow \infty$ creation of a new subroutine becomes more likely. Since the most probable subroutines are those used most often in the past, the CRP results in so-called *rich-get-richer* dynamics, where frequently used subroutines are most likely in the future. The rich-get-richer dynamics of the CRP are a key prediction that differentiates our grammar induction account from the alternative priors—as noted above, this preference for reuse can explain the example in Fig. 3.2.

Putting these pieces together, the grammar induction prior generates sequences of instructions, terminating with probability p_{end} . Instructions are either subroutine calls with probability p_{call} or actions with probability $1 - p_{call}$. Subroutine calls are drawn from the Adaptor Grammar based on a CRP, with new subroutines generated as an instruction sequence and existing subroutines reused in proportion to the number of past calls. An algorithm for sampling is included in the appendix (see Alg. 2).

In the MDL account, using a subroutine more times can make a program shorter, so a preference for shorter programs can indirectly lead to a preference for subroutine reuse. The grammar induction model has an analogous preference because, in general, it assigns higher probability to programs with

fewer instructions or subroutines. However, a distinctive prediction of the grammar induction model is the explicit preference to use subroutines based on how often they’ve been used in the past, which is above and beyond that of the implicit preference that comes from avoiding generation. While the MDL account only indirectly encourages hierarchical structure, the grammar induction model also does so in a more explicit way by assigning higher probability to hierarchies with greater reuse.

Adaptor grammars have previously been used to model human behavior in the context of action sequences, such as predicting action segmentation (Buchsbbaum et al., 2015) and planning behavior (Huys et al., 2015). Adaptor grammars are also formally related to approaches used in probabilistic programming languages to express non-parametric distributions (Goodman et al., 2008a).

In this section, we introduced a generative model over complex, hierarchical plans for the domain of Lightbot. Our model of Bayesian program induction incorporates a likelihood term for programs that solve a given task. We initially considered two simple priors: a prior that minimizes trace length, and the MDL prior that minimizes program length. Finally, we introduced our grammar induction prior, which develops a probabilistic model of reuse. Central to the grammar induction prior is a bias towards reusing subroutines that have been frequently used in the past. This bias gives the prior a preference among hierarchies based on their degree of reuse, a preference that is missing from the alternative priors we consider. With these models over programs, we now turn to experiments based on the process-tracing paradigm, Lightbot.

3.5 EXPERIMENT

In this section, we detail our process-tracing experiment and use our Bayesian program induction framework to predict the hierarchical plans that participants create.

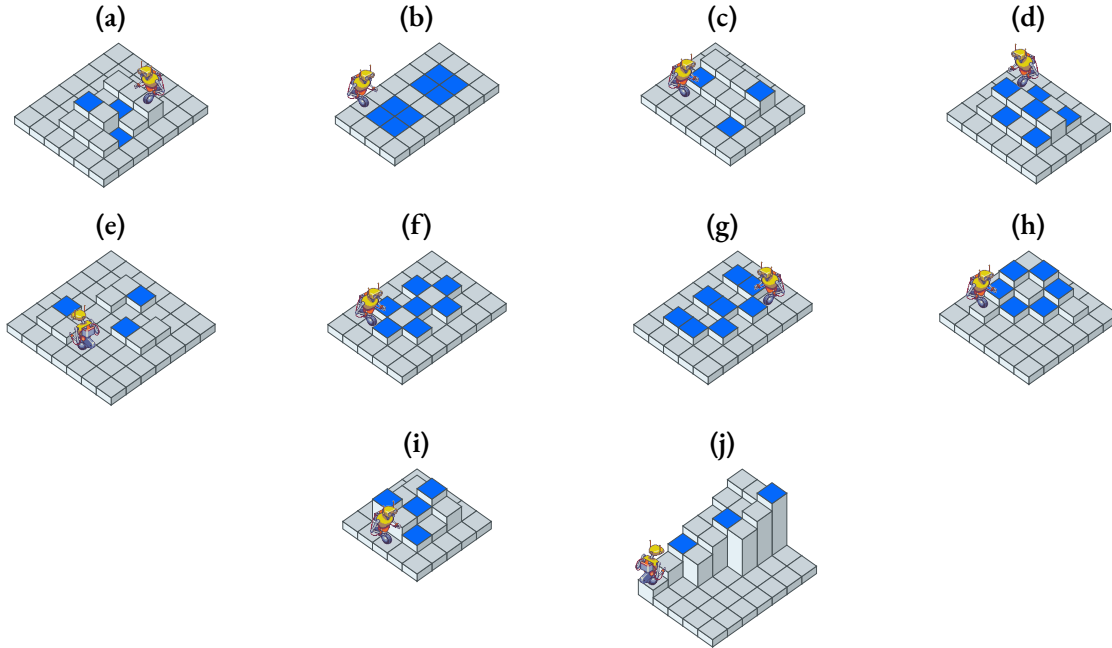


Figure 3.3: The tasks participants completed in the experiment.

3.5.1 METHODS

PROCEDURE

Participants were given an extensive tutorial to ensure they learned how to control Lightbot using the five primitive actions, as well as to ensure they understood how to create and use programs. The tutorial described the functionality of each action, included demonstrations, and the opportunity to solve practice problems to ensure understanding.

In the remainder of the experiment, participants wrote programs to solve 10 Lightbot tasks and were incentivized to write short programs to encourage subroutine use. Participants received instructions about the incentive and were shown multiple examples of subroutine use. The program editing interface had a counter showing program length (Fig. 3.1). Participants who wrote the shortest program for a task received the full bonus of \$0.40 and those who wrote a longer program received a proportionally smaller bonus, calculated after the experiment to ensure the average per-task bonus across participants was \$0.20. Since participants solved 10 tasks, the total bonus ranged from \$0.00 to \$4.00, with an av-

erage bonus of \$2.00. Since the task incentivizes short programs, this could bias participants towards behavior more consistent with the MDL prior.

After being instructed about the incentives, participants went on to write programs to solve each of 10 tasks (Fig. 3.3), with the program editing interface shown in Fig. 3.1. During each task, participants were given the option to skip the problem and forfeit a bonus for the current problem after 3.5 minutes elapsed. In a closing survey, participants were asked about their programming experience and whether they had previously played games similar to the experiment.

PARTICIPANTS

The experiment was run on the Prolific platform. Participants consented to the experimental procedures beforehand, as approved by the Institutional Review Board of Princeton University. A total of 193 participants completed the entire study, and 171 participants (89%) were included because they satisfied the inclusion criteria, having no more than 3 college courses related to computer programming. As noted, participants could skip tasks after 3.5 minutes. 148 participants (87%) skipped no tasks, 16 participants (9%) skipped 1 tasks, 3 participants (2%) skipped 2 tasks, and 4 participants (2%) skipped 3+ tasks. In the below analysis, we analyze all fully completed tasks, even from participants who may have skipped other tasks.

As noted above, participants were asked about their programming experience. In response to “How much experience do you have with computer programming?” 65% of participants responded with “None,” 24% responded with “Between 1 and 3 college courses (or equivalent),” and 11% responded with “More than 3 college courses (or equivalent).” In response to “Have you played Lightbot or another similar programming game before?” 98% of participants responded with “No” and 2% responded with “Yes”.

PARAMETER FITTING

Given a behavioral dataset of Lightbot programs π and the number of times they were produced by research participants n_π , we want to optimize for parameters θ in the model based on the posterior probability they assign to our observed behavior

$$\mathcal{L}(\theta) = \prod_{\pi} p(\pi \mid \Omega = 1, \theta)^{n_\pi}.$$

For models with nuisance parameters ψ , we marginalize them out

$$\mathcal{L}(\theta) = \int_{\psi} \prod_{\pi} p(\pi \mid \Omega = 1, \theta, \psi)^{n_\pi} p(\psi) d\psi.$$

In both these equations, parameters are explicitly passed to the posterior for clarity, but were left implicit in Section 3.4.3. We estimate parameters by maximizing the probability assigned to observed behavior, so our objective for the parameters is

$$\operatorname{argmax}_{\theta} \log\{\mathcal{L}(\theta)\}$$

Because of the intractability of computing the posterior, $p(\pi \mid \Omega = 1)$, we instead compute the approximation in Eq. 3.1.

To predict participant programs, we use six models. One baseline account is a null model of *random choice*, where the prior is uninformative, $p(\pi) \propto 1$, so uniform probability is assigned to programs. We also test the *step cost*, *MDL*, and *grammar induction* priors introduced above. Finally, given the efficacy of the step cost prior in explaining behavior, we combine it with the other two priors for the *grammar induction + step cost* and *MDL + step cost* models. These combined models define a new composite prior that is the product of the two original priors.

Parameters are displayed with fitted values in a table below, but are briefly summarized here. A rela-

tive weight for the prior is fit, which is used to scale the log prior, $\log p(\pi)$, and are named as follows: β_{StepCost} , β_{MDL} , $\beta_{\text{GrammarInduction}}$. For the combined models that include the step cost prior, we also fit a weight of β_{StepCost} . The grammar induction model additionally fits the concentration parameter of the CRP, α , and the probability of a subroutine, p_{call} . In addition, the grammar induction model has the nuisance parameter $\psi = \{p_{\text{end}}\}$, which we approximately marginalize for the values $p_{\text{end}} \in \{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9\}$, reflecting a uniform prior.

Parameter fitting was performed once from an initialization using a default value, and four times with sampled values. The parameters with best fit from these five parameter fitting executions are used to make predictions below. Probabilities (e.g., p_{call}) were constrained to be in $(0, 1)$ and either fit from an initial default value of $\frac{1}{2}$ or a value sampled uniformly from $(0, 1)$. All other parameters were constrained to be positive and either fit from an initial default value of 1 or a value sampled from an *Exponential*(2) distribution.

PREPROCESSING OF PARTICIPANT PROGRAMS

In order to facilitate analysis, programs are preprocessed to remove nuisance differences between participant programs. The steps are as follows: ensuring subroutines are used as often as possible; inlining subroutines only called once or only consisting of one instruction; ensuring subroutines have a canonical ordering determined by their execution order; removing subroutines that are never called; removing instructions that never have an effect (i.e. we remove actions ρ_j^i if for all t where $a_t = \rho_j^i$ the action results in a transition to the same state, so $s_t = s_{t+1} = T(s_t, a_t)$); ordering turns and lights to match the constraints on program search (see Appendix B). Removing these small and subtle issues in programs makes it possible for analysis to focus on the patterns driving behavior while ignoring arbitrary or meaningless variation.

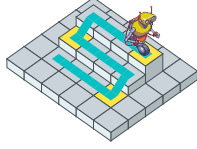
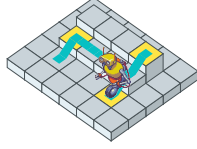
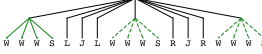
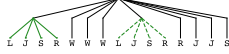
trace		
program		
participant count	59	1
step count	18	15
program length	13	13
grammar induction prior (log)	-31.97	-33.17

Figure 3.4: Comparing a common participant program (left) to an uncommon one (right) for one task, revisiting the example from Fig. 3.2. The two programs have equivalent program lengths. The more common program has a larger step count. The grammar induction prior can explain participant preference for the left program. Subroutines are shown with green lines that connect the subroutine call to its constituent instructions. First use of a subroutine has solid lines, while subsequent uses have dashed lines. The log of the grammar induction prior is shown, under parameters $\alpha = 1$, $p_{end} = \frac{1}{10}$, and $p_{call} = \frac{1}{2}$.

DATA AVAILABILITY

Code for the experiment is available at

<https://github.com/cgc/cocosci-lightbot/tree/v0.4>. The data and analysis code are available at <https://github.com/cgc/lightbot-grammar-induction>.

3.5.2 RESULTS

PARTICIPANT PROGRAMS ARE INCONSISTENT WITH ALTERNATIVE ACCOUNTS

We first qualitatively analyze some example participant programs in Figs. 3.4-3.6, comparing common programs to other programs that were uncommon. The programs in these figures are examples that rule out a straightforward account of human choice based on solely optimizing step cost or program length.

We first examine Fig. 3.4, which revisits the programs introduced in Fig. 3.2. Both programs have

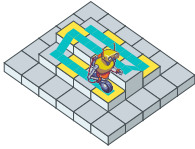
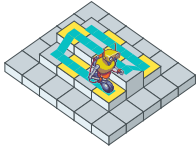
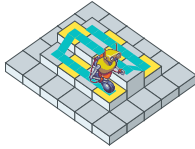
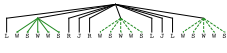
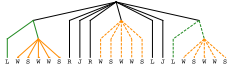
trace			
program			
part. count	50	3	3
step count	22	22	22
prog. length	15	15	18
grammar induction prior (log)	-36.78	-39.45	-45.21

Figure 3.5: Examining a common program (left) to two uncommon programs (middle, right) that all share the same trace. Here, programs are matched by step count, since they share a trace. Participant choice is not explained by program length, but can be explained by the grammar induction model. See Fig. 3.4 for more detail about the figure.

the same length, but differ in their cost. In particular, the most common program (left) has a large cost, while the uncommon program (right) has a smaller cost. This pattern of choice can't be explained solely on the basis of either program length or cost, but is consistent with the predictions of the grammar induction prior: The common program has greater subroutine use, which has higher probability under the grammar induction prior, due to the preference for reuse in the CRP.

We now examine cases where the trace is held constant, but the program varies. By examining a fixed trace, we can directly compare the structural features of programs while holding action-related features constant, such as step cost. For example, Fig. 3.5 shows three programs that generate the most common trace for the problem shown. The common program (left) has a single subroutine. Notably, this program can be written in several ways since two of the three uses of a subroutine are preceded by the instruction `Turn Left`. This preceding instruction can be incorporated by creating an additional subroutine (middle). This program has the same program length, but is written by fewer participants. Under the grammar induction prior, this program is less probable because it has less reuse. An intermediate program between these two is also shown (right), where the original subroutine is instead rewrit-

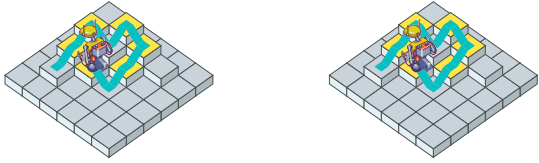
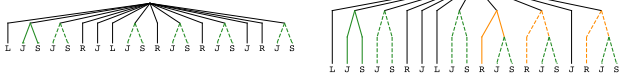
trace		
program		
participant count	19	3
step count	20	20
program length	16	15
grammar induction prior (log)	-35.06	-37.49

Figure 3.6: A case where a common participant program is inconsistent with the predictions of program length. As in Fig. 3.5, the programs have the same trace, so they are matched by step cost. In contrast, now the uncommon figure has a shorter program length. The grammar induction prior can predict this pattern of behavior because of the bias towards reuse. See Fig. 3.4 for more detail about the figure.

ten to include the preceding instruction, reducing the number of places the subroutine can be called.

Fig. 3.6 shows a similar example of programs that generate the most common trace, for a different task. As in the previous example, the common program (left) has a shorter subroutine that appears more often, while the uncommon program (right) has more subroutines that are used less often. Notably, in this case the common program is actually *longer*. The grammar induction prior can capture this pattern for the same reason as above—by way of the CRP, it prefers greater reuse of a single subroutine.

Taken together, these qualitative results challenge a simple account of choice solely on the basis of cost or program length. This is particularly notable since participants have a monetary incentive to minimize the length of their programs in the experiment. Distinctive about the grammar induction account is that, above and beyond the length savings associated with the reuse of a subroutine, we expect additional preference for reuse of subroutines, as predicted by the rich-get-richer dynamics of reuse from the CRP. Next, we move to a quantitative comparison of models in predicting behavior.

Table 3.1: Table of fitted parameters for all models.

Model	Parameter	Fitted Value
MDL	β_{MDL}	0.98
grammar induction	α	4.23
	$\beta_{\text{GrammarInduction}}$	0.48
	p_{call}	0.12
step cost	β_{StepCost}	1.31
MDL + step cost	β_{MDL}	0.53
	β_{StepCost}	0.85
grammar induction + step cost	α	2.52
	$\beta_{\text{GrammarInduction}}$	0.38
	p_{call}	0.09
	β_{StepCost}	0.52

PREDICTING PARTICIPANT PROGRAMS

To more comprehensively examine human behavior, we fit models to the full set of programs people wrote, estimating parameters with maximum likelihood estimation using the procedures described above (see Table 3.1 for fitted parameters). We compare models by using the Bayesian information criterion (BIC; Schwarz, 1978). We first compare the three priors, based on the model BICs shown in Fig. 3.7a. All improve on a random choice model (Likelihood-ratio test for step cost: $\chi^2(1) = 11658.3$, $p < .001$, MDL: $\chi^2(1) = 9844.8$, $p < .001$, grammar induction: $\chi^2(3) = 16454.7$, $p < .001$), and the best fit to behavior is the grammar induction model based on the BIC. Surprisingly, the step cost model is a better fit to behavior than the MDL model—this could mean that participants first identified short traces and then compressed them. In order to control for the influence of this kind of strategy, we also include the combined models described above, in order to compare the MDL and grammar induction models. Compared to a baseline step cost model, we still find an improvement in fit in the MDL + step cost (Likelihood-ratio test, $\chi^2(1) = 1226.6$, $p < .001$) and grammar induction + step

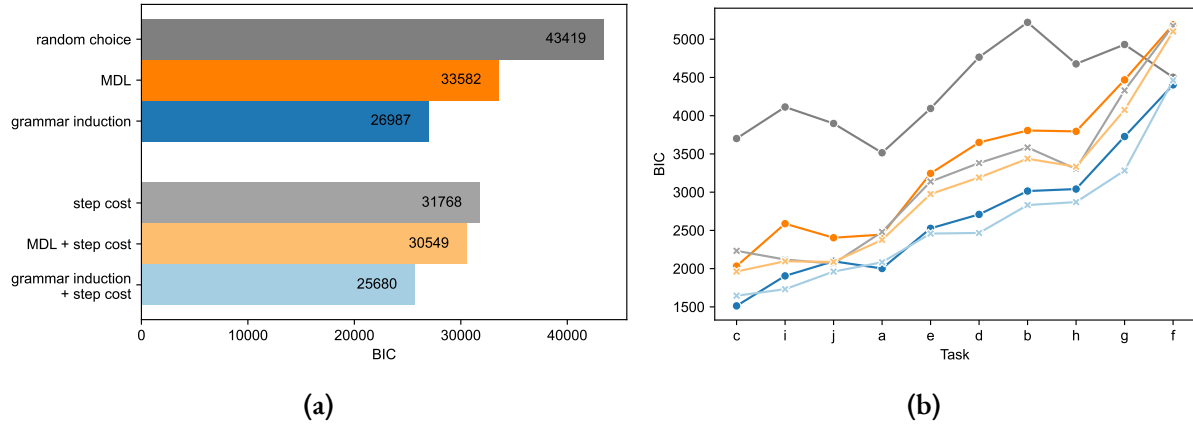


Figure 3.7: Model comparison of accounts of participant program creation. a) Plot of BIC of experimental data as predicted by each of the models, after parameter fitting. Models with smaller BIC are a better account of behavior. b) BIC of data under each model, but split by task. Parameters are the same as in a), so they are the best fit for all tasks. The color of the models is also the same as in a). Models with step cost are marked with a cross, and those without step cost are marked with a point. Task letter is a reference to the subfigure in Fig. 3.3. Tasks are sorted by BIC for the overall best-fitting model, grammar induction + step cost.

cost (Likelihood-ratio test, $\chi^2(3) = 6110.6, p < .001$) models. Among all tested models, the grammar induction + step cost model is the most predictive of behavior.

Holding parameters fixed and examining these model predictions on a per-task basis in Fig. 3.7b, we find patterns that are broadly consistent with the group summary. In particular, we find that one of the grammar induction models is still the best predictor for each individual task, compared to the alternative accounts.

COMMON PROGRAMS ARE EASIER TO WRITE

The following analyses explore how judicious use of hierarchy can make learning and planning easier. One way to quantify the difficulty of planning is the length of time it takes participants to complete a task, a measure that is not specific to our process-tracing paradigm. We find that participants who wrote more common programs were faster at writing their programs ($\beta = -.008, \chi^2(1) = 69.73, p < .001$, Fig. 3.8a). This suggests that people prefer hierarchies that are simpler to use for planning, consistent with prior findings. Another measure of difficulty, which is uniquely measurable in our paradigm, is the number of times participants test the execution of their program in the editor. We

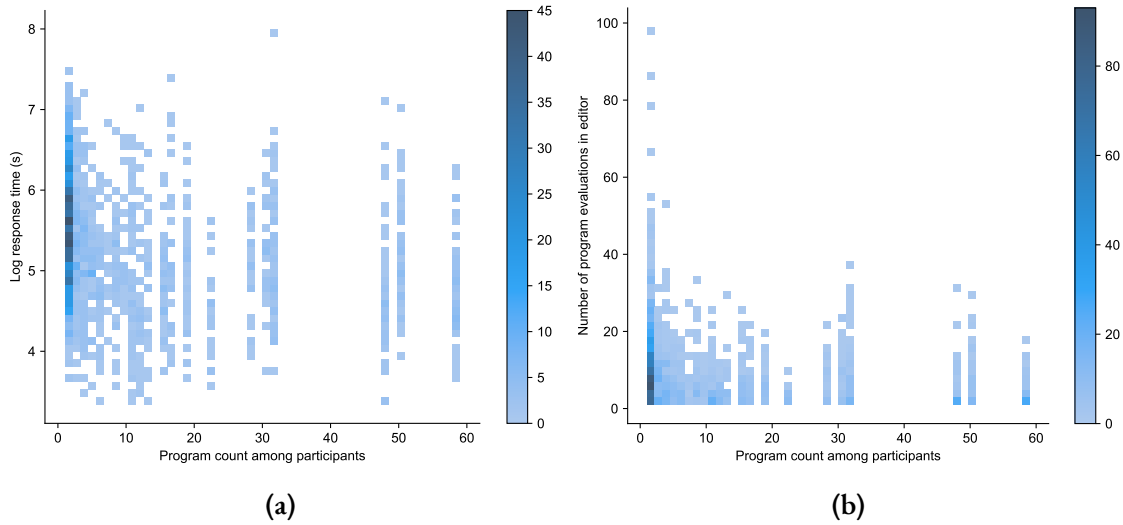


Figure 3.8: Participants write common programs faster and with fewer program evaluations. Bivariate histogram of task responses for each participant on each task, with responses binned along horizontal axis by program count and along vertical axis by either a) log-transformed response times (s) or b) the number of program evaluations. A related regression analysis is reported in the main text, which uses program count to predict response times and number of program evaluations while controlling for program length and cost, and finds results consistent with these plots.

found that, in the process of writing their programs, those with more common programs executed their program fewer times ($\beta = -.053$, $\chi^2(1) = 16.48$, $p < .001$, Fig. 3.8b). So, people prefer hierarchies that are easier to reason about—in this case, people writing common programs can avoid explicit program execution in the editor, perhaps by instead relying on mental simulation.

These models are hierarchical linear mixed-effects models fit using `lmer`. Models were compared by using a likelihood-ratio test to see if the addition of a regressor for program count would improve model fit when added to a baseline model. The baseline model was used to predict either the log response time or the number of program evaluations across tasks and participants. The predictors in the baseline model were an intercept, program cost, and program length as fixed effects along with per-participant and per-task random intercepts.

3.6 DISCUSSION

In this chapter, we studied the human ability to construct complex, hierarchical plans. In order to do so, we used a process-tracing experimental paradigm to collect data about hierarchical plans and developed a framework for program inference to analyze these plans. We ran a behavioral experiment and found that our model of grammar induction was better at predicting behavior compared to alternative accounts that simply assumed participants were efficient as measured by program length or trace length. Key to the grammar induction model is the idea that previous use of subroutines should inform future use. In addition, we used our process-tracing paradigm to examine how the choice of hierarchy can simplify planning. We found that participants who wrote common programs were faster at solving the task and required less use of program execution in the editor.

Our model was motivated by the idea that previous usage should bias reuse. But when is this bias towards reuse sensible? It might be a natural solution when tasks typically have repeated subtasks, since it might lead to more efficient planning or learning (Wen et al., 2020). Another explanation could rest on the theoretical observation that the CRP is well-approximated by a particular kind of information-theoretic cost (Dasgupta & Griffiths, 2022)—this would correspond to a representational cost for the distribution over subroutine calls. One final rationale relates to the idealized computational complexity of search, which is influenced by two factors: the possible choices at each state and the depth of a solution. Abstract actions can accelerate search by making it possible to rapidly reach deeper parts of the search tree. However, they could negatively impact the complexity of search by increasing the number of possible choices. A bias towards reuse could mitigate this increase in complexity by focusing attention on promising subroutines (c.f. Éltető & Dayan, 2023). What psychological mechanisms could implement a bias towards reuse? Our theory could be cognitively implemented by some kind of learning or memorization—it may reflect a simple bias towards recently considered subroutines or could instead reflect an online process of inferring appropriate subroutines based on prior history. In this chapter, we refrained from considering these details in any depth, but simply note these rationalizations

and potential mechanisms.

Since our paradigm is a process-tracing paradigm, we cannot be sure about the relationship between the explicit representation we can measure (the hierarchical plans people submit) and the internal representations people use to act. In contrast to standard approaches for sequence learning, participants never perform the sequence directly, and instead must simulate it mentally or watch it be executed by the robot. In addition, participants are not trained on a fixed sequence; they instead solve the task a single time and simply submit their program. Because behavior in our experiment is not driven by bottom-up statistics in the same way as sequential learning, our behavioral paradigm might isolate a distinct aspect of behavior, namely structural biases of hierarchical planning.

Our approach is a computational-level account (Marr, 1982), which naturally leads to the question of the algorithmic processes people use to implement this model, particularly because of the intractability of Bayesian inference. Many existing approaches to inference explore how varying the parameterization of inference algorithms can lead to a human-like bias. For example, some studies have explored how low-capacity algorithms might recapitulate behavior (Daw & Courville, 2007; Lake & Piantadosi, 2020) or how different proposal distributions could explain auto-correlation in participant hypotheses (Fränken et al., 2022). These results are often interpreted as evidence that human-like biases might arise from rational use of cognitive resources. Future research could pursue process models that are more consistent with findings in the sequence learning literature—for example, it has been observed that larger action sequences might form through concatenation of existing chunks (Tosatto et al., 2022), which could be used to guide the model of program creation. In general, future work could extend our model to investigate how process-level and algorithmic changes can better align with observed behavior. In contrast, the MDL account we compare to is not formulated as a probabilistic model, so it requires non-trivial effort to adapt it to an inference-based algorithm.

A related concern that is agnostic to the inference algorithm is the computational cost of evaluating the terms that comprise the unnormalized posterior, namely the prior and likelihood. In particular, the likelihood in our model requires evaluating a program in the task. Evaluating a policy in the general

case of stochastic environments requires integrating over all possible outcomes which can be computationally costly, driving people to use heuristics when estimating utilities (Lieder et al., 2018). While participants have extensive training on the effect of instructions in Lightbot and have access to a program simulator, effectively using mental resources and available time requires judicious use of simulation (Hamrick et al., 2015; Ullman & Wang, 2023). We found that the most common programs participants wrote required less program evaluation, which suggests that the computational cost of evaluating programs may influence how participants choose among programs. A previous study used the Lightbot domain to examine how execution-related properties of programs influence prior beliefs about programs (Ho et al., 2018)—a natural extension of that analysis could examine whether these computational properties influence the process of search directly, or only by way of influencing program evaluation. Future work could use explicit representations of hierarchy, like in our experiment, in order to finely probe how differences in hierarchy influence mental simulation.

Hierarchy is a key organizational strategy that humans use to structure their behavior in order to make planning and learning efficient (Newell & Simon, 1972; Botvinick et al., 2009), but is difficult to study, requiring indirect measures to infer internal hierarchical representations (Rosenbaum et al., 1983; Verwey, 1996; Huys et al., 2015). In this chapter, we used a process-tracing paradigm to make the hierarchical structure that guides behavior observable, which allowed us to identify that people have a bias towards reuse captured by our generative model of grammar induction. We hope our approach can inspire future efforts to externalize the representations underlying complex mental processes such as planning.

Chapter 4

Program-based strategy induction for reinforcement learning *

4.1 INTRODUCTION

Classic models of reinforcement learning (RL) often assume that humans and other animals incrementally estimate expected rewards or other continuously-varying decision variables, and make noisy decisions based on them (Rescorla & Wagner, 1972; Sutton & Barto, 2018). However, there is increased recognition that the learning strategies actually adopted by humans (and even animals) deviate qualitatively from these models and instead reflect more discrete, idiosyncratic, and perhaps explicit heuristics: elaborating simple state-switching strategies like win-stay, lose-shift (Iigaya et al., 2018; Lau & Glimcher, 2005) and reflecting memories of individual trials rather than summaries (Collins & Frank, 2012; Plonsky et al., 2015; Duncan & Shohamy, 2016; Bornstein et al., 2017). Despite individual attempts to capture these behaviors by elaborating classic models, there is no general formalism spanning the space of such strategies or predicting what variants organisms adopt in particular situations, and how.

Recent work on computational approaches to strategy discovery has started to engage with these

*This chapter is based on the following conference proceedings:
Correa, C. G., Griffiths, T. L., & Daw, N. D. (2024). Program-based strategy induction for reinforcement learning. In *Proceedings of the 46th Annual Conference of the Cognitive Science Society*.

questions. For example, meta-RL has been used to train recurrent neural networks (RNNs) to have recurrent dynamics that implement a learning process (Duan et al., 2016; Wang et al., 2018) which approximate decision-making strategies that are either task-optimized (Ortega et al., 2019) or supervised to distill organisms’ learning behavior (Dezfouli et al., 2019; Ji-An et al., 2023; Miller et al., 2023). However, this approach is lacking in a number of respects. First, neural networks require considerable effort to interpret (making them awkward for theory discovery) and are data- and compute-intensive to optimize (making them unlikely candidates for a process model of how organisms discover strategies). Relatedly, it is not clear that they represent the most appropriate strategy space over which to conduct meta-learning for these purposes. While the divergence of task-optimized RNNs from observed behavior might be addressed by introducing resource costs (Moskovitz et al., 2023; Binz & Schulz, 2022), capacity-limited RNNs have low-dimensional continuous state spaces that do not capture the types of memory-dependent dynamics discussed above.

Accordingly, we propose instead to use Bayesian program induction to identify program-structured strategies for meta-learning in RL problems. This builds on approaches used to study cognitive representations (Goodman et al., 2008b; Yang & Piantadosi, 2022; Ellis et al., 2022) and discover machine learning algorithms (Real et al., 2020) that have been adapted to sequential decision-making (Toussaint & Storkey, 2006; Wingate et al., 2011). Our Bayesian framework has a natural interpretation as a resource-rational approach (Lieder & Griffiths, 2020) by letting the cognitive cost of a program correspond to its description length, so that simpler strategies are more probable under the prior but more effective strategies are more likely to result in optimal behavior under the likelihood. Importantly, we can explore the relationship between previously-reported behavior and the strategies resulting from different trade-offs between effectiveness and program simplicity. By focusing on program-structured strategies for RL, we explore a broader space of strategies than has previously been considered. These programs are interpretable and, since they are small and can be extended with modularity and reuse (Yang & Piantadosi, 2022; Ellis et al., 2021), may also lead to a process-level model of strategy induction.

Using this approach, we identify a range of strategies in various bandit tasks, accounting for some previously observed behavioral phenomena that deviate from classical incremental learning. We find instances where strategies focus exclusively on reward instead of omission, resembling behavioral signatures in rodents (Parker et al., 2016) and asymmetric learning more broadly (Palminteri, 2023). We show how this approach supports adaptive, horizon-specific randomness in exploration, as previously observed (Wilson et al., 2014). We also find strategies that switch between discrete decision states focused on either exploiting known options or exploring new options, as in previous studies (Ebitz et al., 2018). Our framework provides an interpretable alternative to existing methods for strategy discovery, which we hope can provide new insights in other domains.

4.2 PROGRAM-BASED STRATEGY INDUCTION

We formulate strategies as programs and use sampling-based Bayesian inference to find programs that are both simple and effective. We first introduce a formalism for tasks and strategies, then describe our framework for inference, and close with implementation details.

4.2.1 TASKS AND STRATEGIES

We study finite-horizon tasks where at every time step t an agent takes an action a_t , makes an observation o_t , and receives a reward r_t , resulting over time in a history of agent-environment interactions $h_t = (a_{1:t}, o_{1:t}, r_{1:t})$. Observations may only provide partial information about the state of the task (Kaelbling et al., 1998), so future observations and rewards depend on the full history of agent-environment interactions, $p(o_{t+1}, r_{t+1} \mid h_t, a_{t+1})$. Agents are modeled as $p(a_{t+1} \mid h_t)$, producing a task-specific distribution of histories $p(h_{t+1} \mid h_t) = p(a_{t+1} \mid h_t)p(o_{t+1}, r_{t+1} \mid h_t, a_{t+1})$.

Strategies require performing two computations at every time step: updating internal representations to incorporate new observations and rewards, and using that updated internal representation to act. Formally, strategies generate an updated memory m_{t+1} , based on information from the previous trial (memory m_t , action a_t , observation o_t , and reward r_t), which we refer to as the state update func-

tion f ,

$$m_{t+1} = f(m_t, a_t, o_t, r_t). \quad (4.1)$$

Choice behavior is driven by the updated memory m_{t+1} and, for convenience, information from the previous trial, by way of a policy function g that returns action preferences that are exponentiated,

$$p(a_{t+1} \mid h_t) \propto \exp \{g(a_{t+1}, m_{t+1}, a_t, o_t, r_t)\}. \quad (4.2)$$

A completely specified strategy $\pi = (m_1, q_1, f, g)$ also requires an initial memory, m_1 , and action preferences, $p(a_1) \propto \exp \{q_1(a_1)\}$, for the first time step. At subsequent time steps, memory is first updated with Eq. 4.1 and then action is taken with Eq. 4.2. Agents following a strategy are evaluated based on the expected sum of rewards they accrue over T steps,

$$V(\pi) = \mathbb{E}_{h_T} \sum_{t=1}^T r_t.$$

4.2.2 INFERENCE

We use Bayesian program induction to search the space of strategies with a generative prior over strategies and a likelihood based on the effectiveness of the strategy.

Each component of a strategy is a program, a composition of the primitive operations defined in Table 4.1. The primitives correspond to simple arithmetic, logic, and vectors, as well as primitives specific to decision-making, such as previous actions, rewards, and action distributions. For simplicity, we let agent states be vectors of length 4. The space of programs consists of all valid combinations of primitive operations that return the appropriate type (either a vector for m_1 and f , or action probabilities for q_1 and g), which corresponds to a context-free grammar. The prior, $p(\pi)$, assumes expansions in the grammar are sampled uniformly at random, except terminals (like integers and inputs) are 8 times more likely than non-terminals in order to avoid excessive nesting. The integers, from 1 onward, are dis-

Table 4.1: Primitive operations for strategies.

Primitives	Description
Arithmetic, Logic	
0, ..., 49	Integers from 0 to 49 (inclusive)
+, *	Addition, multiplication
-, 1 / (x)	Negation, multiplicative inverse
<, ==	Less than, equals
&&, , !	And, or, negation
if (c, x, y)	Returns x if condition c is true, y otherwise
Vectors	
vec_full(x)	A vector filled with the value x
vec_n(x1, ..., xn)	A vector where the first n entries are supplied and others are 0, e.g., $\text{vec_2}(x, y) = [x, y, 0, 0]$
v[i]	Returns i th entry of v
assign(v, i, x)	Updated copy of v, with $v[i] = x$
add_assign(v, i, x)	Updated copy of v, with $v[i] = v[i] + x$
Inputs	
prev_action	Previous action, a_t
reward	Previous reward, r_t
state	Memory from previous trial m_t for f or current trial m_{t+1} for g
Action probabilities	
logit(l)	For two-action tasks, $l = \log \frac{p(a=0)}{p(a=1)}$
softmax(w, v)	Returns softmax of values in v scaled by w
action(a)	Takes action a
argmax(v)	Takes action with earliest, maximum value in v

tributed according to a geometric distribution with probability 0.5. We represent actions with integers. Invalid programs (non-integral or out of bounds actions/indices, computing 1/0) were assigned a value of $-\infty$. Programs that compute starting values (m_1, q_1) have a restricted set of primitives, excluding conditionals, vector indexing, and other input values.

Following the standard approach for planning by inference (Toussaint & Storkey, 2006; Wingate et al., 2011; Levine, 2018), we formulate our likelihood by introducing a Bernoulli-distributed random

variable Ω that indicates the optimality of a strategy, so that

$$p(\Omega = 1 \mid \pi) \propto \exp \{ \beta V(\pi) \},$$

with β a positive weight on the value. Notably, optimal strategies maximize this likelihood. Combining the likelihood and prior, the posterior probability that programs are optimal is

$$p(\pi \mid \Omega = 1) \propto \exp \{ \beta V(\pi) \} p(\pi). \quad (4.3)$$

The value weight β controls the relative contribution of simplicity and task performance, so that simpler solutions are preferred when $\beta \rightarrow 0$ and more effective solutions are preferred when $\beta \rightarrow \infty$.

For inference, we use Markov chain Monte Carlo (MCMC) with Metropolis-Hastings acceptance to obtain samples from the posterior, as in prior work (Yang & Piantadosi, 2022; Goodman et al., 2008b). We run five chains for 5×10^5 steps per weight β , which are [100, 300, 1000, 3000, 10000, 30000] for the Wilson et al. (2014) task and [10, 30, 100, 300, 1000, 3000] for others. Proposal distributions are subtree-regeneration (Goodman et al., 2008b), and resampling a primitive (retaining arguments), swapping arguments, inserting a subtree between a parent and child, and deleting a subtree while retaining an ancestor (Yang & Piantadosi, 2022). A subset of m_1, q_1, f, g was proposed to, with each program independently sampled for inclusion (probability $\frac{1}{10}$ for m_1 and q_1 and $\frac{1}{5}$ for f and g), avoiding empty subsets by always including a program sampled proportionally to inclusion probabilities. Our implementation is built on the open-source Fleet library used in previous studies (Yang & Piantadosi, 2022).

Value was estimated from 2×10^3 rollouts for the Ebitz et al. (2018) task and 10^4 rollouts in other cases, using expected immediate rewards as well as fixed environmental and agent seeds per chain to decrease variance. To facilitate comparison across chains, we compare normalized value so that chance behavior corresponds to 0 and behavior guided by an oracle (i.e. complete knowledge of the reward probabilities) corresponds to 1. Pareto frontiers were computed by finding the maximal strategies in the space defined by normalized value and the prior, using the the top 100 strategies from each chain.

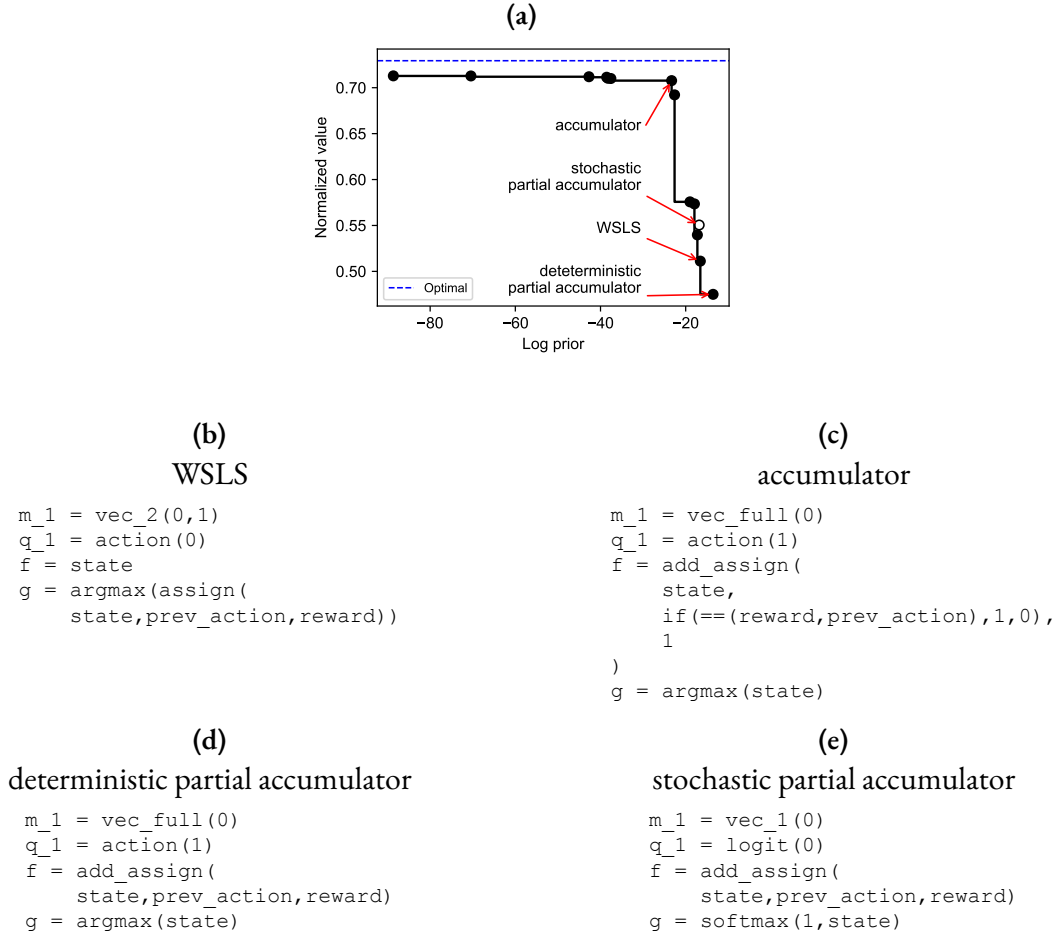


Figure 4.1: Strategies for the two-armed bandit with stationary, Bernoulli rewards. a) The Pareto frontier of deterministic strategies, which are maximal points in the space defined by the normalized value and prior. Points correspond to individual strategies. One stochastic strategy described in the text is included (unfilled point) for comparison. Value is normalized so that chance behavior has a value of 0 and behavior guided by an oracle has a value of 1. Also shown is the Bayes-optimal solution to the task (dotted line). Example strategies are shown in b-e), marked in a), and described in the text. Strategies $\pi = (m_1, q_1, f, g)$ consist of an initial memory state m_1 , initial action preferences q_1 , state update function f , and policy function g , which are further described in Section 4.2.1.

4.3 EXPLORING STRATEGIES FOR A SIMPLE BANDIT TASK

We first survey the strategies discovered for a two-armed bandit, with a finite horizon of 20 trials and stationary, Bernoulli reward probabilities drawn from a uniform distribution. The simplicity of this domain makes it possible to compute the value of strategies exactly. It is also possible to compare to

the optimal solution, which has a simple analytic solution that can be reasonably computed for this horizon.

We first examine strategies when value is exactly computed, restricting attention to deterministic policies for computational reasons. We focus on the Pareto frontier defined by simplicity and utility, that is, solutions that are local maxima in the space defined by simplicity and utility (Fig. 4.1a). Overall, the model exposes a progression of more elaborate and more effective strategies parameterized by this trade-off.

One simple solution we identify is win-stay, lose-shift (WSLS; Fig. 4.1b), where an action is repeated if the previous trial was rewarded, or an alternative action is selected if the previous trial was not rewarded (Robbins, 1952). This solution has a trivial state update f that simply passes a starting vector onward, with the strategy instead implemented entirely in the policy function g . For the sake of clarity, we briefly walk through the evaluation of this strategy. When `prev_action=0` and `reward=1`, the result of `assign` is `[1, 1]`. Since `argmax` returns the earliest maximum value, the subsequent action is 0, so winning results in repeating the action 0. On the other hand, if `reward=0`, the result of `assign` is `[0, 1]`, so `argmax` would result in the action 1, meaning that a loss resulted in a switch. The program behaves similarly for the other action. While a more human-readable implementation might use a conditional, this implementation's careful use of indices avoids the need for a conditional and happens to be more probable under the prior.

A much more effective strategy accumulates the evidence indicated by rewards, so that wins increase the preference for an arm and losses decrease it. The algorithm in Fig. 4.1c implements this by incrementing the state at a conditional index, so that wins for an arm or losses for the other arm result in an increment as indexed by the arm. Since actions are selected by using `argmax`, the arm with the greatest accumulated evidence is selected at any time.

Other discovered strategies worse than the accumulator corresponded to partial accumulators that only incorporated part of the reward feedback in either an arm-specific or reward-specific way (like in Fig. 4.1d). Strategies better than the accumulator were typically minor variants of it, like giving greater

weight to outcomes for the arm that was not initially selected.

For validation, we examined the strategies discovered when value was estimated and stochastic policies were permitted. We found similar policies as above, and use of stochastic policies that were often effectively deterministic due to large values of inverse temperatures.

One question is whether the resource-rational trade-offs implied by this model rationalize idiosyncratic behaviors in biological learning. One notable random policy is the stochastic partial accumulator in Fig. 4.1e, which shows a modest increase in value over WSLS. Interestingly, this strategy ignores losses. This can result in poor performance with deterministic behavior, since choice is fixed to the first rewarding option (Fig. 4.1d, marked in Fig. 4.1a). However, partial accumulation can perform effectively with stochastic choice. This example is broadly consistent with findings of asymmetric learning from positive and negative prediction errors, such as exclusively learning from reward by rodents in a reversal learning task (Parker et al., 2016). It also echoes findings of optimistic learning biases in humans (Palminteri, 2023) that theoretical accounts (in a related task with counterfactual feedback) have shown to be normative in some settings (Lefebvre et al., 2022).

To examine this more closely, we searched exhaustively over a small space of strategies to see whether exclusive accumulation of either rewarded or unrewarded trials was more effective. The space of strategies was a generalization of the partial accumulators in Fig. 4.1, testing starting conditions, inverse temperature, and whether policies were deterministic or not. When rewards were accumulated, the best strategy we found was Fig. 4.1e. When unrewarded trials were accumulated (replacing `reward` with $+(-1, \text{reward})$), the best performing strategy was deterministic and implemented WSLS. Thus, within this limited class of strategies, a bias towards positive information is rational.

4.4 ADAPTIVE RANDOM EXPLORATION

A key feature of optimal behavior in bandit tasks is adaptive use of exploration—exploring is important early on, but over time behavior favors the best choice given previous observations. Wilson et al.

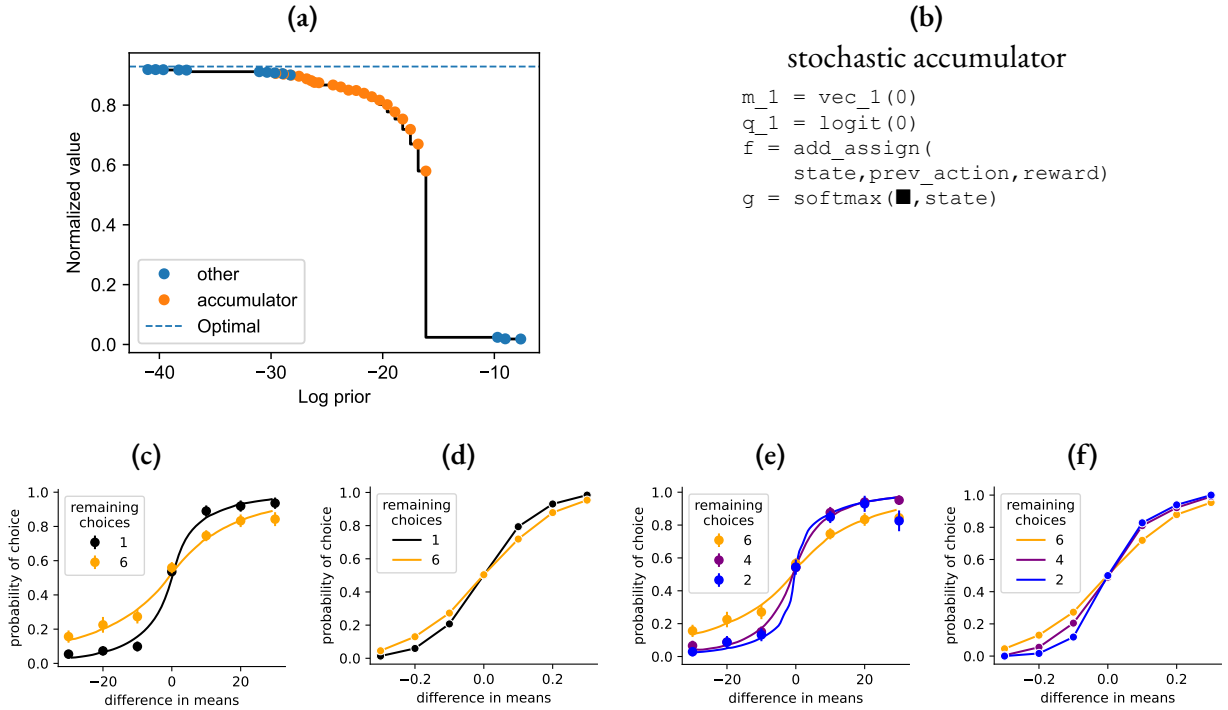


Figure 4.2: Adaptive random exploration, using a stochastic accumulator. a) The Pareto frontier for solutions in the long horizon condition. Many solutions simply accumulate rewards, but increase the determinism of policies (less probable under prior) to achieve greater value. b) The discovered accumulator. The “■” is a placeholder for the inverse temperature. c) Replotted from Fig. 2b in Wilson et al. (2014). Across conditions, there is a horizon-specific adjustment of decision noise. d) The optimal horizon-specific inverse temperature for the stochastic accumulator leads to more random behavior for the long horizon condition. Horizon-specific inverse temperature was selected to maximize Eq. 4.3 with $\beta = 300$. Rewards were scaled by $\frac{1}{100}$. e) Replotted from Fig. 3a in Wilson et al. (2014). In the long horizon condition, there is horizon-specific adjustment of decision noise. f) Memory magnitude in the accumulator grows over time, resulting in less randomness for later trials in the long horizon condition. Same inverse temperature as d).

(2014) showed that humans explore adaptively in a bandit task, with increased preference for unfamiliar and less valuable arms for a longer horizon and decreased preference for both for a shorter horizon. While horizon-dependent exploration of unfamiliar arms is predicted by normative accounts, adaptive random exploration is hard to explain since optimal behavior is deterministic.

The task is a two-armed bandit with normal rewards that starts with four forced-choice trials, to control the information participants have about each arm, then proceeds to either one or six free-choice trials. Forced-choice trials ignored action probabilities, and an additional primitive was added so that

strategies could test whether the previous trial was a forced-choice trial. Strategies were found for each horizon condition independently and, to facilitate comparing conditions, value was the average received reward on free-choice trials.

Applying our framework to search for stochastic strategies for this task, we primarily identified an accumulator along the Pareto front (long horizon shown in Fig. 4.2a, short horizon was similar), with varied inverse temperatures for the `softmax` (Fig. 4.2b, increasing with the value weight because the prior disfavors larger constants). This strategy performs best with a high inverse temperature, resulting in near-deterministic behavior. In this section, we will use this example program to account for the adaptive random exploration found by Wilson et al., focusing on the equal-information conditions where it is most directly exposed.

One pattern of adaptive random exploration was between different time horizons: on the first free-choice trial, participants were more random in the long horizon condition (Fig. 4.2c). Focusing on parametrically varying the strategy in Fig. 4.2b, we found horizon-specific inverse temperatures for the `softmax` that maximized the posterior (Eq. 4.3). The inverse temperature was larger for the short horizon, resulting in more deterministic behavior compared to the long horizon (Fig. 4.2d), consistent with the behavioral findings.

This horizon-dependent randomness results from an interplay between the accumulator’s memory dynamics and the representation cost of the policy. Accumulating rewards instead of weighting them means that the magnitude of entries in the agent’s state will increase over time, leading to greater determinism. So, for the long horizon condition, the inverse temperature primarily serves to make early choices more deterministic. Since integers are geometrically-distributed, larger values of the constant softmax temperature are less probable. This penalizes the determinism of a policy, resembling information-theoretic approaches to penalizing policy complexity (Piray & Daw, 2021; Lai & Gershman, 2021). Combined, this means that a strategy specific to the long horizon condition can afford to be more random early on, avoiding the penalty associated with being deterministic.

The second pattern of adaptive random exploration identified by Wilson et al. (2014) is within the

long horizon condition: participant choice was more random when there were more remaining trials (Fig. 4.2e). While these results incorporated free-choice trials, they attempted to avoid a confound between information and reward by focusing on cases where each arm had been selected an equal number of times. Using the best inverse temperature for the long horizon, we saw similar results (Fig. 4.2f). As described above, the accumulator’s entries grow over time, resulting in a natural shift toward more deterministic behavior.

4.5 DISCRETE DECISION STATES

We next examine strategies that emerge for learning in a bandit task with non-stationary rewards, which is difficult to solve with the accumulation models presented in previous sections. Ebitz et al. (2018) found that both choice behavior and neural activity from monkeys in this setting were captured by a slightly generalized WSLs model that involved switching between distinct states: exploring by random sampling, and exploiting by selecting a preferred action (Fig. 4.3a). Quickly switching between several distinctive regimes of behavior is difficult to model using error-based learning.

We study the task reported by Ebitz et al., which is a 3-armed bandit with Bernoulli rewards. The probability of reward for each arm is in $(0.1, 0.2, \dots, 0.9)$. After every trial, the probability has a 10% chance of drifting up or down by 0.1, while staying in bounds. We set the horizon to 500 trials.

Our framework identifies strategies with similar motifs as the one proposed by Ebitz et al., so we qualitatively examine two in this section. The simpler program in Fig. 4.3b implements WSLs where losses lead to random sampling among options—a loss means `reward=0`, so `softmax` returns a uniform distribution. We also show the program’s dynamics as a state machine in Fig. 4.3c, where each state corresponds to a unique distribution over actions, and the arrows show how wins and losses change the policy. In the state machine, the explore state results in random choice. Wins lead to an action-specific exploit state, and losses lead back to the explore state. This strategy precisely mirrors the schematic proposed by Ebitz et al. However, while their detailed results are consistent with state switch-

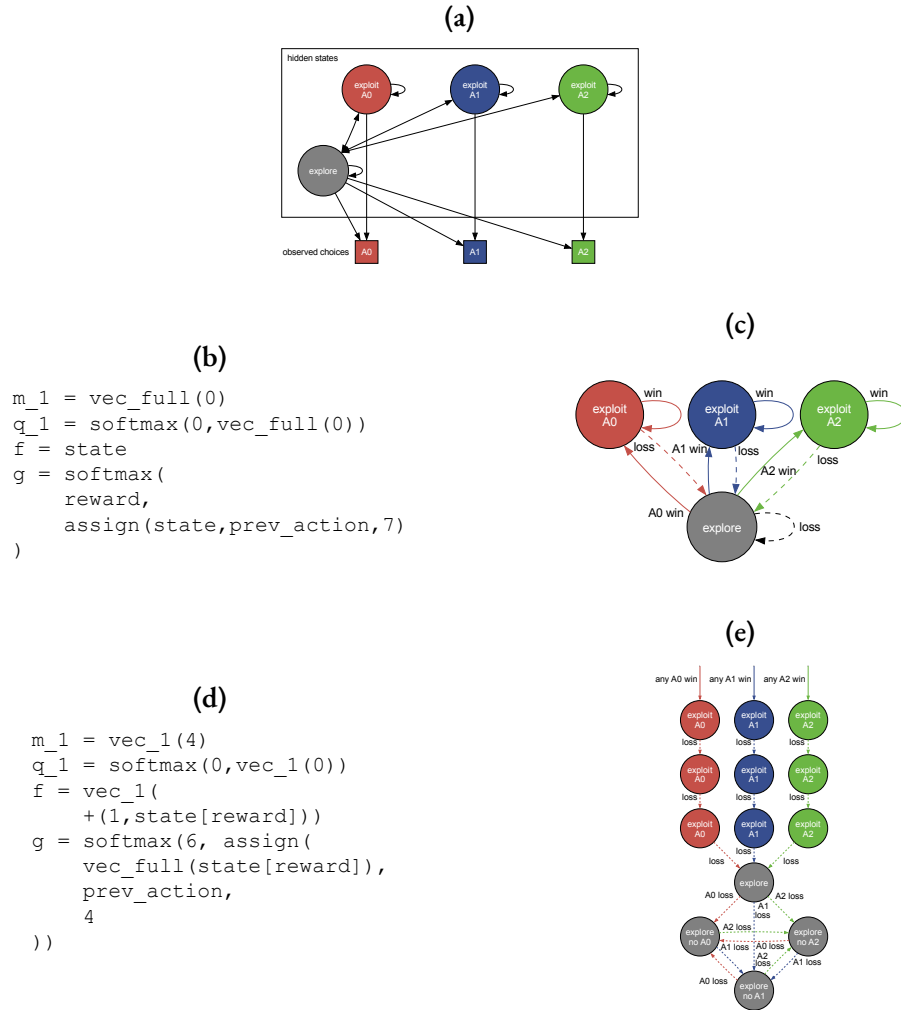


Figure 4.3: Discovering strategies with discrete decision states for a bandit task with non-stationary reward. a) The state-based choice model in Ebitz et al. (2018), featuring distinct states where behavior is either more exploratory or exploitative. Adapted from Fig. 1c in Ebitz et al. (2018). b) A WSLS strategy that randomly samples from actions after a loss. c) A state machine for WSLS strategy. States correspond to distinct action distributions. Edge color indicates action, dashed edges correspond to losses, and edges with action probability of less than 1% were excluded. The initial conditions of the strategy were modified to simplify the state machine. d) A more complex strategy that exploits after a single win, but requires consecutive losses to switch. e) State machine for d). Wins always lead to a state at top (as indicated), so wins are excluded elsewhere. States with 4+ consecutive losses (at bottom) were collapsed because they have similar action probabilities and identical transitions.

ing, they also show that both neural and behavioral measures somewhat violate the Markov conditional independence properties implied by their WSLS model, suggesting the actual state space is more complicated. We explore one such strategy next.

The strategy in Fig. 4.3d (state machine in Fig. 4.3e) elaborates both the exploration and exploitation phases. In particular, when exploiting (top) it switches only after three successive losses; for exploration (bottom) it tries random arms until a win, but avoids the most recent choice. The code implements this by counting consecutive losses in the 0^{th} entry of memory, resetting the counter to 1 following a win, and using the counter to inform choice probabilities. Importantly, this strategy retains the discrete, state-like nature of the proposal in Ebitz et al. Similar to the bandit algorithms discussed above, this strategy also responds more readily to positive feedback, which is also consistent with theoretical findings (in a related task) that choice-confirmation bias is adaptive when reward is non-stationary (Lefebvre et al., 2022).

4.6 DISCUSSION

We have proposed a framework for strategy induction, using inferential methods to identify strategies that are both simple and effective. By searching for program-structured strategies, our approach is able to find strategy representations that are more interpretable than results from other approaches. We use this framework to examine strategies for RL tasks that trade off between simplicity and effectiveness. We were able to discover strategies consistent with previously observed behavior, like asymmetric sensitivity to prediction errors, adaptive random exploration, and use of discrete decision states.

We were able to justify adaptive, horizon-dependent random exploration in our framework, as previously observed (Wilson et al., 2014). Within a fixed class of strategies, our framework can cast this adaptive behavior as the result of a resource-rational trade-off. Other approaches could explain this by penalizing the divergence of policies from a uniform distribution (Lai & Gershman, 2021; Piray & Daw, 2021; Levine, 2018). However, our approach goes beyond simple information-theoretic penalties on action distributions since it can identify structured, task-specific strategies and penalize them based on their description length.

One limitation of our current approach is that the primary cost of a program is its representational

cost. This neglects any execution-related costs the program might incur, like memory use. Penalizing trial-specific computation costs could encourage strategies that conditionally perform fewer computations in some trials, which, for example, could result in habits which can be adaptive under a speed-accuracy trade-off (Dezfouli & Balleine, 2012). Future research could also adjust the primitive operations available to reflect cognitive resource limitations. For example, many kinds of memory are known to be subject to capacity limits or noise, like working memory (Collins & Frank, 2012) and action values (Findling et al., 2019).

Our approach is a computational-level account (Marr, 1982) of strategy induction, leaving open questions about the algorithms underlying strategy discovery. Future work could examine the process-level bias induced by varying the number of samples taken in MCMC or for value estimation. Another question is how library learning, which adds common subprograms as primitives (Ellis et al., 2021), could accelerate inference. Library learning offers a simple way to perform continual learning, which, by contrast, is an open challenge for neural networks (e.g., requiring task-specific network dynamics to be identified and frozen to avoid catastrophic interference in research by Duncker et al., 2020).

While we have focused on resource-rational strategies for learning in a handful of bandit tasks, future research could apply this framework to identify interpretable strategies for sequential decision-making in settings like planning, problem solving, and other learning tasks. Future work could also apply this framework to discover strategies directly from behavioral data, as others have done with RNNs (Miller et al., 2023; Ji-An et al., 2023). We hope that the continued development of strategy discovery methods can accelerate our understanding of cognition by identifying novel features of behavior (Collins & Cockburn, 2020) and avoid misinterpretation of behavioral signatures (Collins & Frank, 2012; Akam et al., 2015).

Chapter 5

Conclusion

Despite the difficulty—and often intractability—of optimal decision-making, organisms behave in ways that are often surprisingly adaptive. This difficulty compounds with the severe resource limitations organisms have, such as noisy and/or capacity-limited memory and slow sequential neural activity. In this dissertation, I have presented research that explores how organisms use hierarchical representations and strategies to simplify decision-making. By using a resource-rational framework, where performance is optimized while resource use is minimized, we were able to formalize how organisms choose among structured representations.

5.1 HIERARCHY

In Chapter 2, we developed a formalism for resource-rational subgoal choice that accounts for the resource costs of planning. In a comparison with other theories of hierarchy choice, we were able to identify relationships between our account and previously-proposed heuristics. Finally, in a large-scale experiment, we found that our account and related heuristics were most predictive of human subgoal choices.

Our formalism could be extended in several ways. We discussed a few extensions in the chapter: examining the algorithms people use to efficiently identify resource-rational subgoals, and incorporating structured or abstract subgoals (which others have begun to investigate; Binder et al., 2021, 2023). Like

previous accounts, we focused on simple hierarchies with a single level of abstraction. In our account, only the lowest level of abstraction (action-level planning) has associated planning costs, which could be naturally extended to consider the planning costs at higher levels of abstraction. Our account assumed no shared information between tasks (besides selected subgoals), but reusing plans to support repeated visits to a subgoal (as in Solway et al., 2014) could further reduce computational cost and make the account both more realistic and tractable.

Another question for future research is the neural basis of subgoal discovery. Neuroimaging studies typically focus on cases with a fixed hierarchy or small set of candidate hierarchies. Examining how subgoal use arises over the course of training could be performed through representational analyses that identify subgoal-specific and task-independent patterns of activity, though the large number of trials typically necessary for neuroimaging studies could make experimental design challenging. Additionally, more dynamic signatures of subgoal use would be difficult to identify with fMRI because of its coarse timescale, so studies might instead use MEG to examine how subgoals influence replayed sequences.

In Chapter 3, we used a process-tracing paradigm where participants created hierarchically-structured programs in order to solve a task. Moving beyond simple accounts, we developed a generative model of program creation with a bias towards reuse that is consistent with behavior.

In contrast to the other two chapters, this one focused on a descriptive analysis of behavior, so a natural question is: does the grammar induction model have a normative interpretation? The probability of sampling an item determines the expected number of samples necessary for the item to appear from random sampling. Solway et al. (2014) use this argument to frame their account as minimizing the cost of searching by random sampling, since hierarchies are selected to maximize the probability of optimal behavior (so they minimize the expected number of samples needed to sample it). A related argument is used by Binz & Schulz (2023) with a model based on importance sampling. Applying this argument to our model, programs that are more probable under the prior have lower search cost under random sampling. While the low probability of programs makes this seem like an implausible algorithmic theory, this is a starting point that builds on the present Bayesian formulation. Comparing to the model

in Chapter 2, the grammar induction model defines a distribution over hierarchies of arbitrary depth, and by the argument above, could be interpreted as minimizing a resource cost. A notable omission in our model is that task structure does not influence the prior in any way. The prior simply is concerned with patterns of reuse, leaving evaluation of the plan to the likelihood term. However, it is clear from experience that visual information is extremely informative and could be important to account for in future work.

Hierarchy has a number of benefits and costs: planning using subgoals permits deeper steps into a search tree, though considering too many options can be counterproductive, and there is also a benefit to either individually solving or memorizing subtasks. Future research could try to isolate the influence of each of these factors in how decompositions are selected, as in related behavioral research which has found that cached value estimates guide option choice (Cushman & Morris, 2015).

Aside from using hierarchy, a number of directions could examine other structural changes people make to simplify planning. For example, state abstraction is an approach to reducing the size of a state space by ignoring state details that are irrelevant to the current subtask, which can accelerate and simplify learning (Dietterich, 2000). Several accounts of hierarchical learning have developed models that learn or infer appropriate state abstractions (Frank & Badre, 2012; Eckstein & Collins, 2020). Outside of hierarchical tasks, related studies have examined how appropriate task features are selected (and irrelevant task features are discarded), focusing on how attention (Niv et al., 2015; Leong et al., 2017) and, more recently, the initial feature selectivity of neurons (Pan-Vazquez et al., 2024) facilitate learning. In research I contributed to, we found that people ignore irrelevant task details while planning (Ho et al., 2022), which we formalized as adaptively simplifying the task transition function. An important consideration in search is how to prioritize candidate states and actions, which recent research has begun to examine (Bear et al., 2020; Wu et al., 2018). In depth-limited search, an approximate value is used once the depth limit is reached, which past behavioral research has assumed is either a fixed evaluation function (van Opheusden et al., 2023) or bootstrapped based on estimates (Keramati et al., 2016). Understanding how people develop task-relevant approximate values, and their role in search, remain open

questions.

While this thesis avoids considering how curriculum impacts hierarchy choice, it is clear that past experiences influence how people solve problems (Luchins, 1942). Recent research has examined how past experiences influence how people choose hierarchies (Collins & Frank, 2013; Tomov et al., 2020) and how they learn program-based causal relations (Rule et al., 2018; Zhao et al., 2023). Lightbot could be an ideal setting to study the effects of curriculum on hierarchies, because of the process-tracing nature of the task.

5.2 STRATEGIES

In Chapter 4, we identified program-based strategies for bandit tasks by using Bayesian program induction methods. We were able to show how some idiosyncratic behaviors from previous studies arise as a result of trading off the complexity of a strategy (i.e. the program length) with its effectiveness (i.e. how much reward it accrues).

Our approach relies on sample-based value estimation for strategies with a fixed seed to reduce variance, resembling previous research (Wingate et al., 2011), though a more principled approach could treat the trajectories used for value estimation as part of the state of the Markov chain in MCMC (Tolpin et al., 2020; Hoffman et al., 2009). As noted in the chapter, library learning (Bowers et al., 2023; Ellis et al., 2021) could be an algorithmic approach to make inference more tractable and also provide an account for continual learning.

Classic research in solving POMDPs noted that optimal policies could be represented as finite-state controllers (FSC; i.e. finite-state machines that can take action, detailed in Kaelbling et al., 1998). This led to many research directions that focused on directly improving FSCs (Hansen, 1998) or using gradient-based methods to optimize stochastic FSCs (Meuleau et al., 1999; Duff, 2002). Previous research has used FSCs to study behavior in simple decision-theoretic contexts (Halpern et al., 2014; Liu & Halpern, 2020) and to exhaustively explore the space of FSCs for a reversal learning task (Ma & Her-

mundstad, 2022). While the strategies we explore have a finite number of possible states, an important difference is that our strategies are interpretable by design, making it possible to compactly define elaborate strategies. In this sense, our approach can be viewed as a way of finding FSCs with regular structure.

More recent research in computer science has used task-optimized neural networks to inform search for interpretable policies (Bastani et al., 2018; Verma et al., 2018), used neural networks to support search for task-optimized policies (Qiu & Zhu, 2022; Trivedi et al., 2021), and used evolutionary methods to find RL algorithms (Co-Reyes et al., 2021).

Other studies have taken a related approach by searching for structured policies, through genetic programming (Bartlett et al., 2023), imitation learning (Silver et al., 2020), or equation discovery (LaFollette et al., 2023). An alternative approach is to infer causal models (Das et al., 2023; Zhao et al., 2023), which can subsequently be used to solve tasks by planning (Sharma et al., 2022; Tsividis et al., 2021).

Typical research in meta-RL uses RNNs, which are a productive tool for searching the space of policies via gradient-based optimization. The approach we outlined, based on program inference, is an alternative that trades away a simple approach for optimization but is more interpretable. An open question is understanding how these two methods are related: under what conditions do they find similar strategies? While task-optimized networks discover strategies that diverge from behavior (Ji-An et al., 2023; Hattori et al., 2023), complexity-penalized networks (Binz & Schulz, 2022; Moskovitz et al., 2023) could provide a bias that facilitates comparison. RNNs have a much clearer relationship to neural algorithms than our program-based approach, so future work could explore these connections—for example, a neural approach to evaluating programs is sketched out by Piantadosi (2021).

5.3 FUTURE DIRECTIONS

This thesis was broadly concerned with computational-level accounts of behavior (Marr, 1982), but the algorithms that organisms use to tractably compute behavior remain an open question. I discussed ap-

proaches that tune sampling-based inference algorithms for efficiency (Daw & Courville, 2007; Lake & Piantadosi, 2020; Fränken et al., 2022). However, recent studies have developed the promising alternative of meta-learning inference algorithms, by training neural networks on tasks from structured distributions (McCoy & Griffiths, 2023; Lake & Baroni, 2023). In research I contributed to, we took a related approach that augmented standard meta-RL with auxiliary losses that align RNN hidden states with structured representations, resulting in more human-like learners (Kumar et al., 2022). Aside from developing computational models, another important direction is to focus on analyzing neural signatures, which could clarify the algorithms used to create and learn about heuristics/strategies, as well as provide more detail about how they are used to directly guide behavior.

While I have studied heuristics and strategies entirely separately in this thesis, I briefly consider how they are related. The hierarchical planning approach described in Chapter 2 first plans among subgoals, then, to reach each subgoal, develops and follows a subgoal-specific plan. The plan among subgoals resembles a rigidly-structured strategy consisting of a sequence of subgoal-specific planning operations, which could be expressed in a framework like the one introduced in Chapter 4 if planning were incorporated as a primitive. Planning itself could even be expressed as a strategy in the framework in Chapter 4, given sufficiently rich primitives. Unifying these approaches as strategies makes it clear that they optimize to produce more compact and/or efficient ways of describing behavior, by either making careful use of planning operations or finding simpler algorithms as an alternative to planning. The available primitives greatly influence which strategies are compact and/or efficient. Linking these primitives to the neural systems involved in decision-making—like error-driven learning by way of the basal ganglia, use of varied memory systems, replay of previously experienced or imagined trajectories, and the ability to meta-learn—could bring these strategies into closer alignment with the resource costs and limits that organisms face, yielding strategies that more closely resemble behavior.

I have presented work that explores how structured representations, like hierarchy and strategies, are used to support efficient decision-making. Optimal decisions are difficult to compute, which forces organisms to find alternative approaches that are both effective and efficient. Focusing on the use of

structured representations, we explored how hierarchy accelerates planning and simple strategies approximate optimal learning. We found that accounting for the resource costs organisms face helped explain the adaptive choices they make. I hope future research builds on these ideas to make sense of the orchestration of cognitive abilities, ultimately linked to their underlying neural basis.

Appendix A

Methods and Appendix for Chapter 2

A.1 METHODS

A.1.1 ETHICS STATEMENT

The following experimental procedures were approved by the institutional review board of Princeton University. In the experiment, participants were shown an electronic consent form providing a written description of the experiment and gave informed consent by clicking a button in lieu of a signature.

A.1.2 EXPERIMENT DESIGN

To probe for participant subgoal choice, we employed an experiment inspired by those previously published in Solway et al. (2014). In our experiment, participants first navigated on a web-like representation of a graph to learn the graph’s structure (“navigation trials”; Fig. 2.5), then answered a series of task-specific and task-independent questions about subgoal choices (“probe trials”). We then quantitatively analyzed their responses to these questions about subgoal choice. In the Design section, we motivate the choice of various experimental details. Then, in the Procedure section we detail the experimental procedure. The experiment pre-registration is available at <https://osf.io/hegf2>. Our pre-registered analysis was a comparison of how well RRTD-IDDFS and Solway et al. (2014) could predict participant probe choice using hierarchical multinomial regression, a subset of the comparisons in Fig. 2.8

and Table 2.2.

DESIGN

The navigation trials were intended to provide participants with an opportunity to learn the structure of the graph. Drawing from results in pilot experiments, we ensured participants could only see the graph edges connected to their current state (Fig. 2.5a). Periodically, the graph with all edges was shown to participants (Fig. 2.5b). Importantly, this was done without signaling any information about future tasks. From pilot experiments, these visual choices (minimizing displayed edges during tasks, but showing all edges periodically between tasks) ensured that participants quickly learned the graph structure instead of relying on the visual representation of the graph.

From pilot experiments, we observed that states with high visit rates coincided with the predictions of RRTD-IDDFS and Betweenness Centrality. To address this confound, the experiment had two types of navigation trials: long and filler. Long trials were optimally solved with more than one action (i.e. the start and goal state were not directly connected) and were intended to give participants exposure to the graph structure. Filler trials were optimally solved with one action (i.e. the start and goal state were directly connected) and were adaptively chosen to ensure a balanced visit rate that avoided overlapping predictions with our model. This adaptive procedure selected from all possible filler tasks by 1) excluding tasks where the start or goal state was most-visited, and then by 2) sampling uniformly from the remaining tasks with the greatest sum of visits to the start and goal states. When all tasks had a most-visited state as a start or goal, the first step was skipped; this circumstance was uncommon and dependent on both participant behavior and the structure of the graph. In effect, this procedure increased the number of most-visited states by increasing visits to states that were nearly (but not) most-visited. In simulations and pilot experiments, this procedure was sufficient to dissociate the visit rate and our model predictions.

Our probes for subgoal choice included two inspired by prior studies (Solway et al., 2014)—the implicit probe and teleportation question—and included a novel variant that explicitly asked about sub-

goal use—the explicit probe. We included these three probes to ensure a reliable and comprehensive measure of subgoal choice. The implicit probe was shown to participants before the explicit probe to avoid biasing participant responses.

Typical visual layouts of graphs often have a relationship to pairwise state distances, which means that a variety of visual heuristics are effective strategies when problem-solving. To avoid these confounding heuristics, we visually represented the graph states in a pseudo-random circular layout (Fig. 2.5). The same circular layout was shown for all trial types, including the periodic display of all graph edges.

PROCEDURE

Each participant was assigned a single graph for the entire experiment, with a fixed circular layout and fixed mapping from nodes to icons. We first introduced them to the experimental interface with an interactive tutorial, showing the visual cues used to mark the current node and goal node and how to navigate along graph edges using the numeric keys of the keyboard. We then introduced the concept of a “subgoal” through the example of a cross-country road trip with a “subgoal” located at the midpoint of the road trip. Participants were asked to describe what they thought “subgoal” meant.

Navigation trials followed this introductory material. Participants completed a few short practice trials, then completed 60 trials alternating between long and filler trial types: 30 long trials were drawn uniformly from those optimally solved with more than one action, and 30 filler trials were adaptively selected from those optimally solved with one action (described in detail above). In navigation trials, participants started from a node and had to navigate to a goal node. They were also prompted to consider the use of subgoals with the message “It might be helpful to set subgoals.” At any point during a navigation trial, participants could only see the edges connected to their current node (Fig. 2.5a). Every four trials (thus, 15 times total) participants were shown all the edges of the graph (Fig. 2.5b). Since a photograph of the graph shown this way could simplify navigation trials, the icons at each node were only shown when the participant hovered over them.

Following navigation trials, we probed for subgoal choice. For all probes, the graph was shown in the

same circular layout, but the edges were hidden. Between every two trials, the graph was shown with all edges as mentioned above. In the context of 10 different tasks, we queried for subgoal choice using the implicit probe: “Plan how to get from A to B. Choose a location you would visit along the way.” For this probe, we excluded both the start and goal nodes from the available options. Then, in the context of the same 10 tasks after shuffling, we queried for subgoal choice using the explicit probe: “When navigating from A to B, what location would you set as a subgoal? (If none, click on the goal).” For this probe, we only excluded the start node from the available options. Before each of the task-specific probes, participants also completed a few practice trials. Finally, we asked a single instance of the teleportation question: “If you did the task again, which location would you choose to use for instant teleportation?” For this probe, all nodes were available options.

Finally, participants responded to a multiple-choice survey question: “Did you draw or take a picture of the map? If you did, how often did you look at it?”

STIMULI

The graphs we studied were sampled from among the 11,117 simple, connected, undirected, eight-node graphs with sufficient probe trials for the study design. For a given graph, probe trials were sampled uniformly from tasks that require at least 3 actions to optimally solve, a threshold selected based on model predictions that these tasks often require the use of hierarchy. After ensuring each graph had 10 distinct tasks that require 3+ actions to optimally solve, this limited the number of possible graphs to 1,676. The 30 graphs we studied were sampled uniformly at random from these 1,676 graphs (Fig. A.1). The order of graph nodes in the circular layout, the icon assigned to each node, and the sequence of navigation and probe trials were all sampled pseudo-randomly. We counter-balanced the assignment of participants to graph, circular layout, and trial orderings.

All icons designed by OpenMoji, the open-source emoji and icon project. License: CC BY-SA 4.0.

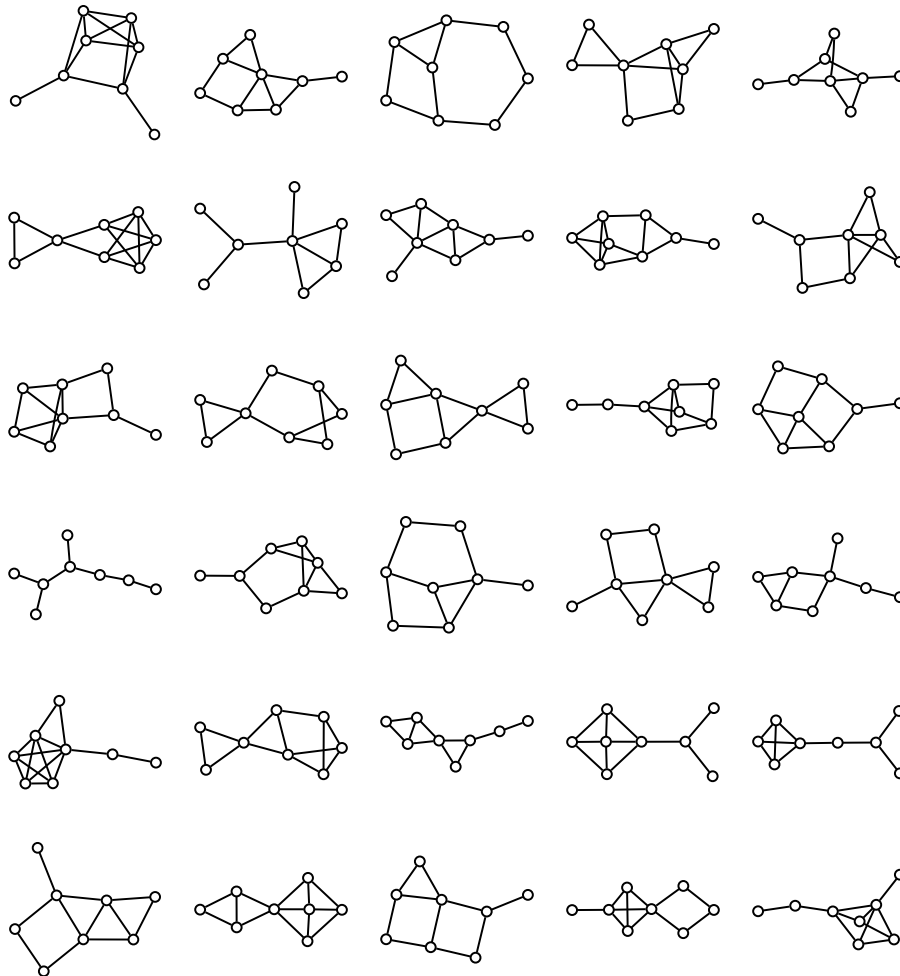


Figure A.1: The 30 undirected, eight-node graphs that were used in the experiment.

A.1.3 ANALYSES

MODEL PREDICTIONS

We define a model’s predictions over subgoals as proportional to a utility or log probability. We do so because model predictions are primarily used to predict probe choice using multinomial regression. This leads to a natural interpretation of the coefficients from multinomial regression. For a utility, multinomial regression is equivalent to a softmax choice model, so the coefficient can be equivalently interpreted as an inverse temperature. For a log probability, the coefficient w can predict a range of

strategies, including random choice for $w = 0$, probability matching for $w = 1$, and maximizing based on probability as $w \rightarrow \infty$.

We define the task distribution, when applicable to a model, as a uniform distribution over all tasks with distinct start and goal states. The reported analyses have qualitatively similar results when the task distribution matches the experiment’s long trials, which only includes tasks (s_0, g) where the start s_0 and goal g are not neighbors, so $(s_0, g) \notin T$.

HIERARCHICAL MULTINOMIAL REGRESSION OF CHOICE

We model participant choice among subgoals using hierarchical multinomial regression with the `mlogit` package in the R programming language. Regression models are fit with 100 draws from the default Halton sequence (parameters `halton=NA`, `R=100`).

For each model, we predict participant choices for each type of probe trial using multinomial regression with model predictions as regressors. Regressors were standardized to be mean-centered with a standard deviation of 1. Since each participant has multiple task-specific probe choices for the explicit and implicit probes, we include random effects for regressors when modeling those probe types. While not explicitly noted above, since the teleportation question was only asked once per participant, prediction of subgoal choice for it was fit without random effects (i.e. non-hierarchical multinomial regression).

For task-specific probes, the set of possible choices available to the model are configured to match those available to participants, as described in the [Procedure](#) section. Additionally, the explicit probe instructs participants to select the goal if they did not use subgoals. For RRTD-based models, we model this with the predicted value for the use of no subgoals. This is equivalent to the sum of the reward and negated planning cost of navigating directly to the goal.

TWO-STAGE MODEL OF CHOICE AMONG OPTIMAL PATHS

Free parameters β_1 and β_2 were constrained to be greater than or equal to zero. Optimization started from initial parameters $\beta_1 = 1, \beta_2 = 1$. The random choice model selects subgoals uniformly at random which corresponds to a special case of the two-stage choice model with fixed parameter $\beta_1 = 0$.

A.1.4 RESOURCE-RATIONAL TASK DECOMPOSITION

The model predictions for a state s are the value of a task decomposition (Eq. 2.3) where the state is a single subgoal, $V(\mathcal{Z}) = V(\{s\})$.

RANDOM WALK

The search algorithm returns a plan $\pi = \langle s_0, s_1, \dots, z \rangle$ and run-time $t = |\pi|$ with probability $P_{\text{RW}}(\pi, t \mid s, z) = \prod_{i>0} \frac{1}{\mathcal{N}(s_i)}$. Since we defined the reward for a plan as $R(\pi) = -|\pi|$, the expected reward over all plans is

$$\begin{aligned} R_{\text{RW}}(s, z) &= \sum_{\pi, t} P_{\text{RW}}(\pi, t \mid s, z) [R(\pi) - t] \\ &= -2 \sum_{\pi, t} P_{\text{RW}}(\pi, t \mid s, z) |\pi|. \end{aligned}$$

Since a constant multiplier does not affect model predictions, we drop the constant and let $R_{\text{RW}}(s, z) = -\sum_{\pi, t} P_{\text{RW}}(\pi, t \mid s, z) |\pi|$.

The negative expected reward $-R_{\text{RW}}(s, z)$ is the expected number of steps until the first visit to z when starting at s , also called the hitting time $H(s, z)$. $H(s, z)$ can be efficiently computed by a recursive equation

$$H(s, z) = 1 + \sum_{s' \in \mathcal{N}(s)} \frac{1}{|\mathcal{N}(s)|} H(s', z)$$

when $s \neq z$, and with $H(s, s) = 0$ otherwise. We thus define $R_{\text{RW}}(s, z) = -H(s, z)$.

Since the use of subgoals will either maintain or increase the number of steps required to reach a goal for a random walk, we make note of implementation differences to accommodate this for RRTD-RW. Formally stated, $H(s, s') + H(s', s'') \geq H(s, s'')$, with $H(s, s') + H(s', s'') = H(s, s'')$ only when all state sequences from s to s'' must contain s' . So, states s' with $H(s, s') + H(s', s'') > H(s, s'')$ will only increase the expected number of steps and would not be in the policy over subgoals defined by Eq. 2.2. To avoid this issue, we require the subgoal policy for RRTD-RW to contain at least one subgoal. By the same argument, a second subgoal can not decrease the expected number of steps, so we can simplify Eq. 2.2 for RRTD-RW to

$$V_{\mathcal{Z}}^g(s) = \max_{z \in \mathcal{Z}} \{-[H(s, z) + H(z, g)]\} \quad (\text{A.1})$$

or when the subgoals are a singleton set $\mathcal{Z} = \{z\}$ simply $V_{\mathcal{Z}}^g(s) = -[H(s, z) + H(z, g)]$.

DEPTH-FIRST SEARCH

The algorithm is recursively defined to take a current state s and plan-so-far π . At each call of the algorithm, it iterates over neighbors of the current state $s' \in \mathcal{N}(s)$ —if the state s' is not in the current plan π then there is a recursive call to the algorithm with state s' and an updated plan π' that ends with s' . When there are no unvisited neighbors s' to consider, the algorithm backtracks to a previous state and plans until it finds one with unvisited neighbors. When the algorithm reaches the subgoal z , it terminates, returning the plan and a run-time based on the number of calls to the algorithm. To avoid bias due to neighbor order, in each call of the algorithm neighbors are randomly shuffled.

BREADTH-FIRST SEARCH

The algorithm has a queue of states to visit and tracks all states that have been visited. At each iteration, it visits the next state s from the queue and adds all not-yet-visited neighbors $s' \in \mathcal{N}(s)$ to the queue. When it visits the subgoal z , it returns the path to z and a run-time based on the number of iterations that were required. As in DFS, we shuffle the neighbors of the current state at each iteration to avoid bias due to neighbor order.

ITERATIVE DEEPENING DEPTH-FIRST SEARCH

A depth-limited DFS augments a standard DFS by terminating when the current “depth” (i.e. the length of the current plan) exceeds a limit, in addition to terminating when the goal is reached. IDDFS iterates by running depth-limited DFS with incrementally larger depths, starting from a depth limit of 1. The algorithm returns when a depth-limited DFS finds a plan to the goal, counting the run-time as the number of recursive DFS calls across all uses of depth-limited DFS. As in other algorithms, neighbors are shuffled to avoid bias due to order.

A.1.5 ALTERNATIVE MODELS

DEGREE CENTRALITY, BETWEENNESS CENTRALITY

Both centrality measures were computed in the Python programming language using the `networkx` library with all parameters left at their defaults except for `endpoints=True` for Betweenness Centrality. As computed by `networkx`, both centrality measures are a fraction—for a given state s , Degree Centrality is proportional to the fraction of states that s is connected to and Betweenness Centrality is the fraction of optimal paths that s is part of. Thus, for both measures, we define the model predictions as the logarithm of these fractions for the reasons noted above.

QCU_T

This section requires more extensive use of graph theory, so we first explicitly connect the task formalism used in the remainder of the text to graphs before describing QCU_T. An undirected graph consists of nodes $\mathcal{V}$ and edges $\{i, j\} \in \mathcal{E}$ between nodes i and j , and we let $n = |\mathcal{V}|$ and $m = |\mathcal{E}|$. The adjacency matrix of a graph $A_{ij} = 1$ if $\{i, j\} \in \mathcal{E}$ and 0 otherwise. The degree of a node i is $d_i = \sum_j A_{ij} = \sum_i A_{ij}$ and the degree matrix $D = \text{diag}(d)$. To connect the notation used in the rest of the paper, we contextually assume the following relationship between undirected graphs and task environments: For environments with reversible actions (formally, $(s, s') \in T \iff (s', s) \in T$), we let states correspond directly to graph nodes and transitions in the environment $(s, s') \in T$ correspond to graph edges $\{s, s'\} \in \mathcal{E}$.

QCU_T divides the states of a graph into two groups based on the Normalized Cut criterion, which admits an approximate solution based on a spectral decomposition of the graph (Shi & Malik, 2000; Menache et al., 2002; Şimşek et al., 2005). The approximate solution to the Normalized Cut criterion is based on the symmetric graph Laplacian $\mathcal{L}_{\text{sym}} = D^{-\frac{1}{2}} (D - A) D^{-\frac{1}{2}} = I - D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$.

Following prior work (Shi & Malik, 2000; Şimşek et al., 2005), we divide graph nodes into two groups based on the eigenvector v with the second-smallest eigenvalue of $\mathcal{L}_{\text{sym}}$. This eigenvector is also the best, non-trivial one-dimensional embedding of the graph states that minimizes the distance between connected states (See Eq. 10 in Shi & Malik, 2000). Our implementation partitions states based on whether they are above or below a threshold in this one-dimensional embedding v —a typical threshold is zero or the median. Since states s with corresponding eigenvector entry v_s closest to the eigenvector mean are considered most central, we define the model prediction for a state s as $-v_s^2$.

SOLWAY ET AL. (2014)

Solway et al. (2014) propose that people choose hierarchies that most efficiently encode problem-solving behavior. The efficiency of an encoding is quantified through the information-theoretic concept of

minimum description length; when applied to encode problem-solving behavior through hierarchical structure, this involves choosing a task decomposition so that solutions have short description length and are composed of subtasks whose solutions can be reused in many tasks. This account takes optimal paths as the behavior to encode and selects hierarchies based on graph partitions.

We now note our implementation details that depart from those of Solway et al. (2014). To predict behavioral choices, we use the optimal behavioral hierarchy to specify a binary regressor that takes a value of 1 for *boundary states* (states with at least one neighbor in a different region) and 0 otherwise—we discuss this choice in the main text. Since multiple state sequences can be optimal, random noise is added to graph edges for the purpose of tie-breaking—in our implementation, the description length of behavior is averaged across 10 samplings of these edge weights in order to reduce the effects of noise. In the original publication, optimization over partitions based on model evidence was performed using a genetic algorithm—in our implementation, we enumerate all graph partitions and select those with the highest model evidence. For a given graph, the original article considers all possible partitions—for our analyses over eight-node graphs, we found it necessary to exclude trivial partitions. So, when possible for a graph, we only considered partitions with at least two regions and required that each region contained at least one state that was not a boundary state. The description length of behavior (the “model evidence”) was computed using R code supplied by Dr. Alec Solway.

TOMOV ET AL. (2020)

Tomov et al. (2020) propose an account of task decomposition as inference over hierarchical structure. Their generative model of hierarchical structure assumes partitions are drawn from a Chinese Restaurant Process and additionally assumes that edges between states are more likely when the states are in the same region. Their model also incorporates terms related to the task distribution and reward function that we expect to have minimal impact on our results because we do not vary the task distribution and the reward function for our task is a constant.

To incorporate this model, we used the analysis in the “Simulation Two: Bottleneck States” section

in the publication (Tomov et al., 2020). The analysis models participants ($N = 40$) as sampling from a generative model over hierarchical graphs, then randomly sampling subgoals (three per participant) from the states connected to a “bridge” edge, which connects distinct regions in the hierarchical graph. Notably, pairs of regions are connected by a single “bridge” edge—this is subtly different from standard graph partitions, where different regions can be connected by any number of edges. All parameters were left as reported in the publication (Tomov et al., 2020), with the exception of choice stochasticity ϵ which we made entirely deterministic by setting $\epsilon = 1.0$. To implement the published analyses, we used the publicly available code at <https://github.com/tomov/chunking>. We define the model prediction as the logarithm of the number of times a subgoal was sampled by this procedure—to avoid issues where a subgoal isn’t sampled due to noise, we add 1 to the subgoal counts before taking the logarithm.

A.2 EXTENDED DATA

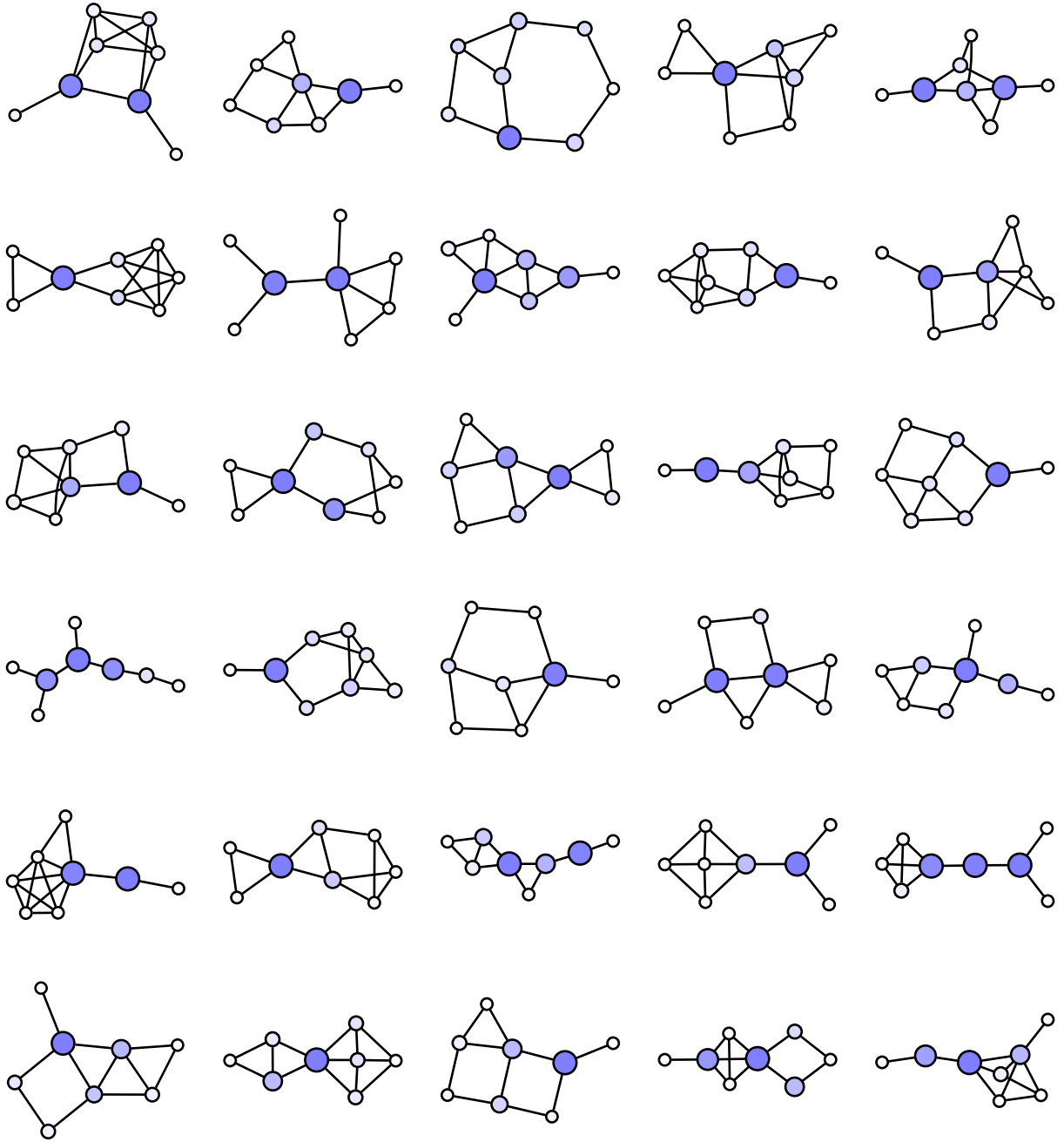


Figure A.2: Visualization of participant behavior by graph. State color and size is proportional to subgoal choice, summed across participants and probe types. Participants per graph ranged from 21–30 and each participant responded to a total of 21 probe questions.

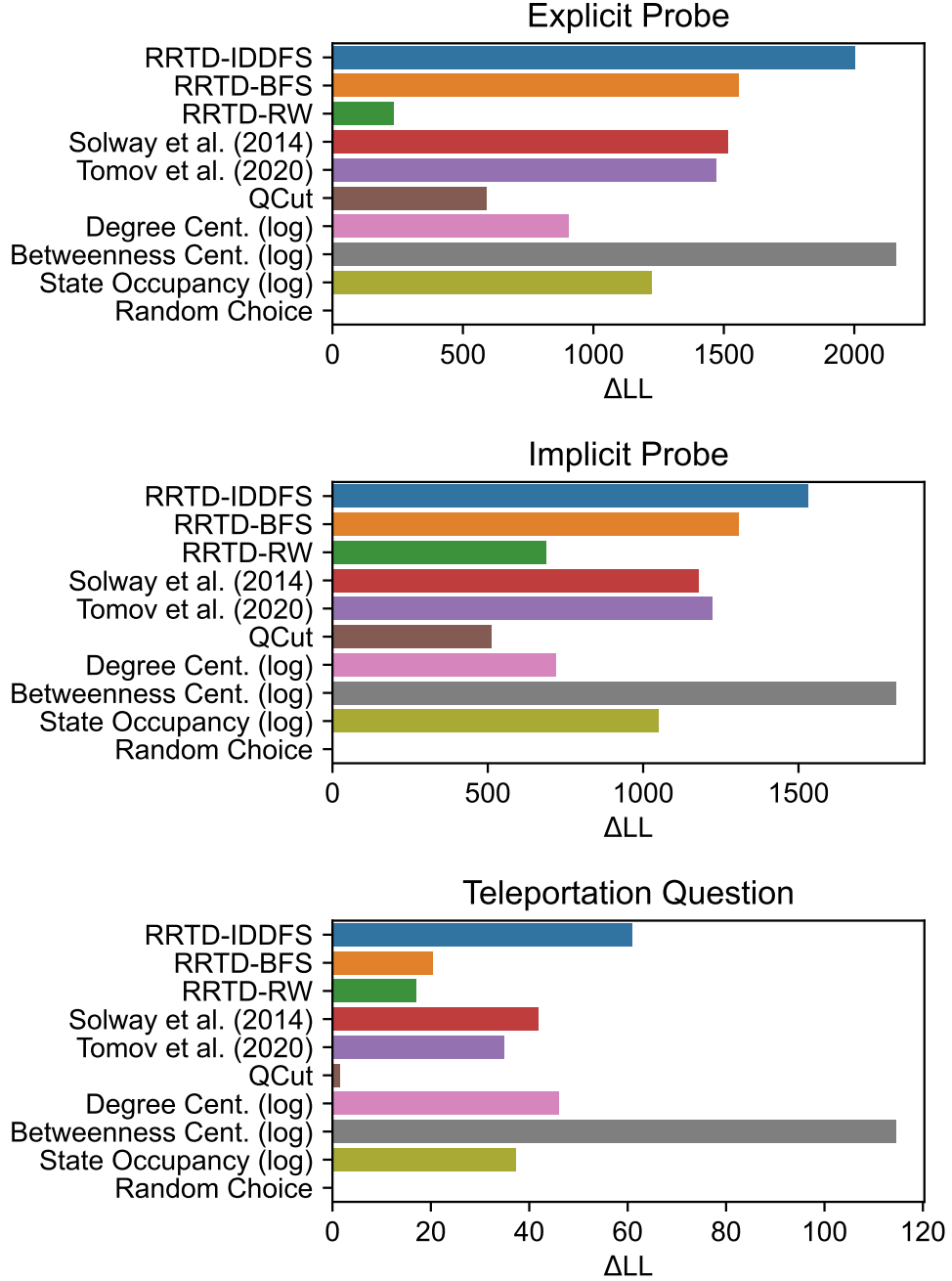


Figure A.3: Comparison of probe choice behavior in the subset of participants ($N = 603$) that reported not using an additional visual aid during the experiment. The analysis is otherwise identical to that in the main text, using mixed-effects multinomial regression to predict subgoal choice behavior for each subgoal probe. Log likelihood (LL) is relative to the minimum model LL for each probe. Larger values indicate better predictions of behavior.

Table A.1: Estimated coefficients with standard errors from hierarchical multinomial regression predicting subgoal choice, in the subset of participants ($N = 603$) that reported no use of visual aid in the experiment. Likelihood-ratio test statistics compare regression models to the null hypothesis of sampling subgoals uniformly at random.

Normative Algorithm	Explicit Probe	Implicit Probe	Teleportation Question
RRTD-IDDFS	$\beta = 1.84$ $SE = 0.04$ $\chi^2(2) = 4004.7$ $p < .001$	$\beta = 1.69$ $SE = 0.04$ $\chi^2(2) = 3058.4$ $p < .001$	$\beta = 0.76$ $SE = 0.07$ $\chi^2(1) = 121.6$ $p < .001$
RRTD-BFS	$\beta = 4.47$ $SE = 0.11$ $\chi^2(2) = 3114.7$ $p < .001$	$\beta = 5.25$ $SE = 0.14$ $\chi^2(2) = 2614.5$ $p < .001$	$\beta = 1.51$ $SE = 0.24$ $\chi^2(1) = 40.8$ $p < .001$
RRTD-RW	$\beta = 0.35$ $SE = 0.03$ $\chi^2(2) = 467.7$ $p < .001$	$\beta = 0.93$ $SE = 0.03$ $\chi^2(2) = 1375.0$ $p < .001$	$\beta = 0.31$ $SE = 0.06$ $\chi^2(1) = 33.9$ $p < .001$
Solway et al. (2014)	$\beta = 0.76$ $SE = 0.02$ $\chi^2(2) = 3031.7$ $p < .001$	$\beta = 0.70$ $SE = 0.02$ $\chi^2(2) = 2357.7$ $p < .001$	$\beta = 0.37$ $SE = 0.04$ $\chi^2(1) = 83.7$ $p < .001$
Tomov et al. (2020)	$\beta = 1.13$ $SE = 0.03$ $\chi^2(2) = 2937.3$ $p < .001$	$\beta = 1.01$ $SE = 0.03$ $\chi^2(2) = 2445.7$ $p < .001$	$\beta = 0.39$ $SE = 0.05$ $\chi^2(1) = 69.7$ $p < .001$
Heuristic	Explicit Probe	Implicit Probe	Teleportation Question
QCut (Şimşek et al., 2005)	$\beta = -0.15$ $SE = 0.01$ $\chi^2(2) = 1177.3$ $p < .001$	$\beta = -0.21$ $SE = 0.01$ $\chi^2(2) = 1025.3$ $p < .001$	$\beta = 0.07$ $SE = 0.04$ $\chi^2(1) = 2.8$ $p = .095$
Degree Cent. (log)	$\beta = 0.71$ $SE = 0.02$ $\chi^2(2) = 1811.1$ $p < .001$	$\beta = 0.64$ $SE = 0.02$ $\chi^2(2) = 1439.2$ $p < .001$	$\beta = 0.46$ $SE = 0.05$ $\chi^2(1) = 91.9$ $p < .001$
Betweenness Cent. (log)	$\beta = 0.89$ $SE = 0.02$ $\chi^2(2) = 4321.3$ $p < .001$	$\beta = 0.85$ $SE = 0.02$ $\chi^2(2) = 3629.0$ $p < .001$	$\beta = 0.58$ $SE = 0.04$ $\chi^2(1) = 229.1$ $p < .001$
State Occupancy (log)	$\beta = 1.02$ $SE = 0.03$ $\chi^2(2) = 2444.8$ $p < .001$	$\beta = 0.92$ $SE = 0.03$ $\chi^2(2) = 2096.2$ $p < .001$	$\beta = 0.38$ $SE = 0.05$ $\chi^2(1) = 74.4$ $p < .001$

A.3 THE INFLUENCE OF GRAPH COMPLEXITY AND STRUCTURE ON SUBGOAL CHOICE

We consider the influence of two graph measures on reports of subgoal choice, in order to understand how graph complexity and structure influence the ability of participants to identify subgoals or plan hierarchically. We compare graph measures to the subgoal choice count, or the number of times participants reported the use of a subgoal instead of the goal in the Explicit Probes. The subgoal choice count is also compared with navigation behavior in the main text.

First, we consider how the complexity of a graph might influence subgoal use. We note that participants may be influenced in either direction: A complex graph could be more difficult to learn, impeding identification of hierarchy. On the other hand, a complex graph may incentivize the use of hierarchy to support efficient planning. We use the number of edges per graph as a simple proxy for complexity. When a graph is represented as an *adjacency list*—a list of all the edges in the graph—it has a description length that is directly related to the number of edges. We find that the edge count and the average per-graph subgoal choice counts are uncorrelated ($r = 0.04$, $p = .821$). This analysis suggests subgoal choice identification and use is minimally impacted by graph complexity as measured by edge count.

The second measure we consider is related to structural properties of the graph—the spectral gap. We describe the spectral gap at length below, but briefly note that the spectral gap corresponds to a measure of graph modularity—graphs with a small spectral gap should be more modular and those with a large spectral gap should be more densely connected. Beyond this relationship to graph modularity, it has a relationship to the *mixing time*, or the rate at which a random walk converges to a stationary distribution (Lovász, 1993). We find a small, but non-significant, correlation between the spectral gap and the average per-graph subgoal choice count ($r = -0.22$, $p = .246$). A negative correlation means that participants selected subgoals more often for modular graphs (smaller spectral gap), and less often for densely connected graphs (larger spectral gap).

Table A.2: Estimated coefficients with standard errors from hierarchical multinomial regression predicting probe trial choice, where models include regressors for the icon shown. Likelihood ratio test is performed for the addition of fixed and random effects for subgoal predictions onto a null model containing only regressors for presented icons.

Algorithm	β	SE	Likelihood ratio test
RRTD-IDDFS	1.80	0.02	$\chi^2(2) = 3901.1, p < .001$
RRTD-BFS	4.38	0.07	$\chi^2(2) = 3335.0, p < .001$
RRTD-RW	0.35	0.02	$\chi^2(2) = 471.5, p < .001$
Solway et al. (2014)	0.68	0.01	$\chi^2(2) = 2786.9, p < .001$
Tomov et al. (2020)	0.96	0.02	$\chi^2(2) = 2545.5, p < .001$
QCut (Şimşek et al., 2005)	-0.26	0.01	$\chi^2(2) = 838.7, p < .001$
Degree Cent. (log)	0.68	0.01	$\chi^2(2) = 1585.7, p < .001$
Betweenness Cent. (log)	0.99	0.01	$\chi^2(2) = 4543.3, p < .001$

A.4 TESTING WHETHER ICONS INFLUENCE PARTICIPANT CHOICE ANALYSIS

We test whether icons influence participant choice by extending the hierarchical multinomial analyses of probe choice in the main text. We first evaluate whether subgoal predictions meaningfully explain the same choice data pooled across probe types when added to a null model containing fixed effects and per-participant random effects for the dummy-coded icon regressors. We use a likelihood ratio test for the addition of fixed and random effects for subgoal predictions to this null model—this test mirrors the likelihood ratio tests in the main text, besides the change in the null model. Across all models, we find that the addition of subgoal predictions is justified (Table A.2). Now, we evaluate whether our model comparison is impacted by the presence of the icon regressor in Fig. A.4. We find that the qualitative results observed in the main text are conserved. This indicates that icons have minimal influence on the results presented in the main text.

A.5 PREDICTING NAVIGATION RESPONSE TIMES

We first examined how participant response times vary during navigation, focusing on the difference between response times at self-reported subgoals and other states. For simplicity, we examine navigation

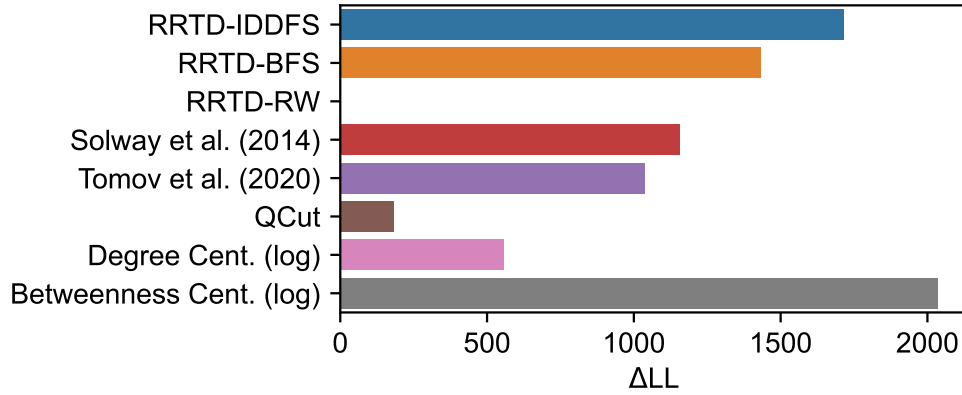


Figure A.4: Predicting probe trial choice using hierarchical multinomial regression using subgoal predictions and additional regressors for the icon displayed. Despite the inclusion of the icon regressor, these qualitative results mirror those in the main text. Log likelihood (LL) is relative to the minimum model LL. Larger values indicate better predictions of behavior.

trials with optimal behavior and an associated Explicit Probe trial where participants reported a subgoal that they also visited during navigation. First, we found that participants respond most slowly at the initial state ($M = 5.36s$, $SD = 7.58$) when compared to their responses at their self-reported subgoals ($M = 2.00s$, $SD = 1.38$; $t(789.8) = 11.9$, $p < .001$) or other states ($M = 1.70s$, $SD = 1.12$; $t(773.4) = 13.0$, $p < .001$). We also found that participants respond more slowly at their self-reported subgoal than at other states ($t(1423.9) = 4.6$, $p < .001$). We found similar results for the Implicit Probe trials. These findings could indicate that participants defer making detailed plans until they are necessary—that is, after reaching a subgoal, participants figure out how to reach the subsequent subgoal or goal.

Based on this result, we used model predictions for subgoals to predict response times during navigation. We analyzed every state choice during long navigation trials where participants took an optimal path. In order to focus on response times at subgoals compared to other states, we excluded the initial state to avoid any effect from initial planning. We found results that were largely qualitatively consistent with those observed elsewhere in the study (Fig. A.5 and Table A.3). The models that fit the observed data best included Betweenness Centrality and RRTD-IDDFS. Most models had positive coefficients, consistent with the idea that response times should be higher at states that are more likely to

Table A.3: Estimated coefficients with standard errors from hierarchical linear regression predicting response times for action choices during navigation trials.

Algorithm	β	SE	Likelihood ratio test
RRTD-IDDFS	0.07	0.03	$\chi^2(8) = 6074.8, p < .001$
RRTD-BFS	0.34	0.12	$\chi^2(8) = 5932.4, p < .001$
RRTD-RW	0.03	0.02	$\chi^2(8) = 6012.7, p < .001$
Solway et al. (2014)	0.03	0.01	$\chi^2(8) = 5964.5, p < .001$
Tomov et al. (2020)	0.03	0.01	$\chi^2(8) = 5966.5, p < .001$
QCut (Şimşek et al., 2005)	-0.05	0.01	$\chi^2(8) = 6182.9, p < .001$
Degree Cent. (log)	0.04	0.01	$\chi^2(8) = 6053.6, p < .001$
Betweenness Cent. (log)	0.05	0.01	$\chi^2(8) = 6028.5, p < .001$

be subgoals. However, two other models also fit the data well: Degree Centrality and QCut. The efficacy of Degree Centrality as a predictor suggests that people simply take longer to act at states that have greater degree (i.e. states that have more edges). The efficacy of QCut is more difficult to interpret in the context of our other results, particularly since it has a negative coefficient and is the best fit to the data.

Hierarchical linear regression analyses were fit using `lme4` and predicted the logarithm of participant response times in seconds. The regression models included an intercept and slope for subgoal predictions as effects at three levels: fixed effects, random effects for each graph, and random effects for each participant. In addition, they included a fixed effect of the trial number, in order to fit differences resulting from learning over the course of trials. Because the regression models have identical effect structure, we compare them by log likelihood (LL).

A.6 A FORMAL RELATIONSHIP BETWEEN QCUT AND RRTD-RW

We prove a formal relationship between QCut (Şimşek et al., 2005; Menache et al., 2002) and RRTD-RW for tasks based on undirected graphs. We show that a low-rank spectral approximation of RRTD-RW is equivalent to QCut for regular graphs. We use the notation introduced in the main text, recapitulated briefly here. An undirected graph has nodes $\mathcal{V}$ and edges $\{i, j\} \in \mathcal{E}$ between nodes i and j . We

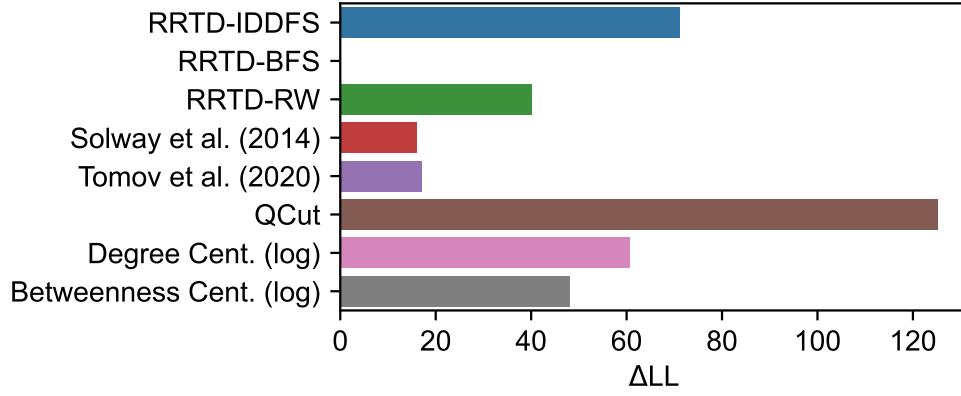


Figure A.5: Predicting navigation response times using hierarchical linear regression. Log likelihood (LL) is relative to the minimum model LL. Larger values indicate better predictions of behavior.

define $n = |\mathcal{V}|$ and $m = |\mathcal{E}|$. The adjacency matrix $A_{ij} = 1$ if $\{i, j\} \in \mathcal{E}$ and 0 otherwise. The degree of a node i is $d_i = \sum_j A_{ij} = \sum_i A_{ij}$ and the degree matrix $D = \text{diag}(d)$. For simplicity, we assume a uniform distribution over tasks $p(s_0, g) = \frac{1}{|\mathcal{S}|^2}$ and optimize for single subgoals z , so that $\mathcal{Z} = \{z\}$.

To start, we introduce notation from (Lovász, 1993) for the spectral definition of the hitting time. The matrix $N = D^{-\frac{1}{2}}AD^{-\frac{1}{2}}$ is symmetric, so it can be decomposed into eigenvalues λ_k and eigenvectors v_k in the definition $N = \sum_{k=1}^n \lambda_k v_k v_k^T$, with $1 = \lambda_1 > \lambda_2 \geq \dots \geq \lambda_n \geq -1$. We denote the i^{th} element of an eigenvector v_k as v_{ki} . The hitting time $H(s, z)$ can be computed using this spectral decomposition of N (Lovász, 1993)—the hitting time from a state s to z and back from z to s is

$$H(s, z) + H(z, s) = 2m \sum_{k=2} \frac{1}{1 - \lambda_k} \left(\frac{v_{kz}}{\sqrt{d_z}} - \frac{v_{ks}}{\sqrt{d_s}} \right)^2$$

Since we assume a uniform distribution over tasks, we can simplify the value of task decomposition $V(\mathcal{Z})$ using the special case of $V_{\mathcal{Z}}^g(s_0)$ for RRTD-RW in Eq. A.1 in the main text.

$$\begin{aligned}
V(\mathcal{Z}) &= \sum_{s_0, g} p(s_0, g) V_{\mathcal{Z}}^g(s_0) \\
&= -\frac{1}{n^2} \sum_{s_0, g} H(s_0, z) + H(z, g) \\
&= -\frac{1}{n} \left(\sum_{s_0} H(s_0, z) + \sum_g H(z, g) \right) \\
&= -\frac{1}{n} \sum_s H(s, z) + H(z, s)
\end{aligned}$$

Having simplified, we can straightforwardly incorporate the spectral form of the hitting time.

$$V(\mathcal{Z}) = -\frac{2m}{n} \sum_{k=2}^n \frac{1}{1 - \lambda_k} \sum_s \left(\frac{v_{kz}}{\sqrt{d_z}} - \frac{v_{ks}}{\sqrt{d_s}} \right)^2 \quad (\text{A.2})$$

We now apply this definition to develop a formal justification for QCut.

A.6.1 JUSTIFYING THE CHOICE OF EIGENVECTOR IN QCUT

To approximate $V(\mathcal{Z})$ using a single eigenvector, a reasonable choice is v_2 since it has largest weight $\frac{1}{1-\lambda_k}$. We can compare this choice of eigenvector to that of QCut, since the matrices N and $\mathcal{L}_{\text{sym}}$ share eigenvectors because $\mathcal{L}_{\text{sym}} = I - N$. The eigenvalues e_k of $\mathcal{L}_{\text{sym}}$ are $e_k = 1 - \lambda_k$ with $0 = e_1 < e_2 \leq \dots \leq e_n \leq 2$, since $\mathcal{L}_{\text{sym}} v_k = (I - N)v_k = (1 - \lambda_k)v_k$. The QCut algorithm makes use of the eigenvector of $\mathcal{L}_{\text{sym}}$ with the second smallest eigenvalue (Shi & Malik, 2000; Şimşek et al., 2005)—the second smallest eigenvalue is e_2 , which means the eigenvector used by QCut is v_2 . Our approximation of $V(\mathcal{Z})$ based on the weight $\frac{1}{1-\lambda_k}$ thus provides an informal rationale for the choice of eigenvector v_2 in QCut. To provide a rationale for the threshold used in QCut, we further simplify this equation for regular graphs.

A.6.2 JUSTIFYING QCut THRESHOLDING IN REGULAR GRAPHS

We can further connect RRTD-RW and QCut by limiting our consideration to regular graphs. We let d be the degree of nodes in a regular graph. For regular graphs, note that $\mathcal{L} = I - D^{-1}A = \mathcal{L}_{\text{sym}}$.

We first establish a helpful lemma: for each eigenvector v_k of $\mathcal{L}$, $\sum_s v_{ks} = 0$. Note that in general for matrices M where the columns j sum to zero (formally, $\sum_i M_{ij} = 0$), it is the case that $\sum_i (Mv)_i = \sum_{i,j} M_{ij}v_j = \sum_j v_j \sum_i M_{ij} = 0$. The columns j of $\mathcal{L}$ sum to zero, since $\sum_i \mathcal{L}_{ij} = \sum_i [I - D^{-1}A]_{ij} = 1 - \frac{1}{d} \sum_i A_{ij} = 0$. So, we know that $\sum_s (\mathcal{L}v)_s = 0$. In particular, $\sum_s (\mathcal{L}v_k)_s = \sum_s e_k v_{ks} = 0$, so $\sum_s v_{ks} = 0$.

We can now further simplify the above expression for $V(\mathcal{Z})$ since the v_k are unit vectors and $\sum_s v_{ks} = 0$ by the above lemma.

$$\begin{aligned} V(\mathcal{Z}) &= -\frac{2m}{dn} \sum_{k=2}^n \frac{1}{1 - \lambda_k} \sum_s (v_{kz} - v_{ks})^2 \\ V(\mathcal{Z}) &= -\frac{2m}{dn} \sum_{k=2}^n \frac{1}{1 - \lambda_k} \sum_s v_{kz}^2 + v_{ks}^2 - 2v_{kz}v_{ks} \\ V(\mathcal{Z}) &= -\frac{2m}{d} \sum_{k=2}^n \frac{1}{1 - \lambda_k} \left(v_{kz}^2 + \frac{1}{n} \right) \end{aligned}$$

Based on the rationale in the previous section, we can approximate this expression using a single eigenvector v_2 as $\tilde{V}(\mathcal{Z}) \propto -v_{2z}^2$. This expression is maximized by subgoals with corresponding eigenvector entry closest to 0. In QCut, states are partitioned based on a threshold of their entry in an eigenvector—typically, values of 0 or the median are used.

Altogether, these results establish a connection between RRTD-RW and QCut. We assume a uniform task distribution, rank-one approximation based on the weight $\frac{1}{1-\lambda_k}$, and that a graph is regular. With these assumptions, we can give a rationale for QCut that uses v_2 with a threshold of 0. Conne-

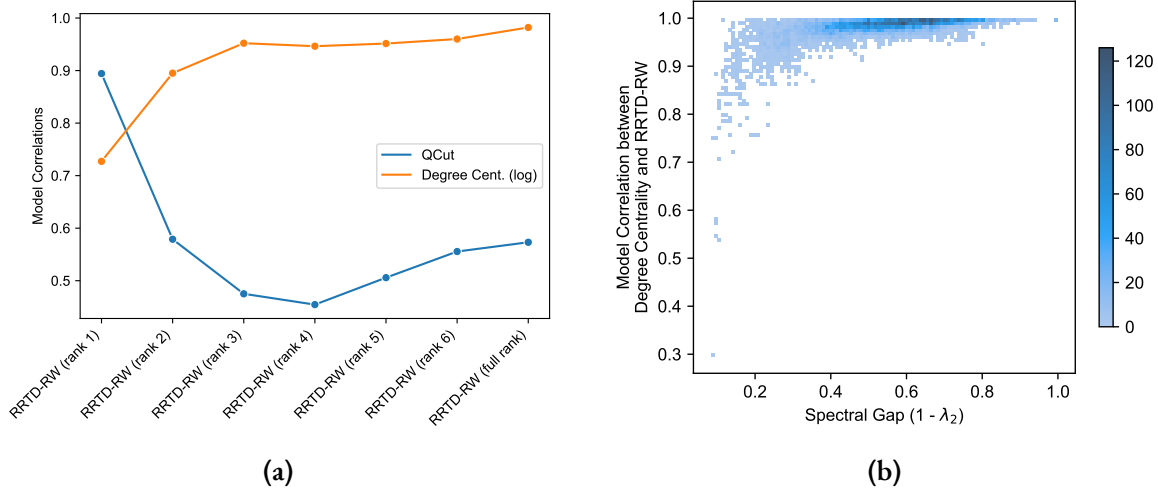


Figure A.6: a) Correlation of two models (QCut in blue and Degree Centrality in orange) plotted as a function of rank of a spectral approximation to RRTD-RW using the below formula for rank- k' spectral approximation. Correlations were computed as in Fig. 2.4 in the main text. b) Bivariate histogram of the 11,117 eight-node graphs based on the spectral gap ($1 - \lambda_2$) and correlation between Degree Centrality and RRTD-RW. A small spectral gap is associated with the existence of a bottleneck, explaining the relative decrease in Degree Centrality/RRTD-RW correlation for those values.

tions between graph spectra, spectral clustering, and random walks have been noted in passing in previous research (Stachenfeld et al., 2017; Von Luxburg, 2007; Shi & Malik, 2000), though the authors are not aware of a result directly relating the models considered in this section.

A.7 EXPLAINING THE RELATIONSHIP BETWEEN RRTD-RW AND DEGREE CENTRALITY

The plot in Fig. 2.4 in the main text validates our formal analysis by demonstrating an empirical result of modest correlation between QCut and RRTD-RW. However, Degree Centrality and RRTD-RW exhibit an even higher correlation which we examine in this section. A critical step in our above proof relating QCut and RRTD-RW requires a rank-one, spectral approximation to RRTD-RW. We examine how this approximation influences the relationships between models as the rank of the approximation is varied in Fig. A.6a, using this variant of Eq. A.2 to compute a rank- k' spectral approximation of

RRTD-RW:

$$V(\mathcal{Z} \mid k') = -\frac{2m}{n} \sum_{k=2}^{k'} \frac{1}{1 - \lambda_k} \sum_s \left(\frac{v_{kz}}{\sqrt{d_z}} - \frac{v_{ks}}{\sqrt{d_s}} \right)^2.$$

In order to ensure eigenvectors with identical eigenvalues contribute equally, so as to avoid any influence relating to eigenvector order, we additionally group eigenvectors by eigenvalue and average their contributions with appropriate weight based on k' . We make sure to group eigenvalues that are only distinct due to small numerical errors.

We find that the relationship between QCut and RRTD-RW is highest when RRTD-RW is approximated with a single eigenvector, which is consistent with the rank-one approximation necessary in the proof. The correlation between QCut and RRTD-RW is also only higher than the correlation between Degree Centrality and RRTD-RW for the rank-one approximation. For ranks greater than one, Degree Centrality has the highest correlation to RRTD-RW, with the maximum correlation to the full-rank RRTD-RW. Our formal analysis makes a stronger prediction for the case of regular graphs—in particular, QCut should be completely correlated with a rank-one approximation of RRTD-RW. When restricting attention to regular graphs, we find that QCut and rank-one RRTD-RW have a correlation of one, aligning with the formal analysis.

To conclude, we briefly and informally interpret the correlation between Degree Centrality and RRTD-RW by using the spectral gap $\lambda_1 - \lambda_2 = 1 - \lambda_2$ as a characterization of graph modularity. The spectral gap can be bounded by the Cheeger constant (Lovász, 1993), a graph theoretic property related to the modularity of a graph—because of this connection, we will informally interpret the spectral gap as we would the Cheeger constant. In graphs with a small spectral gap, we expect more modularity with sparse connectivity between groups. In graphs with a large spectral gap, we expect the absence of obvious groups because of dense connectivity. In Fig. A.6b we plot the correlation between Degree Centrality and RRTD-RW against the spectral gap as a bivariate histogram. In graphs with large spec-

tral gaps, we find that Degree Centrality and RRTD-RW are very correlated—This is consistent with intuitive predictions for a densely connected graph, where the most useful subgoals are those with higher degree because this means they are well-connected. In graphs with small spectral gaps, we find that Degree Centrality and RRTD-RW tend to be less correlated with greater variability. This is consistent with intuitive predictions for a modular graph, where a well-connected subgoal is less useful than a bottleneck.

Appendix B

Appendix for Chapter 3

B.1 ALGORITHMS

In this section, we include an algorithm for program execution (see Alg. 1) and a generative algorithm that returns samples from the grammar induction prior (see Alg. 2).

Algorithm 1 A recursive algorithm for program execution.

Input:

Current subroutine ρ^i

Start time t

Output: End time t

for ρ_j^i in ρ^i do

if $\rho_j^i \in \mathcal{A}$ then

▷ Instruction ρ_j^i is the action a_t

$s_{t+1} \leftarrow T(s_t, \rho_j^i)$

if $s_{t+1} \in \mathcal{G}$ then

Halt program execution because a goal has been reached

end if

$t \leftarrow t + 1$

else

▷ Instruction ρ_j^i is some subroutine ρ^k

$t \leftarrow \text{recurse}(\rho_j^i, t)$

▷ Recursively execute the subroutine

end if

end for

Algorithm 2 Algorithm for sampling from the grammar induction prior. For simplicity, the programs generated by this algorithm can have an unlimited number of subroutines, though we only examine programs with four subroutines in the text.

Output: Program $\pi = (\rho^0, \rho^1, \dots)$

```

1: function INSTRUCTION()
2:   if  $true \sim \text{Bernoulli}(p_{call})$  then
3:      $k \sim p(z_{n+1} \mid z_{1:n})$  ▷ Sample subroutine, Eq. 3.2
4:     if  $\rho^k$  is not defined then ▷ Generate if new subroutine
5:       define  $\rho^k$  ▷ Defining since recursive program is possible
6:        $\rho^k \sim \text{SUBROUTINE}()$ 
7:     end if
8:     return  $\rho^k$ 
9:   else
10:     $a \sim \text{Uniform}(\mathcal{A})$  ▷ Sample from actions  $\mathcal{A}$ 
11:    return  $a$ 
12:   end if
13: end function
14: function SUBROUTINE()
15:   Initialize  $\rho$  as an empty subroutine
16:    $j \leftarrow 0$ 
17:   repeat
18:      $\rho_j \sim \text{INSTRUCTION}()$ 
19:      $j \leftarrow j + 1$ 
20:   until  $true \sim \text{Bernoulli}(p_{end})$ 
21:   return  $\rho$ 
22: end function
23:  $\rho^0 \sim \text{SUBROUTINE}()$  ▷ Initiate sampling of program

```

B.2 TRACE SEARCH

We search for traces using A* search (Hart et al., 1968), with a carefully chosen heuristic and other strategies to ensure the efficiency of searching over traces. The problem domain used to search for traces was hierarchically structured, so that at any point a particular unactivated light was the intended target for navigation. This hierarchical structure made it more straightforward to write heuristics to accelerate search, which we describe below. To further accelerate search, we required that traces consisted of actions that resulted in changes to the state, so $T(s_t, a_t) \neq s_t$. Finally, we exclude certain sequences that are redundant or symmetric: three left turns, three right turns, a right turn followed by or preceding a left turn, and light instructions after turns.

We divided the search heuristic into two parts: the cost to reach the current targeted light, and the cost to subsequently navigate among the remaining lights. To reach the current target, we optimistically model the navigation cost between lights as though the grid were unobstructed (i.e. we use the Manhattan distance and a cost for turning). Since navigating among the lights is equivalent to the traveling salesman problem, we use a simple heuristic by using these heuristic navigation costs to construct a minimum spanning tree (MST) with Prim’s algorithm. The sum of costs of the edges of the MST is an admissible heuristic for the traveling salesman problem (Held & Karp, 1970). We empirically ensured this heuristic is monotone, which ensures the results are optimal given the assumptions of A* search (Russell & Norvig, 2021).

B.3 PROGRAM GENERATION

We expand an execution trace into many possible programs that generate the trace by rewriting the trace using all combinations of possible subroutines. Subroutines are only considered if they are called in at least two places in the resulting program and contain at least two instructions. To match the experimental interface, programs could only contain at most four subroutines. Trace rewriting uses a greedy algorithm that rewrites to maximize subroutine use, considering one subroutine at a time from longest

to shortest.

We add special cases to generate programs consistent with those written by some participants. One case optionally adds post-goal actions to the trace that are consistent with some candidate subroutine. Though post-goal actions are extraneous, when produced by a subroutine, they can still result in an overall reduction in program length. A second case involves recursive subroutines—subroutines that call themselves repeatedly until the goal is reached. We generate these by proposing all trace suffixes as subroutines. A final case combines these: recursive programs with post-goal actions.

Since these programs are used to approximate the posterior, we minimize bias by ensuring that participant programs are in this corpus of programs as much as possible. We do so by searching for traces in order of ascending trace length, taking a minimum of $n = 1000$ traces but continuing to search for traces of equivalent cost to the n^{th} trace. Using A* search to find multiple solutions requires searching over state trajectories, as opposed to simply searching over states. The cost of the n^{th} trace is the maximum trace cost we consider, so participants with programs of greater cost have programs that are not in this set. Using these methods, 78% of participant programs were generated and 94% of participant programs written by at least one other participant were generated. Missing programs largely correspond to programs excluded for the purpose of efficiency, as noted above: greater than maximum trace cost, use of actions that result in no state change, use of redundant instruction sequences, and post-goal actions due to a subroutine used only once in the pre-goal program.

References

Acuna, D. E., Wymbs, N. F., Reynolds, C. A., Picard, N., Turner, R. S., Strick, P. L., Grafton, S. T., & Kording, K. P. (2014). Multifaceted aspects of chunking enable robust algorithms. *Journal of Neurophysiology*, 112(8), 1849–1856.

Akam, T., Costa, R., & Dayan, P. (2015). Simple plans or sophisticated habits? State, transition and learning interactions in the two-step task. *PLOS Computational Biology*, 11(12), 1–25.

Aldous, D. J. (1985). Exchangeability and related topics. In D. J. Aldous, I. A. Ibragimov, J. Jacod, & P. L. Hennequin (Eds.), *École d’Été de Probabilités de Saint-Flour XIII—1983*, Lecture Notes in Mathematics (pp. 1–198). Berlin, Heidelberg: Springer.

Anderson, J. R. (1990). *The Adaptive Character of Thought*. Hillsdale, NJ: Lawrence Erlbaum Associates, Inc.

Badre, D. & Frank, M. J. (2012). Mechanisms of hierarchical reinforcement learning in cortico-striatal circuits 2: Evidence from fMRI. *Cerebral Cortex*, 22(3), 527–536.

Balaguer, J., Spiers, H., Hassabis, D., & Summerfield, C. (2016). Neural mechanisms of hierarchical planning in a virtual subway network. *Neuron*, 90(4), 893–903.

Bartlett, L., Pirrone, A., Javed, N., Lane, P., & Gobet, F. (2023). Genetic programming for developing simple cognitive models. *Proceedings of the Annual Meeting of the Cognitive Science Society*, 45(45).

- Bartolo, R. & Averbeck, B. B. (2020). Prefrontal cortex predicts state switches during reversal learning. *Neuron*, 106(6), 1044–1054.e4.
- Bartolo, R. & Averbeck, B. B. (2021). Inference as a fundamental process in behavior. *Current Opinion in Behavioral Sciences*, 38, 8–13.
- Bastani, O., Pu, Y., & Solar-Lezama, A. (2018). Verifiable reinforcement learning via policy extraction. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, & R. Garnett (Eds.), *Advances in Neural Information Processing Systems*, volume 31: Curran Associates, Inc.
- Bear, A., Bensinger, S., Jara-Ettinger, J., Knobe, J., & Cushman, F. (2020). What comes to mind? *Cognition*, 194, 104057.
- Behrens, T. E. J., Woolrich, M. W., Walton, M. E., & Rushworth, M. F. S. (2007). Learning the value of information in an uncertain world. *Nature Neuroscience*, 10(9), 1214–1221.
- Bellman, R. (1957). *Dynamic programming*. Princeton University Press.
- Bertsekas, D. P. (2007). *Dynamic Programming and Optimal Control*. Athena Scientific.
- Binder, F. J., Mattar, M. G., Kirsh, D., & Fan, J. E. (2021). Visual scoping operations for physical assembly. In *Proceedings of the 43rd Annual Conference of the Cognitive Science Society*.
- Binder, F. J., Mattar, M. G., Kirsh, D., & Fan, J. E. (2023). Humans choose visual subgoals to reduce cognitive cost. *Proceedings of the Annual Meeting of the Cognitive Science Society*, 45(45).
- Binz, M. & Schulz, E. (2022). Modeling human exploration through resource-rational reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 35.
- Binz, M. & Schulz, E. (2023). Reconstructing the einstellung effect. *Computational Brain & Behavior*, 6(3), 526–542.

- Borassi, M. & Natale, E. (2019). KADABRA is an adaptive algorithm for betweenness via random approximation. *ACM J. Exp. Algorithmics*, 24(1.2), 1–35.
- Bornstein, A. M., Khaw, M. W., Shohamy, D., & Daw, N. D. (2017). Reminders of past choices bias decisions for reward in humans. *Nature Communications*, 8(1), 15958.
- Botvinick, M. M., Niv, Y., & Barto, A. G. (2009). Hierarchically organized behavior and its neural foundations: A reinforcement learning perspective. *Cognition*, 113(3), 262–280.
- Bowers, M., Olausson, T. X., Wong, L., Grand, G., Tenenbaum, J. B., Ellis, K., & Solar-Lezama, A. (2023). Top-down synthesis for library learning. *Proc. ACM Program. Lang.*, 7(POPL).
- Brunec, I. K., Bellana, B., Ozubko, J. D., Man, V., Robin, J., Liu, Z.-X., Grady, C., Rosenbaum, R. S., Winocur, G., Barense, M. D., & Moscovitch, M. (2018). Multiple scales of representation along the hippocampal anteroposterior axis in humans. *Current Biology*, 28(13), 2129–2135.e6.
- Buchsbaum, D., Griffiths, T. L., Plunkett, D., Gopnik, A., & Baldwin, D. (2015). Inferring action structure and causal relationships in continuous sequences of human action. *Cognitive Psychology*, 76, 30–77.
- Buzsáki, G. (2015). Hippocampal sharp wave-ripple: A cognitive biomarker for episodic memory and planning. *Hippocampus*, 25(10), 1073–1188.
- Callaway, F., van Opheusden, B., Gul, S., Das, P., Krueger, P. M., Griffiths, T. L., & Lieder, F. (2022). Rational use of cognitive resources in human planning. *Nature Human Behaviour*, 6(8), 1112–1125.
- Castro, P. S. & Precup, D. (2012). Automatic construction of temporally extended actions for MDPs using bisimulation metrics. In S. Sanner & M. Hutter (Eds.), *Recent Advances in Reinforcement Learning* (pp. 140–152). Berlin, Heidelberg: Springer Berlin Heidelberg.
- Chater, N. (1999). The search for simplicity: A fundamental cognitive principle? *The Quarterly Journal of Experimental Psychology Section A*, 52(2), 273–302.

- Co-Reyes, J. D., Miao, Y., Peng, D., Real, E., Le, Q. V., Levine, S., Lee, H., & Faust, A. (2021). Evolving reinforcement learning algorithms. In *International Conference on Learning Representations*.
- Collins, A. G. (2018). Learning structures through reinforcement. In R. Morris, A. Bornstein, & A. Shenhav (Eds.), *Goal-Directed Decision Making* chapter 5, (pp. 105–123). Academic Press.
- Collins, A. G. & Frank, M. J. (2013). Cognitive control over learning: Creating, clustering, and generalizing task-set structure. *Psychological Review*, 120(1), 190–229.
- Collins, A. G. E. & Cockburn, J. (2020). Beyond dichotomies in reinforcement learning. *Nature Reviews Neuroscience*, 21(10), 576–586.
- Collins, A. G. E. & Frank, M. J. (2012). How much of reinforcement learning is working memory, not reinforcement learning? A behavioral, computational, and neurogenetic analysis. *European Journal of Neuroscience*, 35(7), 1024–1035.
- Collins, A. G. E. & Frank, M. J. (2014). Opponent actor learning (OpAL): Modeling interactive effects of striatal dopamine on reinforcement learning and choice incentive. *Psychological Review*, 121(3), 337–366.
- Collins, A. G. E. & Frank, M. J. (2016). Neural signature of hierarchically structured expectations predicts clustering and transfer of rule sets in reinforcement learning. *Cognition*, 152, 160–169.
- Correa, C. G., Griffiths, T. L., & Daw, N. D. (2024). Program-based strategy induction for reinforcement learning. In *Proceedings of the 46th Annual Conference of the Cognitive Science Society*.
- Correa, C. G., Ho, M. K., Callaway, F., Daw, N. D., & Griffiths, T. L. (2023a). Humans decompose tasks by trading off utility and computational cost. *PLOS Computational Biology*, 19(6), 1–31.
- Correa, C. G., Ho, M. K., Callaway, F., & Griffiths, T. L. (2020). Resource-rational task decomposition to minimize planning costs. In *Proceedings of the 42nd Annual Conference of the Cognitive Science Society*.

- Correa, C. G., Sanborn, S., Ho, M. K., Callaway, F., Daw, N. D., & Griffiths, T. L. (2023b). Exploring the hierarchical structure of human plans via program generation. *arXiv preprint arXiv:2311.18644*.
- Şimşek, Ö., Wolfe, A. P., & Barto, A. G. (2005). Identifying useful subgoals in reinforcement learning by local graph partitioning. In *Proceedings of the 22nd International Conference on Machine Learning*.
- Cushman, F. & Morris, A. (2015). Habitual control of goal selection in humans. *Proceedings of the National Academy of Sciences*, 112(45), 13817–13822.
- Dabney, W., Kurth-Nelson, Z., Uchida, N., Starkweather, C. K., Hassabis, D., Munos, R., & Botvinick, M. (2020). A distributional code for value in dopamine-based reinforcement learning. *Nature*, 577(7792), 671–675.
- Das, R., Tenenbaum, J. B., Solar-Lezama, A., & Tavares, Z. (2023). Combining functional and automata synthesis to discover causal reactive programs. *Proc. ACM Program. Lang.*, 7(POPL).
- Dasgupta, I. & Griffiths, T. L. (2022). Clustering and the efficient use of cognitive resources. *Journal of Mathematical Psychology*, 109, 102675.
- Daw, N. D. & Courville, A. C. (2007). The pigeon as particle filter. In *Proceedings of the 20th International Conference on Neural Information Processing Systems* (pp. 369–376).
- Daw, N. D., Gershman, S. J., Seymour, B., Dayan, P., & Dolan, R. J. (2011). Model-based influences on humans’ choices and striatal prediction errors. *Neuron*, 69(6), 1204–1215.
- Daw, N. D., Niv, Y., & Dayan, P. (2005). Uncertainty-based competition between prefrontal and dorsolateral striatal systems for behavioral control. *Nature Neuroscience*, 8(12), 1704–1711.
- De Groot, A. D. (1965). *Thought and choice in chess*. Mouton Publishers.

- Dezfouli, A. & Balleine, B. W. (2012). Habits, action sequences and reinforcement learning. *European Journal of Neuroscience*, 35(7), 1036–1051.
- Dezfouli, A., Griffiths, K., Ramos, F., Dayan, P., & Balleine, B. W. (2019). Models that learn how humans learn: The case of decision-making and its disorders. *PLOS Computational Biology*, 15(6), 1–33.
- Diba, K. & Buzsáki, G. (2007). Forward and reverse hippocampal place-cell sequences during ripples. *Nature Neuroscience*, 10(10), 1241–1242.
- Dietterich, T. G. (2000). Hierarchical reinforcement learning with the MAXQ value function decomposition. *Journal of Artificial Intelligence Research*, 13, 227–303.
- Diuk, C., Tsai, K., Wallis, J., Botvinick, M., & Niv, Y. (2013). Hierarchical learning induces two simultaneous, but separable, prediction errors in human basal ganglia. *Journal of Neuroscience*, 33(13), 5797–5805.
- Donnarumma, F., Maisto, D., & Pezzulo, G. (2016). Problem solving as probabilistic inference with subgoalng: explaining human successes and pitfalls in the Tower of Hanoi. *PLoS computational biology*, 12(4), e1004864.
- Duan, Y., Schulman, J., Chen, X., Bartlett, P. L., Sutskever, I., & Abbeel, P. (2016). RL²: Fast reinforcement learning via slow reinforcement learning. *arXiv preprint arXiv:1611.02779*.
- Duff, M. O. (2002). *Optimal learning: Computational procedures for Bayes-adaptive Markov decision processes*. PhD thesis, University of Massachusetts Amherst.
- Duncan, K. D. & Shohamy, D. (2016). Memory states influence value-based decisions. *Journal of Experimental Psychology: General*, 145(11), 1420–1426.

- Duncker, L., Driscoll, L., Shenoy, K. V., Sahani, M., & Sussillo, D. (2020). Organizing recurrent network dynamics by task-computation to enable continual learning. In *Advances in Neural Information Processing Systems*, volume 33.
- Ebitz, R. B., Albarran, E., & Moore, T. (2018). Exploration disrupts choice-predictive signals and alters dynamics in prefrontal cortex. *Neuron*, 97(2), 450–461.e9.
- Eckstein, M. K. & Collins, A. G. E. (2020). Computational evidence for hierarchically structured reinforcement learning in humans. *Proceedings of the National Academy of Sciences*, 117(47), 29381–29389.
- Ellis, K., Albright, A., Solar-Lezama, A., Tenenbaum, J. B., & O’Donnell, T. J. (2022). Synthesizing theories of human language with Bayesian program induction. *Nature Communications*, 13(1), 5024.
- Ellis, K., Wong, C., Nye, M., Sablé-Meyer, M., Morales, L., Hewitt, L., Cary, L., Solar-Lezama, A., & Tenenbaum, J. B. (2021). Dreamcoder: Bootstrapping inductive program synthesis with wake-sleep library learning. In *Proceedings of the 42nd ACM SIGPLAN International Conference on Programming Language Design and Implementation*, PLDI 2021 (pp. 835–850). New York, NY, USA: Association for Computing Machinery.
- Éltető, N. & Dayan, P. (2023). Habits of mind: Reusing action sequences for efficient planning. *arXiv preprint arXiv:2306.05298*.
- Éltető, N., Nemeth, D., Janacek, K., & Dayan, P. (2022). Tracking human skill learning with a hierarchical Bayesian sequence model. *PLOS Computational Biology*, 18(11), 1–28.
- Findling, C., Skvortsova, V., Dromnelle, R., Palminteri, S., & Wyart, V. (2019). Computational noise in reward-guided learning drives behavioral variability in volatile environments. *Nature Neuroscience*, 22(12), 2066–2077.

- Frank, M. J. & Badre, D. (2012). Mechanisms of hierarchical reinforcement learning in corticostriatal circuits 1: Computational analysis. *Cerebral Cortex*, 22(3), 509–526.
- Fränken, J.-P., Theodoropoulos, N. C., & Bramley, N. R. (2022). Algorithms of adaptation in inductive inference. *Cognitive Psychology*, 137, 101506.
- Geisler, W. S. (2011). Contributions of ideal observer theory to vision research. *Vision Research*, 51(7), 771–781.
- Gershman, S. J., Horvitz, E. J., & Tenenbaum, J. B. (2015). Computational rationality: A converging paradigm for intelligence in brains, minds, and machines. *Science*, 349(6245), 273–278.
- Ghallab, M., Nau, D., & Traverso, P. (2016). *Automated planning and acting*. Cambridge University Press.
- Gittins, J. C. (1979). Bandit processes and dynamic allocation indices. *Journal of the Royal Statistical Society: Series B (Methodological)*, 41(2), 148–164.
- Goldwater, S., Griffiths, T. L., & Johnson, M. (2009). A Bayesian framework for word segmentation: Exploring the effects of context. *Cognition*, 112(1), 21–54.
- Goodman, N. D., Mansinghka, V. K., Roy, D., Bonawitz, K., & Tenenbaum, J. B. (2008a). Church: A language for generative models. In *Proceedings of the Twenty-Fourth Conference on Uncertainty in Artificial Intelligence*, UAI’08 (pp. 220–229): AUAI Press.
- Goodman, N. D., Tenenbaum, J. B., Feldman, J., & Griffiths, T. L. (2008b). A rational analysis of rule-based concept learning. *Cognitive Science*, 32(1), 108–154.
- Griffiths, T. L., Lieder, F., & Goodman, N. D. (2015). Rational use of cognitive resources: Levels of analysis between the computational and the algorithmic. *Topics in Cognitive Science*, 7(2), 217–229.

Halpern, J. Y., Pass, R., & Seeman, L. (2014). Decision theory with resource-bounded agents. *Topics in Cognitive Science*, 6(2), 245–257.

Hampton, A. N., Bossaerts, P., & O’Doherty, J. P. (2006). The role of the ventromedial prefrontal cortex in abstract state-based inference during decision making in humans. *Journal of Neuroscience*, 26(32), 8360–8367.

Hamrick, J. B., Smith, K. A., Griffiths, T. L., & Vul, E. (2015). Think again? the amount of mental simulation tracks uncertainty in the outcome. In *Proceedings of the 37th Annual Conference of the Cognitive Science Society*.

Hansen, E. A. (1998). Solving POMDPs by searching in policy space. In *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence, UAI’98* (pp. 211–219). San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

Harb, J., Bacon, P.-L., Klissarov, M., & Precup, D. (2018). When waiting is not an option: Learning options with a deliberation cost. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Hart, P. E., Nilsson, N. J., & Raphael, B. (1968). A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2), 100–107.

Hattori, R., Hedrick, N. G., Jain, A., Chen, S., You, H., Hattori, M., Choi, J.-H., Lim, B. K., Yasuda, R., & Komiyama, T. (2023). Meta-reinforcement learning via orbitofrontal cortex. *Nature Neuroscience*, 26(12), 2182–2191.

Held, M. & Karp, R. M. (1970). The traveling-salesman problem and minimum spanning trees. *Operations Research*, 18(6), 1138–1162.

Ho, M. K., Abel, D., Correa, C. G., Littman, M. L., Cohen, J. D., & Griffiths, T. L. (2022). People construct simplified mental representations to plan. *Nature*, 606(7912), 129–136.

Ho, M. K., Sanborn, S., Callaway, F., Bourgin, D., & Griffiths, T. (2018). Human priors in hierarchical program induction. In *Proceedings of the 2018 Conference on Cognitive Computational Neuroscience*.

Hoffman, M., Kück, H., de Freitas, N., & Doucet, A. (2009). New inference strategies for solving Markov decision processes using reversible jump MCMC. In J. A. Bilmes & A. Y. Ng (Eds.), *UAI 2009, Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence, Montreal, QC, Canada, June 18-21, 2009* (pp. 223–231).: AUAI Press.

Humplik, J., Galashov, A., Hasenclever, L., Ortega, P. A., Teh, Y. W., & Heess, N. (2019). Meta reinforcement learning as task inference. *arXiv preprint arXiv:1905.06424*.

Huys, Q. J. M., Eshel, N., O’Nions, E., Sheridan, L., Dayan, P., & Roiser, J. P. (2012). Bonsai trees in your head: How the Pavlovian system sculpts goal-directed choices by pruning decision trees. *PLoS Computational Biology*, 8(3), e1002410.

Huys, Q. J. M., Lally, N., Faulkner, P., Eshel, N., Seifritz, E., Gershman, S. J., Dayan, P., & Roiser, J. P. (2015). Interplay of approximate planning strategies. *Proceedings of the National Academy of Sciences*, 112(10), 3098–3103.

Iigaya, K., Fonseca, M. S., Murakami, M., Mainen, Z. F., & Dayan, P. (2018). An effect of serotonergic stimulation on learning rates for rewards apparent after long intertrial intervals. *Nature Communications*, 9(1), 2477.

Ji-An, L., Benna, M. K., & Mattar, M. G. (2023). Automatic discovery of cognitive strategies with tiny recurrent neural networks. *bioRxiv preprint 2023.04.12.536629*.

Jinnai, Y., Abel, D., Hershkowitz, D. E., Littman, M., & Konidaris, G. (2019). Finding options that minimize planning time. In *Proceedings of the 36th International Conference on Machine Learning*.

- Johnson, M., Griffiths, T., & Goldwater, S. (2006). Adaptor grammars: A framework for specifying compositional nonparametric bayesian models. In B. Schölkopf, J. Platt, & T. Hoffman (Eds.), *Advances in Neural Information Processing Systems*, volume 19: MIT Press.
- Jung, M., Wiener, S., & McNaughton, B. (1994). Comparison of spatial firing characteristics of units in dorsal and ventral hippocampus of the rat. *Journal of Neuroscience*, 14(12), 7347–7356.
- Kaelbling, L. P., Littman, M. L., & Cassandra, A. R. (1998). Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101(1), 99–134.
- Kahneman, D. & Tversky, A. (1979). Prospect theory: An analysis of decision under risk. *Econometrica*, 47(2), 263–291.
- Keramati, M., Smittenaar, P., Dolan, R. J., & Dayan, P. (2016). Adaptive integration of habits into depth-limited planning defines a habitual-goal-directed spectrum. *Proceedings of the National Academy of Sciences*, 113(45), 12868–12873.
- Klir, G. J. & Simon, H. A. (1991). *The Architecture of Complexity*. Springer.
- Korf, R. E. (1985a). Depth-first iterative-deepening: An optimal admissible tree search. *Artificial Intelligence*, 27(1), 97–109.
- Korf, R. E. (1985b). Macro-operators: A weak method for learning. *Artificial Intelligence*, 26(1), 35–77.
- Kumar, S., Correa, C. G., Dasgupta, I., Marjeh, R., Hu, M. Y., Hawkins, R., Cohen, J. D., Daw, N., Narasimhan, K., & Griffiths, T. (2022). Using natural language and program abstractions to instill human inductive biases in machines. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, & A. Oh (Eds.), *Advances in Neural Information Processing Systems*, volume 35 (pp. 167–180).: Curran Associates, Inc.

Kurth-Nelson, Z., Economides, M., Dolan, R. J., & Dayan, P. (2016). Fast sequences of non-spatial state representations in humans. *Neuron*, 91(1), 194–204.

LaFollette, K. J., Yuval, J., Schurr, R., Melnikoff, D., & Goldenberg, A. (2023). Data driven equation discovery reveals non-linear reinforcement learning in humans. *PsyArXiv preprint*.

Lai, L. & Gershman, S. J. (2021). Policy compression: An information bottleneck in action selection. In K. D. Federmeier (Ed.), *The Psychology of Learning and Motivation*, volume 74 of *Psychology of Learning and Motivation* (pp. 195–232). Academic Press.

Lai, L., Huang, A. Z., & Gershman, S. J. (2022). Action chunking as policy compression. *PsyArXiv preprint*.

Lake, B. M. & Baroni, M. (2023). Human-like systematic generalization through a meta-learning neural network. *Nature*, 623(7985), 115–121.

Lake, B. M. & Piantadosi, S. T. (2020). People infer recursive visual concepts from just a few examples. *Computational Brain & Behavior*, 3(1), 54–65.

Lake, B. M., Salakhutdinov, R., & Tenenbaum, J. B. (2015). Human-level concept learning through probabilistic program induction. *Science*, 350(6266), 1332–1338.

Lau, B. & Glimcher, P. W. (2005). Dynamic response-by-response models of matching behavior in rhesus monkeys. *Journal of the Experimental Analysis of Behavior*, 84(3), 555–579.

Lee, R. S., Sagiv, Y., Engelhard, B., Witten, I. B., & Daw, N. D. (2024). A feature-specific prediction error model explains dopaminergic heterogeneity. *bioRxiv preprint 2022.02.28.482379*.

Lefebvre, G., Summerfield, C., & Bogacz, R. (2022). A normative account of confirmation bias during reinforcement learning. *Neural Computation*, 34(2), 307–337.

- Leong, Y. C., Radulescu, A., Daniel, R., DeWoskin, V., & Niv, Y. (2017). Dynamic interaction between reinforcement learning and attention in multidimensional environments. *Neuron*, 93(2), 451–463.
- Levine, S. (2018). Reinforcement learning and control as probabilistic inference: Tutorial and review. *arXiv preprint arXiv:1805.00909*.
- Lewis, R. L., Howes, A., & Singh, S. (2014). Computational rationality: Linking mechanism and behavior through bounded utility maximization. *Topics in Cognitive Science*, 6(2), 279–311.
- Lieder, F. & Griffiths, T. L. (2017). Strategy selection as rational metareasoning. *Psychological Review*, 124(6), 762–794.
- Lieder, F. & Griffiths, T. L. (2020). Resource-rational analysis: Understanding human cognition as the optimal use of limited computational resources. *Behavioral and Brain Sciences*, 43, e1.
- Lieder, F., Griffiths, T. L., & Hsu, M. (2018). Overrepresentation of extreme events in decision making reflects rational use of cognitive resources. *Psychological Review*, 125(1), 1–32.
- Lieder, F., Krueger, P. M., & Griffiths, T. L. (2017). An automatic method for discovering rational heuristics for risky choice. In *Proceedings of the 39th Annual Conference of the Cognitive Science Society*.
- Lieder, F., Plunkett, D., Hamrick, J. B., Russell, S. J., Hay, N., & Griffiths, T. (2014). Algorithm selection by rational metareasoning as a model of human strategy selection. In *Advances in Neural Information Processing Systems*.
- Liu, X. & Halpern, J. (2020). Bounded rationality in Las Vegas: Probabilistic finite automata play multi-armed bandits. In J. Peters & D. Sontag (Eds.), *Proceedings of the 36th Conference on Uncertainty in Artificial Intelligence (UAI)*, volume 124 of *Proceedings of Machine Learning Research* (pp. 1298–1307).: PMLR.

- Liu, Y., Dolan, R. J., Kurth-Nelson, Z., & Behrens, T. E. J. (2019). Human replay spontaneously reorganizes experience. *Cell*, 178(3), 640–652.e14.
- Liu, Y., Mattar, M. G., Behrens, T. E. J., Daw, N. D., & Dolan, R. J. (2021). Experience replay is associated with efficient nonlocal learning. *Science*, 372(6544), eabf1357.
- Lovász, L. (1993). Random walks on graphs: a survey. In D. Miklós, V. T. Sós, & T. Szönyi (Eds.), *Combinatorics, Paul Erdős is Eighty*, volume 2 (pp. 353–397). Budapest: János Bolyai Math Society.
- Luce, R. D. (1959). On the possible psychophysical laws. *Psychological Review*, 66(2), 81–95.
- Luchins, A. S. (1942). Mechanization in problem solving: The effect of *einstellung*. *Psychological Monographs*, 54(6), 1–95.
- Luong, M.-T., Frank, M. C., & Johnson, M. (2013). Parsing entire discourses as very long strings: Capturing topic continuity in grounded language learning. *Transactions of the Association for Computational Linguistics*, 1, 315–326.
- Ma, T. & Hermundstad, A. M. (2022). A vast space of compact strategies for highly efficient decisions. *bioRxiv preprint 2022.08.10.503471*.
- Maisto, D., Donnarumma, F., & Pezzulo, G. (2015). Divide et impera: subgoalting reduces the complexity of probabilistic inference and problem solving. *Journal of The Royal Society Interface*, 12(104), 20141335–20141335.
- Marr, D. (1982). *Vision*. W. H. Freeman.
- Mattar, M. G. & Daw, N. D. (2018). Prioritized memory access explains planning and hippocampal replay. *Nature Neuroscience*, 21(11), 1609–1617.
- McCoy, R. T. & Griffiths, T. L. (2023). Modeling rapid language learning by distilling Bayesian priors into artificial neural networks. *arXiv preprint arXiv 2305.14701*.

- McGovern, E. A. (2002). *Autonomous discovery of temporal abstractions from interaction with an environment*. PhD thesis, University of Massachusetts Amherst.
- McNamee, D., Wolpert, D. M., & Lengyel, M. (2016). Efficient state-space modularization for planning: theory, behavioral and neural signatures. In *Advances in Neural Information Processing Systems*.
- Menache, I., Mannor, S., & Shimkin, N. (2002). Q-cut—dynamic discovery of sub-goals in reinforcement learning. In *European Conference on Machine Learning* (pp. 295–306).
- Meuleau, N., Kim, K.-E., Kaelbling, L. P., & Cassandra, A. R. (1999). Solving POMDPs by searching the space of finite policies. In *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence, UAI'99* (pp. 417–426). San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.
- Mikulik, V., Delétang, G., McGrath, T., Genewein, T., Martic, M., Legg, S., & Ortega, P. (2020). Meta-trained agents implement bayes-optimal agents. In H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, & H. Lin (Eds.), *Advances in Neural Information Processing Systems*, volume 33 (pp. 18691–18703).: Curran Associates, Inc.
- Miller, G., Galanter, E., & Pribram, K. (1960). *Plans and the Structure of Behavior*. Henry Holt and Co.
- Miller, G. A. (1956). The magical number seven, plus or minus two: Some limits on our capacity for processing information. *Psychological Review*, 63(2), 81–97.
- Miller, K. J., Botvinick, M. M., & Brody, C. D. (2017). Dorsal hippocampus contributes to model-based planning. *Nature Neuroscience*, 20(9), 1269–1276.
- Miller, K. J., Eckstein, M., Botvinick, M. M., & Kurth-Nelson, Z. (2023). Cognitive model discovery via disentangled RNNs. *bioRxiv preprint 2023.06.23.546250*.
- Montague, P., Dayan, P., & Sejnowski, T. (1996). A framework for mesencephalic dopamine systems based on predictive hebbian learning. *Journal of Neuroscience*, 16(5), 1936–1947.

- Moskovitz, T., Miller, K., Sahani, M., & Botvinick, M. M. (2023). A unified theory of dual-process control. *arXiv preprint arXiv:2211.07036*.
- Nasiriany, S., Pong, V., Lin, S., & Levine, S. (2019). Planning with goal-conditioned policies. In *Advances in Neural Information Processing Systems*.
- Newell, A. & Simon, H. A. (1972). *Human problem solving*. Prentice-Hall.
- Niv, Y. (2009). Reinforcement learning in the brain. *Journal of Mathematical Psychology*, 53(3), 139–154.
- Niv, Y., Daniel, R., Geana, A., Gershman, S. J., Leong, Y. C., Radulescu, A., & Wilson, R. C. (2015). Reinforcement learning in multidimensional environments relies on attention mechanisms. *Journal of Neuroscience*, 35(21), 8145–8157.
- O'Donnell, T. J. (2015). *Productivity and Reuse In Language: A Theory of Linguistic Computation and Storage*. The MIT Press.
- Ortega, P. A., Wang, J. X., Rowland, M., Genewein, T., Kurth-Nelson, Z., Pascanu, R., Heess, N., Veness, J., Pritzel, A., Sprechmann, P., Jayakumar, S. M., McGrath, T., Miller, K., Azar, M., Osband, I., Rabinowitz, N., György, A., Chiappa, S., Osindero, S., Teh, Y. W., van Hasselt, H., de Freitas, N., Botvinick, M., & Legg, S. (2019). Meta-learning of sequential strategies. *arXiv preprint arXiv:1905.03030*.
- Palminteri, S. (2023). Choice-confirmation bias and gradual perseveration in human reinforcement learning. *Behavioral Neuroscience*, 137(1), 78–88.
- Pan-Vazquez, A., Araujo, Y. S., McMannon, B., Louka, M., Bandi, A., Haetzel, L., Laboratory, I. B., Pillow, J., Daw, N., & Witten, I. (2024). Pre-existing visual responses in a projection-defined dopamine population explain individual learning trajectories. *bioRxiv preprint 2024.02.26.582199*.

- Parker, N. F., Cameron, C. M., Taliaferro, J. P., Lee, J., Choi, J. Y., Davidson, T. J., Daw, N. D., & Witten, I. B. (2016). Reward and choice encoding in terminals of midbrain dopamine neurons depends on striatal target. *Nature Neuroscience*, 19(6), 845–854.
- Parr, R. & Russell, S. (1997). Reinforcement learning with hierarchies of machines. In M. Jordan, M. Kearns, & S. Solla (Eds.), *Advances in Neural Information Processing Systems*, volume 10: MIT Press.
- Payne, J. W., Bettman, J. R., & Johnson, E. J. (1988). Adaptive strategy selection in decision making. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 14(3), 534–552.
- Perruchet, P. & Pacton, S. (2006). Implicit learning and statistical learning: one phenomenon, two approaches. *Trends in Cognitive Sciences*, 10(5), 233–238.
- Pezzulo, G., van der Meer, M. A. A., Lansink, C. S., & Pennartz, C. M. A. (2014). Internally generated sequences in learning and executing goal-directed behavior. *Trends in Cognitive Sciences*, 18(12), 647–657.
- Pfeiffer, B. E. & Foster, D. J. (2013). Hippocampal place-cell sequences depict future paths to remembered goals. *Nature*, 497(7447), 74–79.
- Piantadosi, S. T. (2021). The computational origin of representation. *Minds and Machines*, 31(1), 1–58.
- Piantadosi, S. T., Tenenbaum, J. B., & Goodman, N. D. (2012). Bootstrapping in a language of thought: A formal model of numerical concept learning. *Cognition*, 123(2), 199–217.
- Piray, P. & Daw, N. D. (2021). Linear reinforcement learning in planning, grid fields, and cognitive control. *Nature Communications*, 12(1), 4942.
- Planton, S., van Kerkoerle, T., Abbi, L., Maheu, M., Meyniel, F., Sigman, M., Wang, L., Figueira, S., Romano, S., & Dehaene, S. (2021). A theory of memory for binary sequences: Evidence for a mental compression algorithm in humans. *PLOS Computational Biology*, 17(1), 1–43.

- Plonsky, O., Teodorescu, K., & Erev, I. (2015). Reliance on small samples, the wavy recency effect, and similarity-based learning. *Psychological Review*, 122(4), 621–647.
- Poesia, G. & Goodman, N. D. (2023). Peano: learning formal mathematical reasoning. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 381(2251), 20220044.
- Prystawski, B., Mohnert, F., Tošić, M., & Lieder, F. (2022). Resource-rational models of human goal pursuit. *Topics in Cognitive Science*, 14(3), 528–549.
- Puterman, M. L. (1994). *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc.
- Qiu, W. & Zhu, H. (2022). Programmatic reinforcement learning without oracles. In *International Conference on Learning Representations*.
- Ramkumar, P., Acuna, D. E., Berniker, M., Grafton, S. T., Turner, R. S., & Kording, K. P. (2016). Chunking as the result of an efficiency computation trade-off. *Nature Communications*, 7(1), 1–11.
- Real, E., Liang, C., So, D., & Le, Q. (2020). AutoML-zero: Evolving machine learning algorithms from scratch. In H. D. III & A. Singh (Eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research* (pp. 8007–8019).
- Rescorla, R. A. & Wagner, A. R. (1972). A theory of Pavlovian conditioning: Variations in the effectiveness of reinforcement and nonreinforcement. In A. H. Black & W. F. Prokasy (Eds.), *Classical conditioning II: Current research and theory* (pp. 64–99). New York: Appleton-Century-Crofts.
- Ribas-Fernandes, J. J., Solway, A., Diuk, C., McGuire, J. T., Barto, A. G., Niv, Y., & Botvinick, M. M. (2011). A neural signature of hierarchical reinforcement learning. *Neuron*, 71(2), 370–379.
- Rieskamp, J. & Otto, P. E. (2006). SSL: A theory of how people learn to select strategies. *Journal of Experimental Psychology: General*, 135(2), 207–236.

- Robbins, H. (1952). Some aspects of the sequential design of experiments. *Bulletin of the American Mathematical Society*, 58(5), 527–535.
- Rosenbaum, D. A., Kenny, S. B., & Derr, M. A. (1983). Hierarchical control of rapid movement sequences. *Journal of Experimental Psychology. Human Perception and Performance*, 9(1), 86–102.
- Rule, J., Schulz, E., Piantadosi, S. T., & Tenenbaum, J. B. (2018). Learning list concepts through program induction. In *Proceedings of the Annual Meeting of the Cognitive Science Society*.
- Rule, J. S., Tenenbaum, J. B., & Piantadosi, S. T. (2020). The child as hacker. *Trends in Cognitive Sciences*, 24(11), 900–915.
- Russell, S. J. & Norvig, P. (2021). *Artificial Intelligence: A Modern Approach*. Pearson Education, Inc.
- Sacerdoti, E. D. (1974). Planning in a hierarchy of abstraction spaces. *Artificial Intelligence*, 5(2), 115–135.
- Sanborn, S., Bourgin, D. D., Chang, M., & Griffiths, T. L. (2018). Representational efficiency outweighs action efficiency in human program induction. In *Proceedings of the 40th Annual Conference of the Cognitive Science Society*.
- Schuck, N. W., Cai, M. B., Wilson, R. C., & Niv, Y. (2016). Human orbitofrontal cortex represents a cognitive map of state space. *Neuron*, 91(6), 1402–1412.
- Schuck, N. W. & Niv, Y. (2019). Sequential replay of nonspatial task states in the human hippocampus. *Science*, 364(6447), eaaw5181.
- Schwarz, G. (1978). Estimating the dimension of a model. *The Annals of Statistics*, 6(2), 461–464.
- Sharma, S., Curtis, A., Kryven, M., Tenenbaum, J. B., & Fiete, I. R. (2022). Map induction: Compositional spatial submap learning for efficient exploration in novel environments. In *International Conference on Learning Representations*.

- Shi, J. & Malik, J. (2000). Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8), 888–905.
- Silver, T., Allen, K. R., Lew, A. K., Kaelbling, L. P., & Tenenbaum, J. (2020). Few-shot bayesian imitation learning with logical program policies. In *AAAI Conference on Artificial Intelligence*.
- Simon, H. A. (1955). A behavioral model of rational choice. *The Quarterly Journal of Economics*, 69(1), 99–118.
- Şimşek, Ö. & Barto, A. G. (2009). Skill characterization based on betweenness. In *Advances in Neural Information Processing Systems*.
- Skaggs, W. E. & McNaughton, B. L. (1996). Replay of neuronal firing sequences in rat hippocampus during sleep following spatial experience. *Science*, 271(5257), 1870–1873.
- Skirzyński, J., Becker, F., & Lieder, F. (2021). Automatic discovery of interpretable planning strategies. *Machine Learning*, 110(9), 2641–2683.
- Solway, A., Diuk, C., Córdova, N., Yee, D., Barto, A. G., Niv, Y., & Botvinick, M. M. (2014). Optimal behavioral hierarchy. *PLOS Computational Biology*, 10(8), 1–10.
- Stachenfeld, K. L., Botvinick, M. M., & Gershman, S. J. (2017). The hippocampus as a predictive map. *Nature Neuroscience*, 20(11), 1643–1653.
- Starkweather, C. K., Babayan, B. M., Uchida, N., & Gershman, S. J. (2017). Dopamine reward prediction errors reflect hidden-state inference across time. *Nature Neuroscience*, 20(4), 581–589.
- Stolle, M. & Precup, D. (2002). Learning options in reinforcement learning. In S. Koenig & R. C. Holte (Eds.), *Abstraction, Reformulation, and Approximation* (pp. 212–223). Berlin, Heidelberg: Springer Berlin Heidelberg.

- Sutton, R. S. & Barto, A. G. (2018). *Reinforcement Learning: An Introduction*. Cambridge, MA: MIT Press.
- Sutton, R. S., Precup, D., & Singh, S. (1999). Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 112(1-2), 181–211.
- Thrun, S. & Pratt, L. (1998). *Learning to learn: Introduction and overview*, (pp. 3–17). Springer US: Boston, MA.
- Todd, P. M. & Gigerenzer, G. (2007). Environments that make us smart: Ecological rationality. *Current Directions in Psychological Science: A Journal of the American Psychological Society*, 16(3), 167–171.
- Tolpin, D., Zhou, Y., & Yang, H. (2020). Bayesian policy search for stochastic domains. In *International Conference on Probabilistic Programming (PROBPROG)*.
- Tomov, M. S., Yagati, S., Kumar, A., Yang, W., & Gershman, S. J. (2020). Discovery of hierarchical representations for efficient planning. *PLOS Computational Biology*, 16(4), 1–42.
- Tosatto, L., Fagot, J., Nemeth, D., & Rey, A. (2022). The evolution of chunks in sequence learning. *Cognitive Science*, 46(4), e13124.
- Toussaint, M. & Storkey, A. (2006). Probabilistic inference for solving discrete and continuous state Markov decision processes. In *Proceedings of the 23rd International Conference on Machine Learning, ICML '06* (pp. 945–952). New York, NY, USA: Association for Computing Machinery.
- Trivedi, D., Zhang, J., Sun, S.-H., & Lim, J. J. (2021). Learning to synthesize programs as interpretable and generalizable policies. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, & J. W. Vaughan (Eds.), *Advances in Neural Information Processing Systems*, volume 34 (pp. 25146–25163).: Curran Associates, Inc.

Tsividis, P. A., Loula, J., Burga, J., Foss, N., Campero, A., Pouncy, T., Gershman, S. J., & Tenenbaum, J. B. (2021). Human-level reinforcement learning through theory-based modeling, exploration, and planning. *arXiv preprint arXiv:2107.12544*.

Ullman, T. D. & Wang, Y. (2023). Resource bounds on mental simulations: Evidence from a fluid-reasoning task. *PsyArXiv preprint*.

van Opheusden, B., Kuperwajs, I., Galbiati, G., Bnaya, Z., Li, Y., & Ma, W. J. (2023). Expertise increases planning depth in human gameplay. *Nature*, 618(7967), 1000–1005.

Verma, A., Murali, V., Singh, R., Kohli, P., & Chaudhuri, S. (2018). Programmatically interpretable reinforcement learning. In J. Dy & A. Krause (Eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research* (pp. 5045–5054).: PMLR.

Verwey, W. B. (1996). Buffer loading and chunking in sequential keypressing. *Journal of Experimental Psychology. Human Perception and Performance*, 22(3), 544–562.

Von Luxburg, U. (2007). A tutorial on spectral clustering. *Statistics and Computing*, 17(4), 395–416.

Wang, J. X., Kurth-Nelson, Z., Kumaran, D., Tirumala, D., Soyer, H., Leibo, J. Z., Hassabis, D., & Botvinick, M. (2018). Prefrontal cortex as a meta-reinforcement learning system. *Nature Neuroscience*, 21(6), 860–868.

Watkins, C. (1989). *Learning from delayed rewards*. PhD thesis, University of Cambridge, England.

Wen, Z., Precup, D., Ibrahimi, M., Barreto, A., Van Roy, B., & Singh, S. (2020). On efficiency in hierarchical reinforcement learning. In H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, & H. Lin (Eds.), *Advances in Neural Information Processing Systems*, volume 33 (pp. 6708–6718).: Curran Associates, Inc.

- Wilson, R. C., Geana, A., White, J. M., Ludvig, E. A., & Cohen, J. D. (2014). Humans use directed and random exploration to solve the explore–exploit dilemma. *Journal of Experimental Psychology: General*, 143(6), 2074–2081.
- Wingate, D., Goodman, N. D., Roy, D. M., Kaelbling, L. P., & Tenenbaum, J. B. (2011). Bayesian policy search with policy priors. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence - Volume Two* (pp. 1565–1570).
- Wu, C. M., Schulz, E., Speekenbrink, M., Nelson, J. D., & Meder, B. (2018). Generalization guides human exploration in vast decision spaces. *Nature Human Behaviour*, 2(12), 915–924.
- Wu, S., Éltető, N., Dasgupta, I., & Schulz, E. (2023). Chunking as a rational solution to the speed-accuracy trade-off in a serial reaction time task. *Scientific Reports*, 13(1), 7680.
- Yang, Y. & Piantadosi, S. T. (2022). One model for the learning of language. *Proceedings of the National Academy of Sciences*, 119(5), e2021865119.
- Zhao, B., Lucas, C. G., & Bramley, N. R. (2023). A model of conceptual bootstrapping in human cognition. *Nature Human Behaviour*, (pp. 1–12).