UC Merced

Proceedings of the Annual Meeting of the Cognitive Science Society

Title

Habits of Mind: Reusing Action Sequences for Efficient Planning

Permalink

https://escholarship.org/uc/item/4gc7189b

Journal

Proceedings of the Annual Meeting of the Cognitive Science Society, 45(45)

Authors

Elteto, Noemi
Dayan, Peter

Publication Date

2023

Peer reviewed

Habits of Mind: Reusing Action Sequences for Efficient Planning

Noémi Éltető[1] **(noemi.elteto@tuebingen.mpg.de) & Peter Dayan**[1,2] **(dayan@tue.mpg.de)**

[1]Computational Neuroscience, MPI for Biological Cybernetics   [2]University of Tübingen

Abstract

When we exercise sequences of actions, their execution becomes more fluent and precise. Here, we consider the possibility that exercised action sequences can also be used to make planning faster and more accurate by focusing expansion of the search tree on paths that have been frequently used in the past, and by reducing deep planning problems to shallow ones via multi-step jumps in the tree. To capture such sequences, we use a flexible Bayesian action chunking mechanism which finds and exploits statistically reliable structure at different scales. This gives rise to shorter or longer routines that can be embedded into a Monte-Carlo tree search planner. We show the benefits of this scheme using a physical construction task patterned after tangrams.

Keywords: Planning; Sequence models; Bayesian nonparametrics; Action chunking;

Introduction

The *law of exercise* holds that animals have a tendency to repeat or perseverate on frequent choices, even when this reduces reward (Thorndike, 1898; Miller, Shenhav, & Ludvig, 2019). This leveraging of structure in behaviour complements the leveraging of structure in the environment. Gershman (2020) suggested that perseveration arises from a trade-off between reward maximization and policy complexity, with repetition making policies more compact. This theory was focused on isolated actions; however, animals also pick up on *sequences* of action. The action chunking process that underlies skill or habit learning (Dhawale, Wolff, Ko, & Ölveczky, 2021) has also been observed in multi-step decision-making tasks, for instance with rats repeating whole trajectories of choices (Dezfouli & Balleine, 2012). A hallmark of such action chunks is that they are performed quickly and uninterrupted by state assessment, that is, open-loop.

The law of exercise may apply not only to the *execution* of actions in multi-step problems, but also to the *planning* thereof. Huys et al. (2015) tested how human participants simplified a deep planning problem. One of the three heuristics they found was stochastic memoization, in which participants became increasingly prone to repeat previous action paths in their entirety or chunks of them, even though these might not be optimal. In their interpretation, this marks a transfer from flexible but costly computation to the less flexible but cheaper reliance on past experience. A similar observation was made in a task where participants planned and constructed complex structures that were required to be stable in a virtual environment simulating rigid-body physics

(W. McCarthy, Kirsh, & Fan, 2020). During training, participants' construction procedures became increasingly stereotyped.

The trouble with sequences of actions is that their number tends to grow exponentially with the length of the sequence. This makes methods that try to be comprehensive (Riegler, Polani, & Steuber, 2021) computationally infeasible. Indeed, we know of no scalable or cognitively plausible model of action chunk reuse in planning. Here, in keeping with a general trend towards internalized decision-making (Dayan, 2012), we examine the benefits of using a non-parametric Bayesian sequence model that can infer sparse chunks of various lengths, depending on the statistics of the sequences executed in past episodes. We use this model to augment Monte-Carlo Tree Search (MCTS) in two ways: using single actions that it predicts as being likely, to bias one-step tree expansions, and proposing popular multi-step expansions. We test our model on a physical construction task called the Sticky Tangram.

MCTS-with-HABITS

To examine the benefits of biasing planning towards previously learned action sequences, we used a Bayesian infinite sequence model (a hierarchical Dirichlet process) to augment Monte Carlo Tree Search with open-loop, multi-action, node expansions. We call the new planning algorithm MCTS-with-HABITS (Figure 1).

Consider the search tree in Figure 1, representing a deterministic planning problem in which nodes correspond to states and the edges correspond to actions. In Monte Carlo Tree Search (MCTS; Kocsis and Szepesvári (2006)), the tree is traversed from the root node to a leaf by repeated action choices based on their simulated expected value. This decision mechanism is called the tree policy. A leaf is any node that has a potential child whose value has not been estimated. Once a leaf is reached, it is expanded by considering the actions that are available in that state and choosing one of them. Then, the value of the node is estimated by simulating a random rollout using a uniform random policy. The reward of the rollout (in our case of a puzzle, 1 for a correct solution; 0 otherwise) is backpropagated to the selected node and its ancestors. The four ingredients – selection, expansion, simulation, and backpropagation – constitute one step of the MCTS algorithm. Since it estimates the value of nodes via simula-
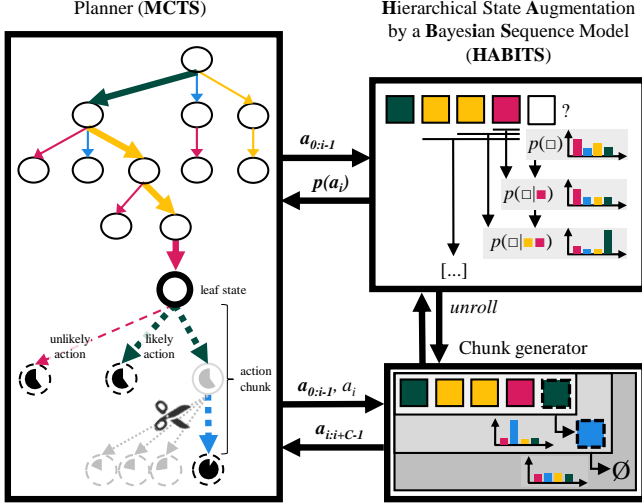
Figure 1: Schematic of the MCTS-with-HABITS model. The planner builds a search tree with nodes representing states and edges representing different possible actions, marked by colors (left box). The planner traverses the tree by choosing actions that ultimately lead to more simulated wins and that are more predictable by the HABITS sequence model (upper right box). At the leaf state (marked by the bold node) several potential actions (marked by dashed edges) are considered that lead to states (marked by dashed nodes) that have not been added to the tree yet. The *red* and the *green* actions are primitive actions whose winning values (marked as pies) are the same. Yet, the *green* action has a higher likelihood under the sequence model, when conditioned on the action trace from the root to the leaf node. Therefore, the *green* action will be preferred over *red*. The *green* action is also predictably followed, given the context of the action trace, by the *blue* action. Therefore, the chunk generator (lower right box) appends the *blue* action to the *green* one. Conditioned on the previous action trace *and* the *blue* action, no further action is strongly predicted and the chunk generation is terminated. Thus we extend the available action set by the *green-blue* chunk. Note that the primitive *green* action, from which we unrolled a chunk, is also kept as an alternative. If the action chunk is selected by the planner then it jumps over the state marked by a grey node, not considering the available actions from that state, also marked by grey. This effective stunting results in a different value estimate for the chunk *green-blue* (marked by the larger pie in the node) compared to the primitive action *green*; in this example, the chunk will be preferred by the tree policy.

tion, MCTS focuses the search on promising paths without the need for handcrafted distance heuristics as in A*. In order to balance this exploitative focus (from the fraction $\frac{w_t}{n_t}$ of wins w_t from n_t tries from a node on the tth MCTS step) with exploration of potentially even better states, a bonus is added to actions leading to lesser tried nodes ($\sqrt{\frac{lnN_t}{n_t}}$, where N_t is

the total number of simulations run from the parent node; as in the UCT algorithm).

To this conventional tree policy, we add a habit value term, that is, the scaled likelihood of the action a planned to get to a node, making its total propensity be:

$$Q(a) = \frac{w_t}{n_t} + c * \sqrt{\frac{lnN_t}{n_t}} + h * p_t(a_i = a|a_0 : a_{i-1}) \quad (1)$$

where c is the exploration coefficient, here fixed to 1; h is the habit coefficient; a_i is the action directly leading to the node; and $a_0 : a_{i-1}$ is the path of all preceding planned actions from the root, where i indicates the current depth in the tree. Note that the win proportion and exploration terms are node-dependent, while the habit term is node-independent. Probability $p_t(a_i = a|a_0 : a_{i-1})$ comes from a Bayesian non-parametric sequence model that flexibly combines the predictive power of dependencies at variable depths (Teh, 2006). The model was recently used to explain motor skill learning in humans (Élteto̊, Nemeth, Janacsek, & Dayan, 2022)). Here, we adapt it to the task of hierarchical state augmentation (HABITS) in which it provides a flexible window onto the action path leading to a state, for determining the likelihood of the next action in the search (upper right box of Figure 1). Its predictions are based on previous action sequence outputs of the planner. Formally (temporarily dropping the iteration index t), the model is an hierarchical Dirichlet process (HDP):

$$a_i \sim \mathbf{G}_{\mathbf{u}_i} \sim HDP(\alpha, \mathbf{G}_{\pi(\mathbf{u}_i)}) \quad (2)$$

where $\mathbf{G}_{\mathbf{u}_i}$ is the vector of action probabilities given the context $\mathbf{u}_i$ of some previous actions before the i-th action; α is a strength parameter, controlling the resemblance to the base distribution and, therefore, the speed of sequence learning; and the base distribution $\mathbf{G}_{\pi(\mathbf{u}_i)}$, which is the probability distribution over actions given the truncated context $\pi(\mathbf{u}_i)$ containing all but the earliest action. Applying Equation 2 recursively performs a weighted smoothing across the action probabilities conditioned on preceding action chunks of shrinking sizes. For more details and algorithms, see Élteto̊ et al. (2022). In sum, the sequence model flexibly augments the states with a *weighted* window onto the previous actions, efficiently utilizing the ones that have predictive power.

Assume that the HABITS module has been trained on action sequence outputs of the planner from previous episodes (irrespectively of the quality of those plans). Then, the HABITS module interacts with the planner in two ways. It allows for biasing the one-step search towards predictable action sequence paths through the habit term in Equation 1. Moreover, by unrolling the sequence model, we also extend the node's action repertoire by action chunks (lower right box in Figure 1). Action chunks were created by sampling and attaching successor actions while the conditional entropy of the distribution over the next action was lower than the open-loop threshold parameter ω. When the action chunks were considered by the tree policy, their habit value was determined by

the predictability of the first action in the chunk. By choosing an action chunk, the agent jumped deeper in the tree without adding the intermediate node(s) to the tree and evaluating them, effectively stunting all the other branches that would have grown from the intervening nodes. As such, the MCTS-with-HABITS can be viewed as a hybrid between open- and closed-loop. The complexity of the chunks proposed by the HABITS scaled with the complexity of the sequences that the agent executed in past episodes, opportunistically saving as much planning cost as possible.

Our experiments tested the unique benefits of the two separate mechanisms: the sequence bias in the one-step search, realized by the habit term in the tree policy, and the open-loop action chunks, realized by unrolling the sequence model until the open-loop entropy threshold ω is reached. Therefore, we compared three variants of our model: MCTS-with-HABITS$_{full}$, utilizing both one-step biases and open-loop action chunks; MCTS-with-HABITS$_{open-loop}$, utilizing no one-step bias for the actions but having the alternative of unrolling primitive actions into chunks; and MCTS-with-HABITS$_{one-step}$ having no action chunk alternatives but utilizing sequence predictions to bias its action choices; along with the control model, MCTS$_{vanilla}$. For brevity, we will refer to MCTS-with-HABITS$_{full}$ as MCTS-with-HABITS. Within the scope of this paper, we did not optimize the model parameters, but selected values for demonstration (Table 1).

Table 1: Parameter settings for four alternative model variants. The node budget b, exploration coefficient c and sequence learning parameter α were fixed across the model variants, while we varied the habit coefficient h and the open-loop threshold ω.

Model name	b	c	α	h	ω
MCTS-with-HABITS$_{full}$	50	1	1	5	1.5
MCTS-with-HABITS$_{open-loop}$	50	1	1	0	1.5
MCTS-with-HABITS$_{one-step}$	50	1	1	5	0
MCTS$_{vanilla}$	50	1	-	0	0

Apart from the labelled additions to the tree policy, the MCTS algorithm was used conventionally, looping through the node selection, expansion, rollout with uniform random policy, and backpropagation of a binary value (success or failure) to the selected node and its ancestors.

Results

Task

We tested our model on a physical construction task inspired by the classical tangram puzzle, versions of which were recently employed by Bapst et al. (2019) and W. P. McCarthy, Mattar, Kirsh, and Fan (2021). Our version is called the Sticky Tangram (Figure 2A). The agent had to plan a construction that matched a target silhouette, using the seven building blocks in the inventory. The blocks could be placed

into a grid space which discretized and constrained the state space. The coarse discretization is essential in order to keep the puzzle in a moderate complexity regime. The blocks had to be placed in such a sequence that they touched each other, i.e. floating blocks were not allowed.
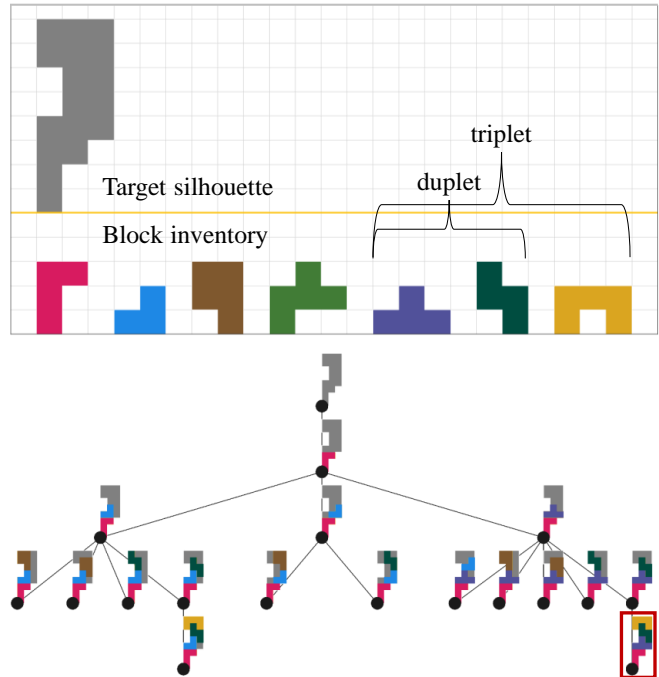

Figure 2: (A) Schematic of an example trial of the Sticky Tangram task. The silhouette (grey area) is the target that should be built up using the building blocks (shown in different colors). The first chosen block has to be placed on the floor (yellow line); the following blocks have to be placed such that they touch at least a unit segment of a previous block's edge (hence the 'stickiness'). In the duplet condition, two blocks always occurred together, in the same relative position; in the triplet condition, three blocks formed such a chunk. Throughout the paper, these two particular chunks will be used as examples. (B) The full search tree of the example problem in (A). The leaf node in red brackets is the only solution, the other leaf nodes represent dead ends.

A state included the target silhouette and the action history of the relative positions of the previously placed building blocks (Figure 2B). Adding a block to the construction constituted a primitive action; adding several blocks at once constituted an action chunk. In any state, only valid actions were considered – that is, block placements that were inside the borders of the target silhouette and that resulted in no disjoint parts. Note that this applied to action chunks as well, meaning that actions that were unrolled into chunks that would have violated the rules of the game were discarded.

Experiments

Reusing variable-size chunks

We randomly generated silhouettes that were composed of four building blocks (Figure 2A). In the duplet condition, two building blocks formed a chunk such that they would always occur together, in the same relative position and had to be placed in a fixed order. In the triplet condition, three formed such a chunk. The unchunked blocks occurred in unpredictable relative positions. 'Chunky silhouettes' comprised the duplet and two random unchunked building blocks in the duplet condition or the triplet and one random unchunked building block in the triplet condition.

In the chunky silhouettes, the marginal probabilites of the chunked blocks were 1, but in the unchunked blocks they were < 1. That is because the elements of the chunks were *always* present in the chunky silhouettes, per definition, but the other blocks were not. Since we wanted to study the advantage of *sequence structure* in planning problems, we ensured that the overall marginal probabilities of the blocks were equal, and a chunked block was more predictable only conditioned on the other elements of the chunk. Therefore, we introduced 'random silhouettes' comprising only unchunked blocks. A 4:3 ratio of the chunky and random silhouettes in the training set ensured that the marginal probability distribution of the blocks was uniform – that is, the agent used all the blocks equally frequently for the correct solutions.

Since the chunky silhouettes were generated using more constraints than the random ones (i.e. the chunk constraint), on average, the full search trees of chunky silhouettes are less complex. We quantified the problem complexity as the median number of nodes that the $MCTS_{vanilla}$ evaluated until finding the solution. Then we sampled sets of the chunky and random silhouettes such that they were matched by their complexity. We restricted the problem complexity to be ≤ 50. The example silhouette shown in Figure 2A and B has a full search tree of 18 nodes and a low complexity of 12.

We trained four MCTS variants on 19 trials (a randomized sequence of chunky and random silhouettes). During training, all model variants used a budget of 50 nodes, enabling nearly perfect performance due to the silhouettes having complexity values ≤ 50 (Figure 3A and E). In the case of the three MCTS-with-HABITS model versions, the sequence module was trained on the action sequence output of the planner. We ran 32 simulations for both the duplet and triplet conditions, from different random seeds, and averaged the results.

All model variants showed nearly perfect performance given a node budget sufficient for the problems ($> 96\%$ in the duplet condition and $> 94\%$ in the triplet condition; Figure 3A and E), albeit with different patterns of chunk use. The sequence model of the MCTS-with-HABITS model versions gradually extracted the structure in the action sequences generated by the planner, that is, the action duplets and triplets in the respective conditions. The MCTS-with-HABITS and MCTS-with-HABITS_open-loop models began to propose action chunks and these chunks were used more often in the
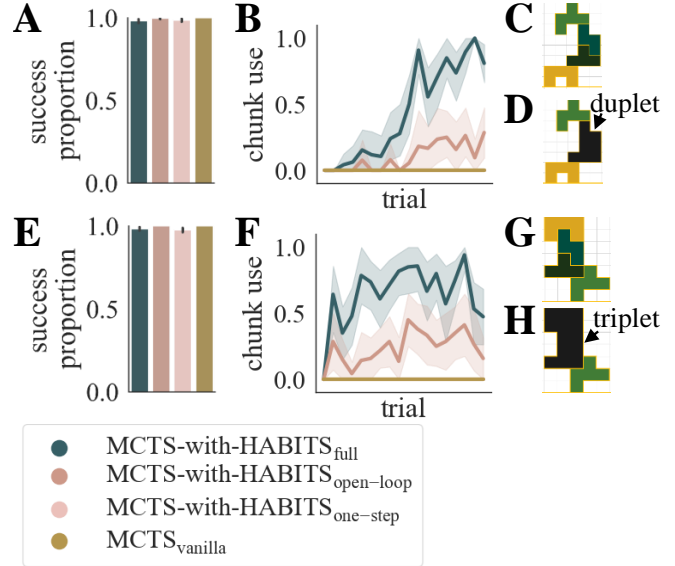

Figure 3: Emergence of action sequence reuse in the duplet condition (A-B) and the triplet condition (C-D). Error bars and bands indicate the 95% CI. (A)(E) All model variants had nearly perfect performance both in the duplet and triplet conditions. (B)(F) The two model versions that were permitted to use open-loop action chunks gradually picked up on the structure in the action sequences and became more likely to use an action chunk for the solution. In the triplet condition (F), chunk use emerged even faster than in the duplet condition (B). (C)(G) Examples of a correct solutions by the $MCTS_{vanilla}$. (D)(H) Examples of solutions by the MCTS-with-HABITS model, using an action duplet (D) and an action triplet in (H). The black polygons mark chunks.

correct solutions throughout the training, both in the duplet (effect of *trial*: $F(1, 696) = 293.70$, $p < .001$; Figure 3B) and in the triplet conditions (effect of *trial*: $F(1, 712) = 29.51$, $p < .001$; Figure 3F). In both conditions, the MCTS-with-HABITS was more disposed to include chunks in its plans (*model* effect: $F(1, 696) = 207.58$, $p < .001$ for the duplet condition; $F(1, 712) = 150.39$, $p < .001$ for the triplet condition). Problems that were solved in four primitive steps by the $MCTS_{vanilla}$ (Figure 3C and G) were solved by the MCTS-with-HABITS in only three steps in the duplet condition (Figure 3D) and in two steps in the triplet condition (Figure 3 H). Note that action chunks were reused in spite of the same states never reoccurring across episodes since the sequence model proposes chunks in a state-independent manner. In sum, the MCTS-with-HABITS model found and utilized the largest exploitable chunks that reoccurred in planned action sequences.

Solving high complexity problems with less compute

After the training trials, we gradually reduced the node budget from the initial value of 50 to 12, 8, 5, and 1. In order to compare performance across these node budgets, we froze

Duplet condition

A Random silhouettes   B Chunky silhouettes

Triplet condition

C Random silhouettes   D Chunky silhouettes

- MCTS-with-HABITS$_{full}$
- MCTS-with-HABITS$_{open-loop}$
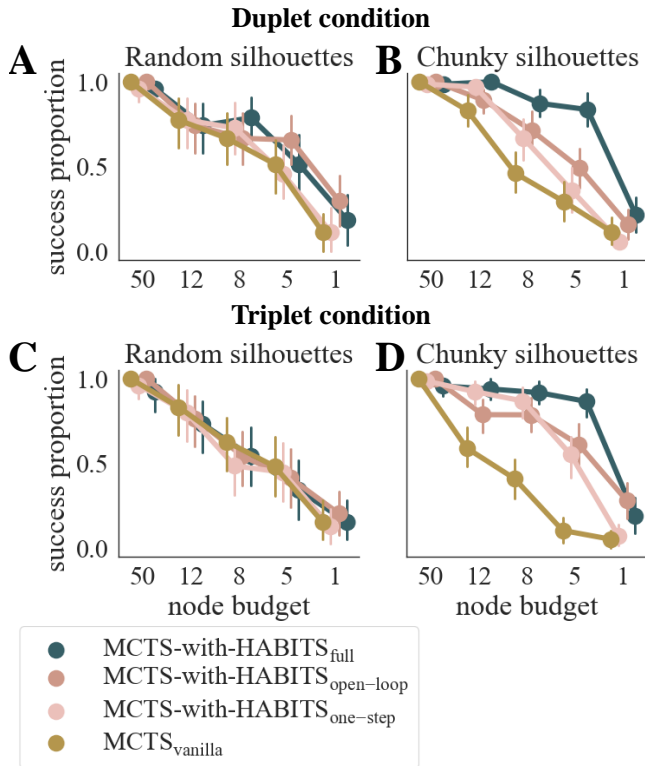- MCTS-with-HABITS$_{one-step}$
- MCTS$_{vanilla}$

Figure 4: Model performance in the face of node budget restriction in the test phase of Experiment 1, in the duplet condition (A-B) and in the triplet condition (C-D). Error bars indicate the 95% CI. (A)(C) Performance on random silhouettes decreased for all model variants to similar degrees due to the node budget restriction. (B)(D) For chunky silhouettes, the performance of the MCTS-with-HABITS$_{open-loop}$ and MCTS-with-HABITS$_{one-step}$ was more resilient to the node budget restriction than that of the MCTS$_{vanilla}$. The full MCTS-with-HABITS model showed the best performance in the face of node budget restriction.

the sequence model at the test trials. At each reduced node budget value, the agents solved four silhouettes. We ran 32 simulations for both the duplet and triplet conditions separately, from different random seeds, and averaged the results.

When solving random silhouettes, all model variants' performance dropped to similar degrees as a function of node budget restriction (effect of *budget*: $F(1, 1872) = 124.58$, $p < .001$; effect of *model*: $F(1, 1872) = 0.42$, $p < .73$; Figure 4A and C). In the case of the chunky silhouettes, when previously learned action sequences were appropriately reusable, the resource-constrained model performance depended on whether the model was allowed to use one-step biases, action chunks, or both; this held both in the duplet (effect of *model*: $F(1, 1244) = 17.03$, $p < .001$; Figure 4B) and triplet conditions (effect of *model*: $F(1, 1188) = 57.16$, $p < .001$; Figure 4D).

We then analyzed the planning performance in the

Duplet condition

A Random silhouettes   B Chunky silhouettes

Triplet condition
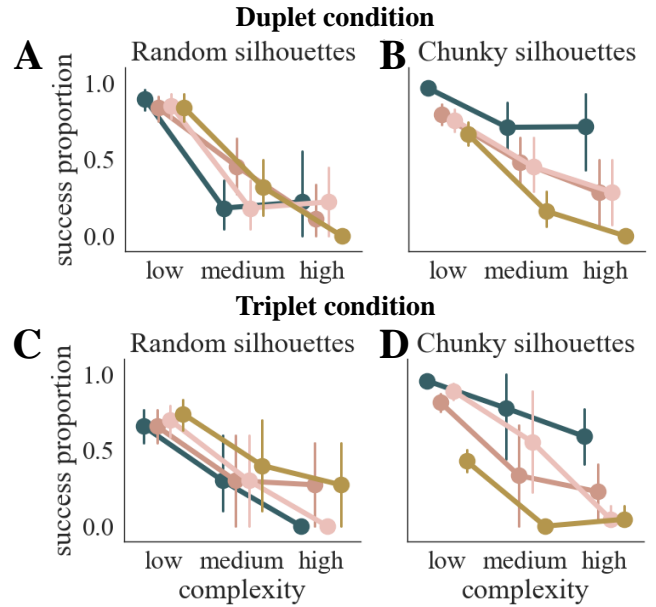
C Random silhouettes   D Chunky silhouettes

Figure 5: Model performance as a function of problem complexity in the test phase of Experiment 1, in the duplet condition (A-B) and in the triplet condition (C-D). Error bars indicate the 95% CI. Color coding is identical to that of Figure 4.

resource-restricted regime of 12, 8, and 5 nodes, as a function of problem complexity. In the case of the random silhouettes, the different model variants' performance scaled with the problem complexity similarly, both in the duplet (*model*complexity* interaction: $F(6, 384) = 1.72$, $p = .11$; Figure 5A) and triplet conditions (*model*complexity* interaction: $F(6, 364) = .52$, $p = .78$; Figure 5C). However, performance on the chunky silhouettes scaled differently with complexity across the model variants, in both conditions (*model*complexity* interaction in duplet condition: $F(6, 744) = 1.89$, $p = .07$; Figure 5B; *model*complexity* interaction in triplet condition: $F(6, 764) = 4.07$, $p < .001$; Figure 5D). This suggests that the *effective* complexity of problems was reduced by relying on past action sequence statistics.

Solving ambiguous problems

We trained the model variants on chunky silhouettes that enforced an action chunk – that is, the chunked blocks had to be placed in fixed relative positions and order for the correct solution. Here, we test the model variants on so-called 'sequence-ambiguous silhouettes' that have multiple solutions allowing for placing the chunked blocks in different orders (Figure 6A and E). Crucially, such sequence-breaking was unlikely under the sequence model, making the sequence-breaking alternative a lower-value plan.

Indeed, solutions containing chunks were preferred by both MCTS-with-HABITS and MCTS-with-HABITS$_{open-loop}$ models (Figure 6C and G). We allowed for a flexible node budget and measured the number of node evaluations
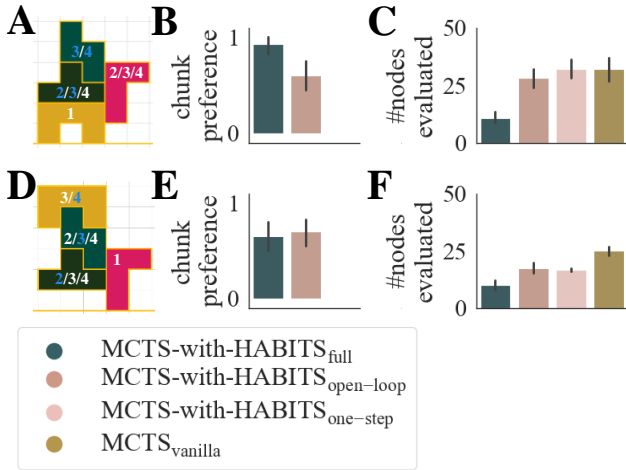
Figure 6: Action chunk reuse in sequence-ambiguous problems in the duplet (A-C) and triplet conditions (D-F). Error bars indicate the 95% CI. (A)(D) Example silhouettes whose correct solutions allowed for flexible ordering of the chunk elements. The digits indicate the possible serial positions of blocks in different solutions. The blue digits indicate the serial positions of the blocks in solutions that preserved the chunked order (same example chunks as in Figure 3). (B)(E) Solutions containing chunks were preferred by both model variants that were enabled to generate chunks. (C)(F) The MCTS-with-HABITS$_{full}$ model evaluated the least number of nodes until finding a solution.

performed until the solution was reached (Figure 6D and H). In both the duplet and triplet conditions, the number of evaluated nodes was significantly different between the model variants, and the lowest in the case of the MCTS-with-HABITS (*model* effect: $F(3, 156) = 25.86$, $p < .001$ and *model* effect: $F(3, 156) = 42.99$, $p < .001$. To conclude, our model prefers to reuse past action sequences even when other alternatives are viable and this confers it the computational benefit of planning with smaller trees.

Discussion

Discovering the building blocks of a problem is key to its efficient solution. The behavior of animals suggest that they form action chunks that they flexibly employ for solving novel tasks. We proposed that such actions chunks can be usefully integrated into planning as well, effectively reducing the depth of problems. We introduced the MCTS-with-HABITS, a model that finds and leverages action sequence patterns at variable scales and integrates them into the planning process in order to save computational costs.

We tested the model on a physical construction task that was constrained such that certain blocks were placed in predictable positions relative to each other. Our method used a flexible Bayesian sequence model, which can find predictable action sequences of different sizes in past episodes, as induced by the task constraints, which it can then reuse for the

planning problem at hand. We allowed the sequence model to influence the one-step search inherent to MCTS, and also to propose multi-step node expansions to the planner. Both mechanisms gave the model an edge in performance over the baseline model when tested under resource-constraints. We showed that this was related to the model being more economical with its node budget and searching deeper on predictable paths via the multi-step expansions.

Various extensions to the method are possible. For instance, in our current algorithms, the learned action sequences only influenced MCTS' tree policy, that is, the search among the nodes whose values were evaluated. One could also use the sequence model to inform the rollout policy, potentially improving the simulation step. Equally, it would be possible to store information about the complexity of subsequent planning with nodes, and use this to adjust the use of the sequence model depending on overall demands of planning space or time.

In the current work, the construction task that our model was tested on was more complex than typical psychology experiments but less complex than the ones many real-life applications pose. Testing the model on more complex problems and problems with partially transferable structure will be revealing about the scalability of our approach. It would be particularly interesting to look at stochastic problems, and those with intermediate rewards, which exploit more of the power of MCTS.

Our model extracts and reuses action patterns in past behavior. Another class of approaches focuses on past state visitation patterns. McGovern and Barto (2001) proposed that so-called bottleneck states can be identified at rare state transitions and that they are useful subgoals that link subproblems. Several methods based on the successor representation (Machado et al., 2017; Machado, Barreto, Precup, & Bowling, 2021) use the eigendecomposition of state succession patterns to carve the environment into subproblems. However, these approaches require the agent to have explored the entire state space extensively, a requirement that often cannot be met in machine learning and robotics.

Applying the law of exercise to the realm of planning was inspired by the cognitive science literature. How model-free (Daw, Niv, & Dayan, 2005) or value-free (Miller et al., 2019) statistics interact with model-based reasoning has been a long-standing interest in the field. Here we proposed a hybrid model that flexibly adjusts the weighting of value-based and value-free assessments by relying on predictably repeated chunks of behavior. Such sequence reuse yields a solution for optimizing the direction and depth of search (Sezener, Dezfouli, & Keramati, 2019; Kuperwajs & Ma, 2021), reminiscent of humans' resource-rational strategies (Callaway et al., 2022). Future experiments should investigate whether humans indeed use habits for planning, whether they utilize such habits of mind adaptively in order to save planning costs, and, if so, whether their planning is well described by the model presented here.

Acknowledgments

References

Bapst, V., Sanchez-Gonzalez, A., Doersch, C., Stachenfeld, K., Kohli, P., Battaglia, P., & Hamrick, J. (2019). Structured agents for physical construction. In *International conference on machine learning* (pp. 464–474).

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.

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.

Dayan, P. (2012). How to set the switches on this thing. *Current opinion in neurobiology*, 22(6), 1068–1074.

Dezfouli, A., & Balleine, B. W. (2012). Habits, action sequences and reinforcement learning. *European Journal of Neuroscience*, 35(7), 1036–1051.

Dhawale, A. K., Wolff, S. B., Ko, R., & Ölveczky, B. P. (2021). The basal ganglia control the detailed kinematics of learned motor skills. *Nature Neuroscience*, 24(9), 1256–1269.

Éltető, N., Nemeth, D., Janacsek, K., & Dayan, P. (2022). Tracking human skill learning with a hierarchical bayesian sequence model. *bioRxiv*.

Gershman, S. J. (2020). Origin of perseveration in the trade-off between reward and complexity. *Cognition*, 204, 104394.

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

Kocsis, L., & Szepesvári, C. (2006). Bandit based monte-carlo planning. In *European conference on machine learning* (pp. 282–293).

Kuperwajs, I., & Ma, W. J. (2021). Planning to plan: a bayesian model for optimizing the depth of decision tree search. In *Proceedings of the annual meeting of the cognitive science society* (Vol. 43).

Machado, M. C., Barreto, A., Precup, D., & Bowling, M. (2021). Temporal abstraction in reinforcement learning with the successor representation. *arXiv preprint arXiv:2110.05740*.

Machado, M. C., Rosenbaum, C., Guo, X., Liu, M., Tesauro, G., & Campbell, M. (2017). Eigenoption discovery through the deep successor representation. *arXiv preprint arXiv:1710.11089*.

McCarthy, W., Kirsh, D., & Fan, J. (2020). Learning to build physical structures better over time. In *Cogsci*.

McCarthy, W. P., Mattar, M. G., Kirsh, D., & Fan, J. E. (2021). Connecting perceptual and procedural abstractions in physical construction. In *Proceedings of the annual meeting of the cognitive science society* (Vol. 43).

McGovern, A., & Barto, A. G. (2001). Automatic discovery of subgoals in reinforcement learning using diverse density.

Miller, K. J., Shenhav, A., & Ludvig, E. A. (2019). Habits without values. *Psychological review*, 126(2), 292.

Riegler, B., Polani, D., & Steuber, V. (2021). Embodiment and its influence on informational costs of decision density—atomic actions vs. scripted sequences. *Frontiers in Robotics and AI*, 8, 535158.

Sezener, C. E., Dezfouli, A., & Keramati, M. (2019). Optimizing the depth and the direction of prospective planning using information values. *PLoS computational biology*, 15(3), e1006827.

Teh, Y. W. (2006). A hierarchical bayesian language model based on pitman-yor processes. In *Proceedings of the 21st international conference on computational linguistics and 44th annual meeting of the association for computational linguistics* (pp. 985–992).

Thorndike, E. L. (1898). Animal intelligence: An experimental study of the associative processes in animals. *The Psychological Review: Monograph Supplements*, 2(4), i.