Neural sampling from cognitive maps supports goal-directed planning and imagination

(May 14, 2025)

Hui Lin[1,2], Yukun Yang[2], Rong Zhao[1], Giovanni Pezzulo[3], Wolfgang Maass[2,†]

[1]Department of Precision Instruments, Center for Brain-Inspired Computing Research (CBICR), Tsinghua University, Beijing, China..
[2]Institute of Machine Learning and Neural Computation, Graz University of Technology, Graz, 8010, Austria.
[3]Institute of Cognitive Sciences and Technologies, National Research Council, Rome, Italy.
[†]Corresponding author.

Contributing authors: linhui21@mails.tsinghua.edu.cn; yukun.yang@tugraz.at; giovanni.pezzulo@istc.cnr.it; maass@igi.tugraz.at;

Abstract

Imagining plans for solving problems is a cornerstone of our higher cognition. We can even create plans for reaching goals that we never encountered before. However, the ways the brain addresses goal-directed imagination is largely unknown and current AI methods provide limited solutions to this problem. Here, we introduce a novel computational model – the generative cognitive map learner (GCML) – that successful addresses goal-directed imagination and provides a new hypothesis about the brain mechanisms supporting it. The GCML uses stochastic samples from learned cognitive maps – data structures in which brains encode learned relations between actions and states – to form trajectories towards both known and novel goals. In a series of simulations, we show that the GCML provides surprisingly effective heuristic solutions to spatial navigation tasks, generic problem solving tasks, and compositional tasks that require generating imagined trajectories to novel goals. The sampled trajectories capture key characteristics of hippocampal replay activity, which provide a candidate mechanism for goal-directed imagination in the brain. Since the GCML only requires simple biologically plausible local synaptic plasticity and shallow neural networks, it can be readily implemented in energy-efficient neuromorphic hardware.

1 Introduction

We have the remarkable ability to set up goals, including those that we never experienced before, and imagine sequences of actions to achieve them. This capability is apparent in multiple domains, such as when planning routes to reach goal locations or when imagining solutions to novel and abstract problems. However, we still lack a mechanistic understanding of how the brain supports goal-directed imagination and our current AI systems fall short providing flexible and efficient solutions to this problem.

An emerging perspective in neuroscience is that goal-directed planning and imagination are based upon learned *cognitive maps* – data-structures that encode relations between states and actions – among which the most widely studied are allocentric spatial maps in the hippocampal formation (Behrens et al., 2018; Moser et al., 2008; O'Keefe and Nadel, 1978; Raju et al., 2024). Hippocampal cognitive maps support goal-oriented spatial navigation, by providing self-localization and a sense

of direction towards goals (Bush et al., 2015; Epstein et al., 2017; Tsoar et al., 2011). Cognitive mapping mechanisms might be in play also beyond the spatial domain. Converging evidence suggests that hippocampal cognitive maps could have been re-purposed and expanded in the human brain to enable navigation and mental travel also in abstract concept spaces (Behrens et al., 2018; Bottini and Doeller, 2020; Buzsáki and Moser, 2013; Tavares et al., 2015; Xiao et al., 2025).

Crucially, various lines of evidence suggest that cognitive maps might also support goal-directed planning and imagination, beyond previous experience. A rodent study showed that spontaneously generated ("replay") sequences of place cells in the hippocampus before the animal navigates a familiar 2D environment depict imagined trajectories resembling future plans (Pfeiffer and Foster, 2013). Notably, when novel barriers are introduced in the environment, replay trajectories depict goal-directed trajectories around them and largely avoid barrier violations, without changes in hippocampal place fields (Widloski and Foster, 2022). Hippocampal replay can also represent novel experiences, by recombining previous experiences in compositional ways. For example, in rodents, replay activity can show never experienced "shortcuts" between two connected parts of a maze (Gupta et al., 2010). In humans, replay activity was associated with compositional tasks, such as deciding whether a 2D silhouette can be decomposed into a known set of building blocks (Schwartenbeck et al., 2023). These and other studies suggest that replay from (spatial and non-spatial) cognitive maps provide a neural substrate for goal-directed imagination and compositional computations, but few studies so far advanced mechanistic explanations of these phenomena (Bakermans et al., 2025; Foster, 2017; Kurth-Nelson et al., 2023; Stoianov et al., 2022; Wittkuhn et al., 2021). Current AI planners addressing goal-directed imagination and compositional computations require solving NP-hard problems (Moore and Robson, 2001; Russell and Norvig, 2016), for which no known algorithms can avoid using computational resources that grow exponentially with problem size. In sum, understanding how the brain supports goal-directed imagination and compositional computations and devising devising AI algorithms that provide efficient solutions to these problems are still open challenges.

In this study, we address these challenges by leveraging an existing model of cognitive map formation – the *cognitive map learner (CML)* (Stöckl et al., 2024) – and extending it with the capability to support goal-directed imagination and compositional computations. We use the CML as a starting point, given its appealing capability to form cognitive maps by learning predictions of action outcomes, through local and biologically-plausible synaptic plasticity rules. Furthermore, and importantly, the emergent geometrical structure of cognitive maps provides a *sense of direction* towards the current goal, such as the desired location in spatial navigation. This implies that to reach a distal goal, it suffices to choose an action whose embedding into the state space points towards the goal.

The key innovation of this study consists in extending the CML model into a probabilistic generative model – which we call *generative cognitive map learner (GCML)* – supporting goal-directed imagination. For this, we replace action-dependent next state observations with predicted observations and impose noise on the heuristic action selection. This novel mechanism permits stochastically sampling possible trajectories in the cognitive map to reach both known and novel goals. From a biological perspective, our move is motivated by experimental evidence showing a dual role of hippocampal cognitive maps, for learning states in an associative mode and supporting sequential activity in a predictive mode (Liu et al., 2023). The GCML represents a novel hypothesis about the generative, cognitive map-based mechanisms used by the brain for goal-directed imagination.

In the remaining of this article, we first describe the novel GCML model. Then, in a series of simulations, we show that the GCML supports efficient goal-directed imagination in spatial navigation (Simulation 1), generic problem solving (Simulation 2) and challenging compositional tasks involving novel start and goal states (Simulation 3). Furthermore, we show that stochastic samples from the GCML resemble replay trajectories observed empirically, hence suggesting a candidate mechanism for goal-directed imagination in the brain.

2 Results

2.1 The generative cognitive map learner (GCML): learning cognitive maps and using them for solving problems

The starting point of our study is the cognitive map learner (CML): a model for learning cognitive maps through exploration of a physical (or mental) space (Stöckl et al., 2024). In the CML, sensory inputs (observations) and own actions are embedded into a common high-dimensional space (Figure 1a). In biological terms, this high-D space might correspond to the space of neural activity patterns

in the hippocampus (or another region, like the parietal cortex) – a coding space whose dimension is equal to the number of neurons in that area. However, a much smaller dimension – of around 1000 – suffices for the tasks considered in this study. Similarly, it suffices that we employ linear maps $\mathbf{Q}$ and $\mathbf{V}$ that embed into this high-D coding space sensory inputs and own actions, respectively, and that we employ simple local synaptic plasticity rules to learn these embeddings during exploration.

The fundamental principle used by the CML model to encode the relationships between observations and actions into the cognitive map is called the *predictive coding goal*. It consists in optimizing $\mathbf{Q}$ and $\mathbf{V}$ so that they enable predicting the next observation from the current observation and action:

$$\mathbf{Q}\mathbf{o}_{t+1} \approx \mathbf{Q}\mathbf{o}_t + \mathbf{V}\mathbf{a}_t \tag{1}$$

where $\mathbf{a}_t$ is the action carried out at time step t and $\mathbf{o}_t$ is the observation at time step t.

As shown in (Stöckl et al., 2024), a gradient descent for this predictive coding principle gives rise to a simple local rule for for updating the weights of the linear embeddings $\mathbf{Q}$ and $\mathbf{V}$ (see Equation 5 and Equation 6 in Section 4 for details). Furthermore, the CML learns through Hebbian learning a matrix $\mathbf{W}$ that maps state differences $\mathbf{s}_{t+1} - \mathbf{s}_t$ onto actions a_t that cause them, see Fig. 1b. After learning $\mathbf{Q}$, $\mathbf{V}$, and $\mathbf{W}$, the CML can be tasked with reaching an arbitrary goal state $\mathbf{s}^* = \mathbf{Q}\mathbf{o}^*$ that results from a desired observation $\mathbf{o}^*$; for example, a goal location for spatial navigation or, more generally, a problem to solve.

A key property of the CML model is that the learned matrix $\mathbf{W}$ can be used to select actions that are likely to be useful for reaching this goal state $\mathbf{s}^*$. More precisely, $\mathbf{W}$ assigns to any possible action $\mathbf{a}_t$ a utility that scores its estimated usefulness (or value) for reaching $\mathbf{s}^*$ from $\mathbf{s}_t$. If $\mathbf{s}^*$ was already reached from $\mathbf{s}_t$ during learning through a single action $\mathbf{a}_t$, $\mathbf{W}$ assigns the highest utility to $\mathbf{a}_t$. If $\mathbf{s}^*$ cannot be reached from $\mathbf{s}_t$ with a single action, $\mathbf{W}$ assigns the highest utility to the action whose embedding $\mathbf{V}$ best points into the direction of $\mathbf{s}^*$. More precisely, multiplying the vector $\mathbf{s}^* - \mathbf{s}_t$ with the matrix $\mathbf{W}$ produces for every possible action $\mathbf{a}$ an estimate of the action's utility for moving from s_t into the direction of $\mathbf{s}^*$. These utility estimates provide a *sense of direction* to the CML model.

The utility estimates of the CML are analogous to value estimates in reinforcement learning (Sutton, 1992; Sutton and Barto, 2018). However, the learned matrix $\mathbf{W}$ provides a universal value function that assigns values for reaching any possible state. Further, the CML does not require the computationally expensive and time consuming iteration of value estimates in order to propagate value estimates for states adjacent to the goal state to more distal states. However, similar to RL, in the CML model it is necessary to multiply the utility estimate of an action by an *affordance* factor that vetoes actions that cannot be executed in the current state (or to increase or decrease the value of actions that are easy or challenging to execute in the current state). We denote the product of the current utility of an action (for reaching some goal $\mathbf{s}^*$) with this affordance factor as its current *eligibility*. At each time point, goal-directed navigation can be realized by simply selecting the action $\mathbf{a}_t$ with the highest eligibility. After action execution, the CML makes a transition to a new state $\mathbf{s}_{t+1}$, which it can estimate from the observation $\mathbf{o}_{t+1}$ it receives (using $\mathbf{s}_{t+1} = \mathbf{Q}\mathbf{o}_{t+1}$) and from which it can iterate its action selection process, until it reaches the goal.

Extending the CML model to goal-directed planning and imagination – or the generation of a sequence of goal-directed actions without executing them – simply requires using predictions instead of observations to infer future states. Specifically, after generating an action $\mathbf{a}_t$, the model does not receive any observation $\mathbf{o}_{t+1}$, but it estimates its future (imagined) state $\hat{\mathbf{s}}_{t+1} = \mathbf{s}_t + \mathbf{V}\mathbf{a}_t$ and continue from there to generate the next action. Using this mechanism permits generating, for any given initial state $\mathbf{s}_t$ and any given goal state $\mathbf{s}^*$, an imagined trajectory $\mathbf{s}_t$, $\hat{\mathbf{s}}_{t+1}$, $\hat{\mathbf{s}}_{t+2}$, ... in the direction of the goal $\mathbf{s}^*$, without receiving any observation; see Figure 1c for an illustration and Figure 1d for the underlying generative model based on bootstrapping of state predictions.

Finally, by introducing noise into this imaginary trajectory generation process – more specifically, by adding random values from Gaussian distribution to the current estimate of the eligibility of each action – it is possible to obtain a probabilistic generative model that samples from the set of trajectories that start from an initial state $\mathbf{s}_t$ and reach a goal state $\mathbf{s}^*$. We call such probabilistic generative model the *generative cognitive map learner (GCML)*.

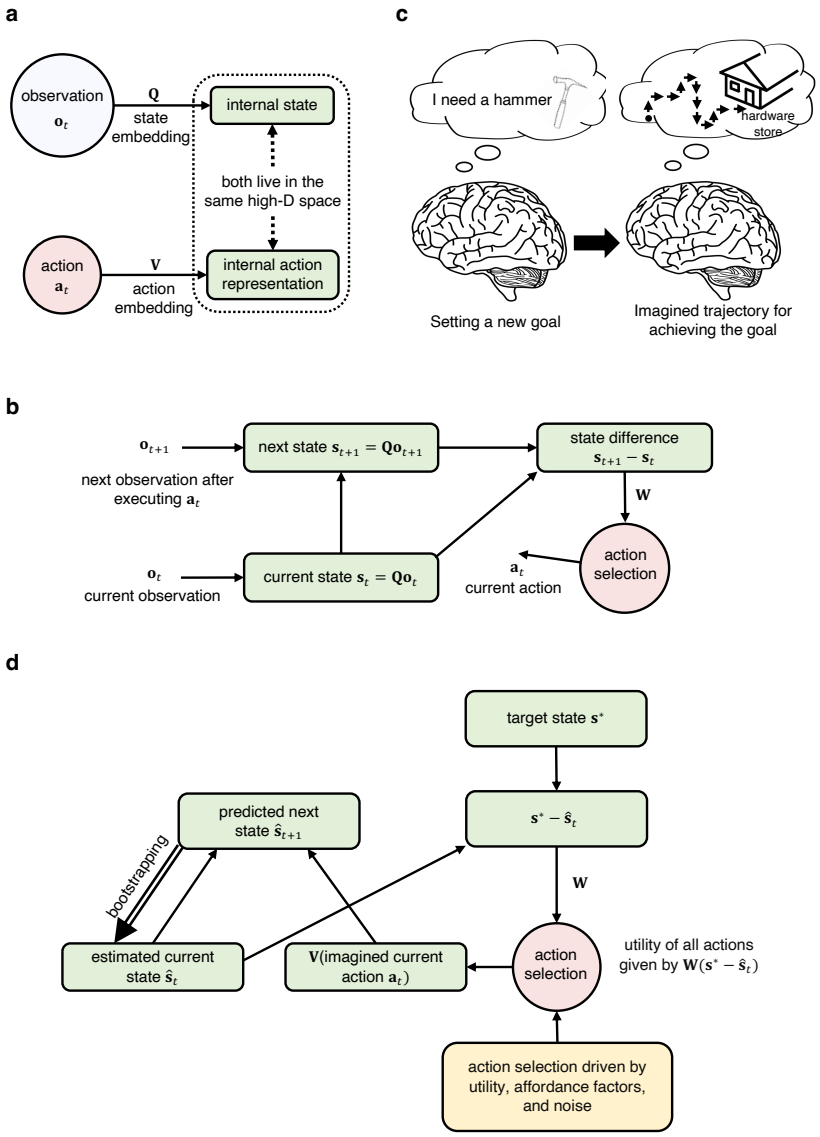

Fig. 1: **From the cognitive map learner (CML) to the generative cognitive map learner (GCML).** (a) The key mechanism supporting cognitive map learning in the CML model consists in embedding observations and (efferent copies of) actions into a common high-D state space, whose geometrical structure captures their relations (Stöckl et al., 2024). Suitable linear embeddings $\mathbf{Q}$ and $\mathbf{V}$ can be learned by simple local rules for synaptic plasticity (b) Learning mechanisms of the CML model. The model learns predictions of the next observation that results from a selected action. Furthermore, the model learns an inverse in the form of the matrix $\mathbf{W}$, which maps state differences into actions which caused them, see (Stöckl et al., 2024) for details. (c) Illustration of goal-directed imagination. In this example, a specific goal problem ("I need a hammer") is solved by imagining a solution path. This path may involve various types of actions, such as navigating to a hardware store to buy the hammer. (d) Key step from the CML to the GCML: Actions are selected like in the CML on the basis of how well they point into the direction of the goal (utility) and affordance ("can this action be carried out in the current state?"). However, importantly, no actions are carried out. Rather, predictions $\hat{\mathbf{s}}_{t+1}$ of resulting next observations (states) replace actual feedback from the environment. The next action is virtually selected for the next predicted state (bootstrapping). The difference between a goal state $\mathbf{s}^*$ and the estimated current state $\hat{\mathbf{s}}_t$ drives action selection through the learned matrix $\mathbf{W}$ like for real action selection, thereby enabling goal-directed generation of a sequence of future actions and state estimates. See Figures S1 and S2 for schematics of neural network implementations of the learning and planning processes of the GCML and Section B for details on algorithmic implementation.

2.2 Simulation 1. The GCML supports goal-directed imagination of spatial trajectories

The study of (Pfeiffer and Foster, 2013) addressed replay during rodent spatial navigation and planning in an open arena. A key finding of the study is that before navigation towards a known goal location, the hippocampus showed replay trajectories from the current to the goal locations. The decoded trajectories exhibited a startling diversity, even for the same starting and goal locations, and were rarely straight (see Fig. 2 and Suppl. Fig. 11 and 17 of (Pfeiffer and Foster, 2013)), resembling samples from a generative model (Stoianov et al., 2022; Ujfalussy and Orbán, 2022).

We simulated the same navigation scenario using the GCML (Figure 2). In the first, learning phase, the GCML explores the environment (a 30×30 grid) to create a cognitive map, using the CML approach (Stöckl et al., 2024). The model can execute four possible actions, which can be interpreted (after learning) as move up, move down, move left, move right; it does not have any prior knowledge of the effects of each action. Through learning, the high-dimensional states of $\mathbf{Qo}$ become progressively organized into 30×30 cognitive map (a 2D projection of the map is shown in Figure S3).

In the subsequent, test phase, we assign arbitrary start $\mathbf{s}_{t+1}$ and goal locations $\mathbf{s}^*$ and test the model ability to generate imaginary trajectories from the current to the goal locations. Our simulations shows that the GCML generates imaginary trajectories that are qualitatively similar to those observed in the hippocampus (Pfeiffer and Foster, 2013), for any given start and goal location (Figure 2).

As shown in Figure 2a, each sample corresponds to a different trajectory towards the goal. The variance of imagined trajectories for a fixed start and goal can be regulated by the level of noise that is superimposed on utility estimates for selecting virtual actions (Figure 2d). Given its simplicity, the GCML model can also be implemented with spiking neurons. Figure 2d shows the spiking activity of a selected subset of neurons during generation of the imagined trajectory above.

Furthermore, the GCML reproduces another intriguing index of the generative nature of hippocampal replay trajectories: the fact that they include path segments that were never experienced during navigation (Gupta et al., 2010). This type of generalization can also be observed in the GCML model, see the red marked segments in Figure 2c that had never been used during or after exploration.

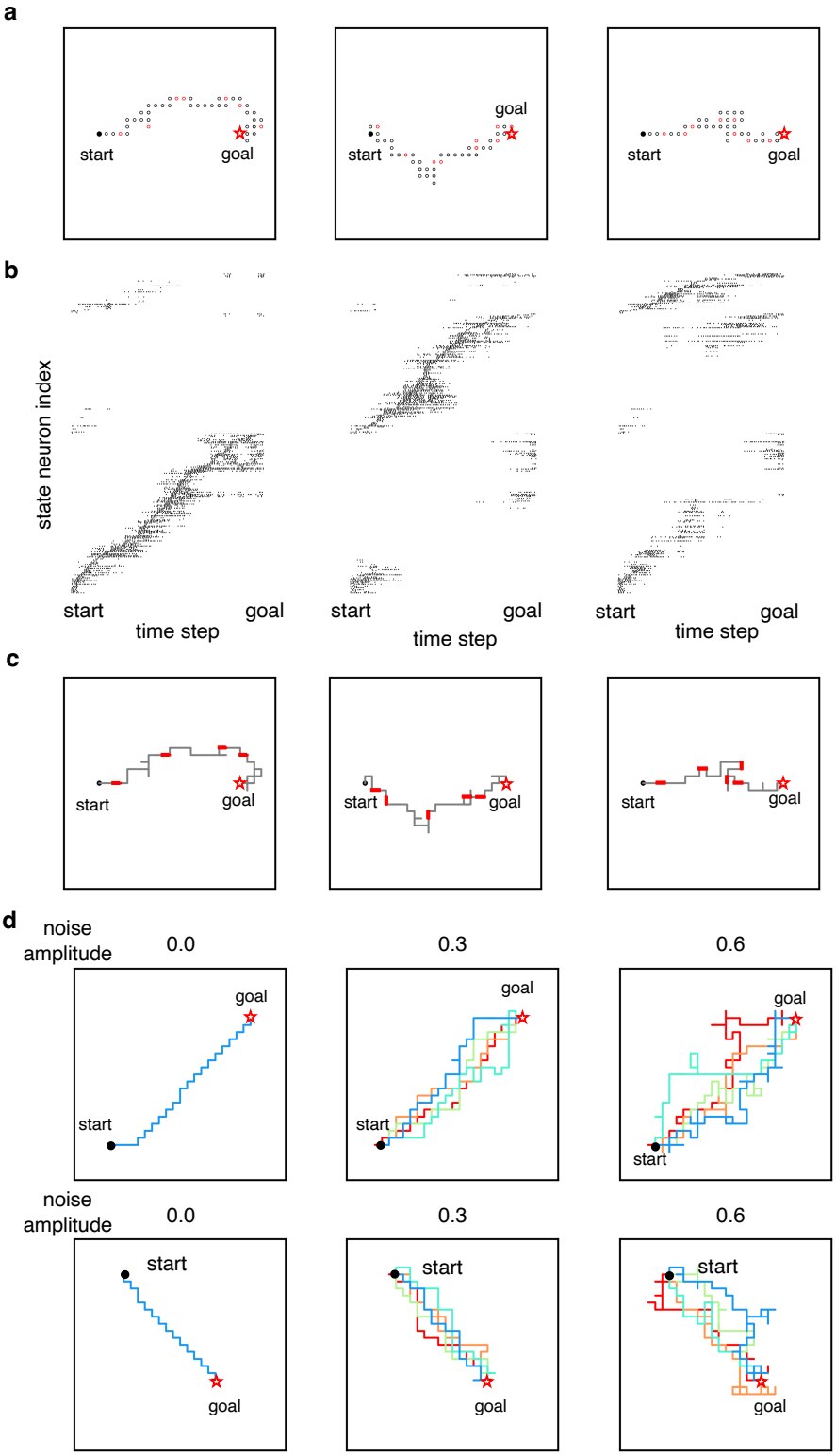

Fig. 2: **The GCML supports goal-directed imagination of spatial trajectories.** (a) Three trajectories generated by the GCML from start to goal, under noise scale 0.6. The black dot indicates the start position. The red star denotes the goal position. The red dots are previously unvisited nodes in the training process. (c) The red edges are previously unvisited edges (i.e., unvisited actions) employed by the 3 trajectories shown in panel a. (d) Impact of the noise levels (0, 0.3 and 0.6) during action selection on the variance of the generated trajectories, shown for two different pairs of start and goal locations.

To sum up, this simulation illustrates that after learning a cognitive map, the GCML model supports goal-directed imagination of trajectories from start to goal locations, capturing key qualitative patterns of rodent hippocampal replay in the same setting.

2.3 Simulation 2. The GCML supports goal-directed imagination for generic problem solving and constraint satisfaction problems

In the previous simulation, we exemplified the functioning of the GCML in a spatial navigation domain. Here, we show that the GCML can address generic problem solving tasks, beyond navigation.

Problem solving is commonly modeled in AI and cognitive science as the task to find a short path from a start to a goal node in an graph – where nodes problems states and edges represent possible actions (Geffner and Bonet, 2013; Hills et al., 2015; Newell et al., 1972; Russell and Norvig, 2016). Consider the generic random graph with 32 nodes shown in Figure 3a. Finding a short path from start to goal in this and other graphs non-trivial size is a challenging AI problem, but the GCML provides an efficient heuristic solution to it. This is because, it uses the intrinsic "sense of direction" of the learned cognitive map (see Figure 3b) to plan ahead with foresight.

The efficiency of the GCML can be appreciated by considering that it provides a heuristic online solution to the well-known k-shortest path problem (Eppstein, 1998), where the objective is finding for some k the k shortest paths from a given start to a given goal. As shown in Figure 3c, most trajectories generated by the GCML in a low-noise condition (noise = 0.15) have the shortest possible length. The top five shortest paths generated by GCML with low noise are shown in Figure S4. They represent all the possible shortest paths of length five between the given start and goal nodes, demonstrating that GCML provides a good approximation of the top-k shortest path algorithm. Higher noise levels (noise = 0.25) yield slightly longer paths, as expected (Figure 3d). Of note, the remarkable noise tolerance of the GCML is an added benefit of the model's inherent sense of direction. Even if a bad move is initially selected as an effect of noise, it is later compensated by choosing on average subsequent correct actions that point towards the goal. This feature implements a self-correcting homing-in heuristic and ensures that even in the presence of noise, the goal-directed paths tend to be short.

Interestingly, the capability to generate not just one but a diverse set of solutions turns out to be very useful to solve *constraint satisfaction* problems, which include constraints not represented in the graph structure. For example, planning in operations research needs to consider start-up costs for using certain tools, machines, or human resources, that make it desirable not to switch them too often, or to involve them preferably in a particular sequential order. To address these problems, it is useful to generate a menu of different solutions (paths) and then evaluate them with regard to their desirability (Russell and Norvig, 2016).

To test the capability of the GCML to address constraint satisfaction problems, we associate rewards (red) or losses (blue) to the nodes of the graph studied above (Figure 3e). Then, we test how much reward the GCML achieves with various noise levels, ranging from 0 to 0.4. For this, we let the GCML generate 40 trajectories for each noise level and select the trajectory with the highest cumulative reward. We repeat this procedure 100 times, to obtain a distribution of results for each noise level. Our results show that as the noise increases, the GCML is able to collect more rewards. Rather, at very low levels, it privileges shortest paths associated with lower reward. Therefore, manipulating noise levels permits effectively spanning the path length-reward trade-off (Figure 3f). These result go beyond the capabilities of previous neural networks models that used sampling to address constraint satisfaction problems, such as the Traveling Salesman Problem, Boolean Formula Satisfiability, or Sudoku problems (Buesing et al., 2011; Habenschuss et al., 2013; Jonke et al., 2016), but required the concrete problem instance that to be programmed into the network, through its synaptic weights. In contrast, the GCML can be used to design solution paths to any node in a graph. Another practical disadvantage of earlier neural sampling models was that they only produced a set of elements for a possible solution, not a sequence of solution steps.

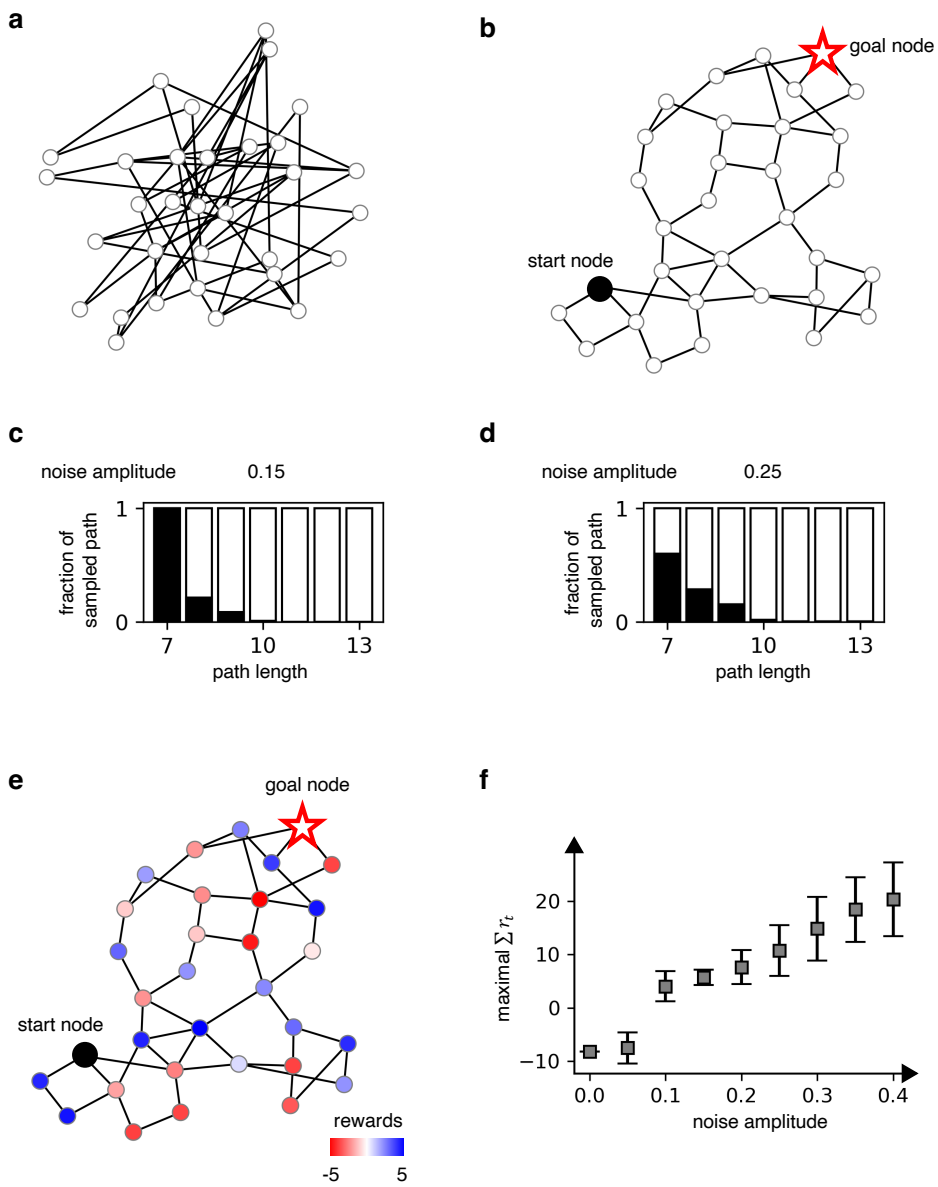

Fig. 3: **Imagining solutions to problems by finding a path from start to goal nodes in a graph.** (a) A random graph with 32 nodes. (b) 2D projection (via t-sne) of the cognitive map that the CML creates after exploring this graph. As apparent in the figure, the geometry of the graph supports heuristic action selection – or the possibility of choosing actions towards the goal node – for virtually all possible start and goal nodes. (c) and (d) show the length statistics of paths that generated by the GCML for a generic choice of start and goal node (see panel b). Panel c shows that a sufficiently small noise level tends to produce virtually all paths of minimal length. 40 trajectories were generated by the GCML for each noise amplitude. (e) If there are additional criteria besides length for evaluating the quality of a path, e.g. maximizing the sum of rewards for the nodes that are visited, higher noise amplitudes in the GCML tend to be beneficial. In panel f we show as an example the maximal sum of rewards obtained by one of 40 trajectories generated by the GCML for any given level of noise. The error bars indicate the standard deviation over 100 times repeated experiments with different random seeds. The starting node and goal node are the same as in b and e.

From a biological perspective, the mechanism used by the GCML – the stochastic generation of multiple paths and their subsequent selection based upon expected reward – could be associated with the neural circuit formed by the hippocampus and the ventral striatum. Various studies suggest that

this circuit supports goal-directed navigational planning and "vicarious trial-and-error", with the hippocampus generating candidate future trajectories and the ventral striatum assessing the reward associated with those trajectories (Pennartz et al., 2011; Pezzulo et al., 2014; Redish, 2016; Van Der Meer et al., 2012). Furthermore, during offline periods, the hippocampus and the ventral striatum show a coordinated replay, supporting place-reward associations (Lansink et al., 2009).

The capability of generating samples also allows the GCML to readily adapt to unforeseen changes in the graph structure, without retraining the cognitive map. To show this, we removed two edges from start to goal from the graph considered in this study (Figure S5). When generating paths from start to goal, the GCML finds novel paths that avoid the missing edges – a capability not present in the CML algorithm from which the GCML originates (Stöckl et al., 2024). Intriguingly, besides showing the flexibilty of the GCML, this capability provides a potential explanation for the puzzling finding that hippocampal replay trajectories show flexible rerouting around novel barriers, without changes (remapping) in place fields (Widloski and Foster, 2022). Another effective way to adapt the GCML to the changed graph structure without retraining is by using barrier-related affordances – an information that, in a biological setting, might be related to barrier cells (Rivard et al., 2004). This would allow the GCML to generate trajectories that avoid the missing edges, without novel experience.

To sum up, this simulation illustrates that the GCML can address generic problem solving and constraint satisfaction problems. The cognitive maps learned by the GCML afford a geometrical heuristic for choosing actions "with foresight". Furthermore, the GCML can exploit noise to generate multiple candidate solutions that can be evaluated in relation to constraints not included in the graph structure. A limitation of the tasks addressed in this and the previous simulations is that the GCML had already experienced the start and the goal locations. The next simulation addresses a more challenging task, in which start and goal states are novel.

2.4 Simulation 3. The GCML can plan through states never encountered before using compositional cognitive maps

Humans are able to imagine plans to achieve truly novel goals, such as sitting together with a goat on the Schafberg mountain, when they receive a verbal (or otherwise symbolic) description of such goals. In terms of cognitive maps, these goals correspond to states never encountered before – therefore, the cognitive map structure around them could not be shaped by experience.

We asked whether it is possible to apply the geometrical heuristic and the inherent sense of direction of the GCML for goal-directed imagination, even when the start or the goal states lie in a previously not encountered part of the state space. We had already encountered in Fig. 2c the case where learned relational information that was instantly generalized beyond prior experience: never encountered path segments (actions) between states were automatically integrated into imagined paths. But can a cognitive map provide insights for planning movements between states that were never encountered before?

We reasoned that this might be possible by endowing the GCML with cognitive maps having a compositional structure, with known tokens that can be assembled together to compose an unlimited set of never-encountered states. To test this possibility, we used a task recently adopted in a magnetoencephalography (MEG) study aimed at understanding how humans solve compositional planning tasks (Schwartenbeck et al., 2023). Participants were asked to decide whether a given silhouette (in black in Figure 4a) could be decomposed using a given set of building blocks (Figure 4b). The study showed that during the decision, participants considered the possible building blocks that could be removed from the silhouette in a sequential manner, starting with the most obvious candidates. The authors suggested that replay sequences supported the hypothesis testing process of silhouette decomposition.

This problem may appear to be computationally easier than natural language processing, but it is actually NP-hard (Moore and Robson, 2001). This implies that there is no known algorithm that can solve this task without using computational resources that grow exponentially with the size of the silhouette. Furthermore, it is questionable whether such algorithm can ever be found for our current generation of digital computers, even for silhouettes that are just composed of the first 4 building blocks shown in Fig. 4b.

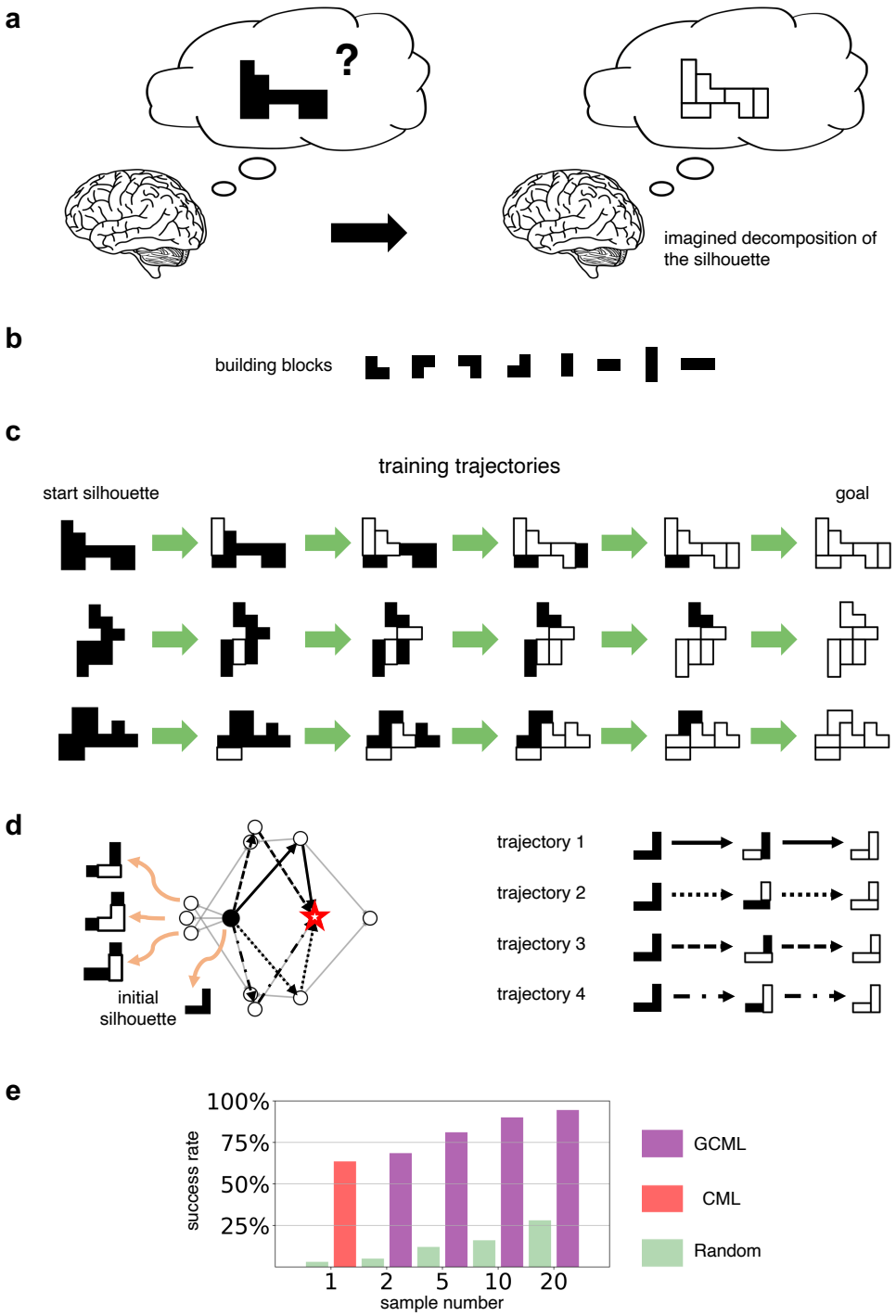

Fig. 4: Application of the GCML to goal-directed sampling for a compositional benchmark task. (a, b) Illustration of the task: Find a decomposition of the silhouette into the building blocks shown in panel b, provided there exists such a decomposition. (c) Three example trajectories that were used for training. The five BBs used to generate each silhouette were removed in the randomly shuffled order. (d) 2D PCA projection of the resulting cognitive map for the embedding Q of silhouettes with affordance factors. (e) Test of the ability of the GCML to generate decompositions for decomposable silhouettes consisting of 8 building blocks, after training on decomposition of silhouettes with 5 building blocks. Hence none of the test silhouettes had been encountered by the GCML before. See Figure S9 for additional examples of learned cognitive maps and Figures S8 and S10 for example solutions to the silhouette decomposition problem.

Here, we use the GCML to address the silhouette decomposition problem illustrated in (Schwartenbeck et al., 2023), but we remove one of the building blocks (an atomic square tile) that renders the task too easy. We first train the GCML with example trajectories from a start silhouette to a goal silhouette, where the actions correspond to removing building blocks. The training problems require removing 5 building blocks. Figure 4c shows three example training problems, in which the building blocks that have already been removed are indicated in white with a black outline. This visual representation indicates the temporal decomposition process quite well, but is slightly deceptive from a mathematical perspective, because removed tiles are no longer represented in intermediate or final silhouettes. In particular, the goal state is always the same one: an empty silhouette. Hence in this case the goal state is familiar, but the start state is in general novel.

Following the hypothesis that generating paths between never encountered states benefits from compositional cognitive maps, we made the embedding $\mathbf{Q}$ of observations (here, 2D silhouettes) compositional. The simplest possible compositional embedding $\mathbf{Q}$ is the identity map over binary vectors that have a 1 in each position in the vector that corresponds to a position in the 2D grid that is black in the given silhouette. However, we found that the cognitive map acquires a better sense of direction if the embedding $\mathbf{Q}$ slightly departs from this simple compositional structure and assigns greater weights to black pixels surrounded by white pixels, see Section 4.4 for details and Figure S10 for examples. Using this heuristic increases the utility of removing black pixels that protrude from a silhouette, where the choice of a building block is more constrained and therefore less error-prone. A mathematically equivalent way is assigning a greater affordance to actions that remove pixels protruding from the silhouette.

Figure 4d shows a 2D projection of (a portion of) the cognitive map learned during training. In the map, the black node represents the start silhouette and the red start represents the goal (and empty silhouette). The four paths indicated with solid, dotted, dashed and dashdotted edges from start to goal correspond to four correct trajectories that solve the problem, shown to the right. All these correct trajectories include nodes that lie between the start and goal states. The left part of the figure also shows that three silhouettes that result from removing a building block from the starting silhouette, but do not allow solving the whole decomposition problem, lie on the opposite direction. The structure of the cognitive map therefore permits solving the silhouette decomposition problem using the geometrical heuristic illustrated in the previous simulations, which consists in selecting actions towards the goal, even if the paths pass through never encountered states. Additional examples of cognitive maps and generated trajectories to decompose silhouettes are shown in Figures S9, S8, and S10.

Remarkably, after training to solve example problems requiring 5 building blocks (Figure 4c), the GCML is able to decompose silhouettes that it never encountered before and that were composed of 8 building blocks (Figure 4e). The figure shows that with any number of samples, the GCML largely outperforms a baseline algorithm that removes building blocks randomly and the CML model. Furthermore, the GCML shows a remarkable, near perfect success rate, even with few samples.

Finally, the GCML endowed with the same compositional embedding can solve two more challenging variants of the silhouette decomposition task, which were not considered in (Schwartenbeck et al., 2023) (Figure 5). The first variant (partial decomposition task) consists in providing as a goal a partial decomposition of the starting silhouette instead of the empty silhouette – therefore making the goal state novel (Fig. 5a). Figure 5c shows that the GCML solves this more general task very well. The second variant (rebuilding task) consists in providing as a goal a silhouette that requires adding building blocks as opposed to only removing them (Figure 5b). Figure 5d shows that the GCML can solve also this more challenging task quite well, although at a somewhat lower performance level.

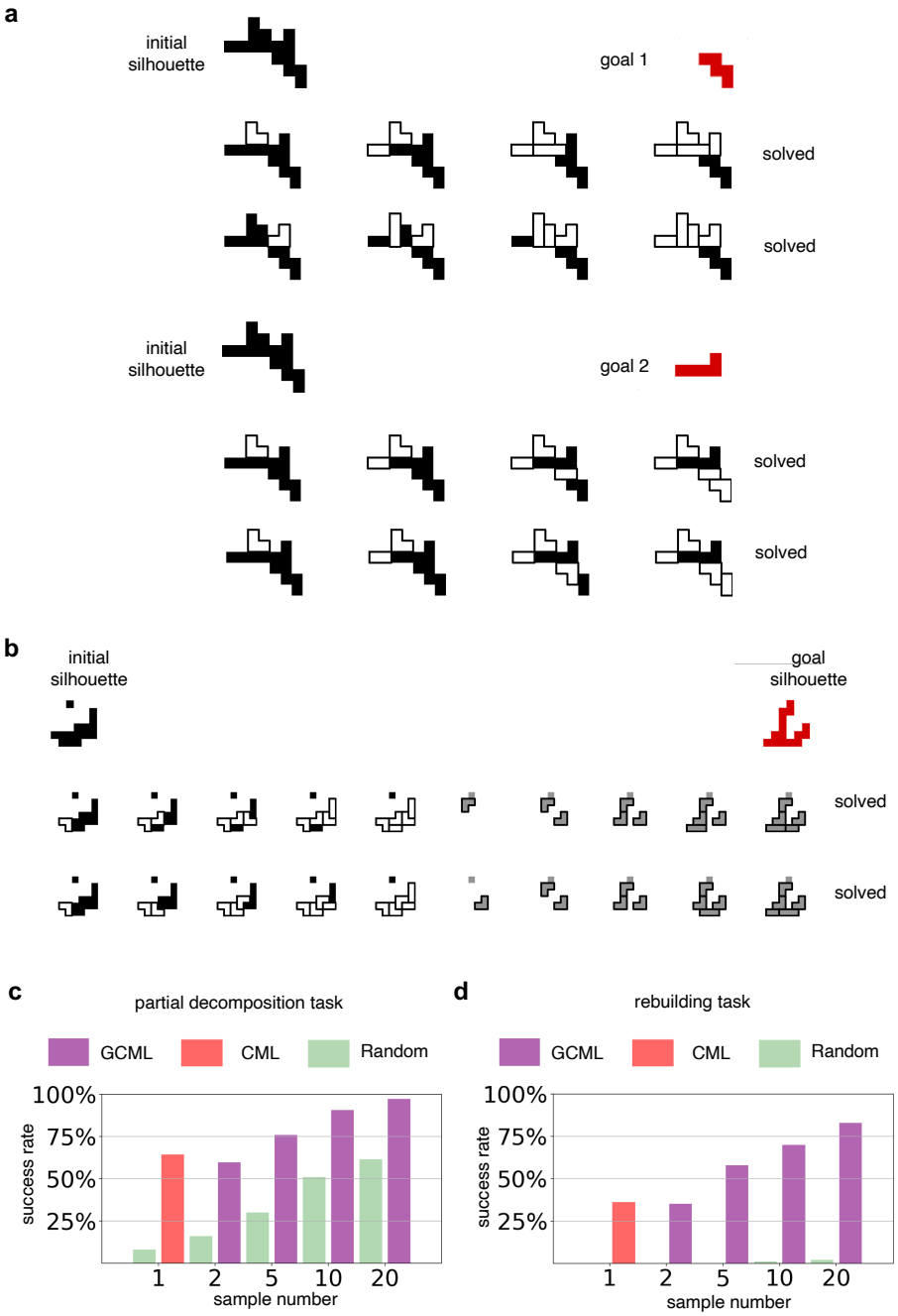

Fig. 5: Application of the GCML for generating possible solutions to more demanding versions of the silhouette decomposition task. (a) Decomposition to a specific non-empty silhouette (containing 2 BBs, as shown in red) from an initial silhouette that contains 6 BBs. Successful decompositions by the GCML are shown for two examples of start- and goal-silhouettes. (b) We tested a more demanding version of the silhouette decomposition task. After decomposition, the model must add BBs to form a goal silhouette, which, like the initial silhouette, contains 5 BBs. A random pixel was introduced to evaluate the robustness of our model. Added BBs are shown in gray for two example imagined paths to the goal generated by the GCML. (c, d) Evaluation of the performance of the GCML for both types of extensions of the silhouette decomposition task.

To sum up, this simulation illustrates that endowing the cognitive maps of the GCML with compositional structure permits addressing challenging compositional tasks and generating trajectories through never encountered states. The compositional nature of the maps, which reflects the relations between building blocks and their possible combinations, permits generalizing the goal-directed imagination of the GCML to novel silhouette decomposition problems.

3 Discussion

We have a remarkable capability to imagine plans to achieve open-ended goals, including novel goals that we never encountered before. However, we still do not know how the brain supports goal-directed imagination and how to design efficient algorithms to address it. In this study, we introduce a novel computational model – the Generative Cognitive Map Learner (GCML) – that provides a candidate solution to these challenges. The GCML is a probabilistic generative model that stochastically samples possible trajectories to reach both known and novel goals from a learned cognitive map, which provides a "sense of direction" towards the goals.

In three simulations, we show that the GCML supports goal-directed imagination in various tasks and reproduces empirical findings about "replay" activity that might index the same capability in the brain. The first simulation addresses a spatial navigation task requiring finding novel paths to goals in known 2D spatial maps, analogous to the rodent study of (Pfeiffer and Foster, 2013). Our results show that after creating a cognitive map of the 2D environment through experience, the GCML is capable to create novel goal-directed plans resembling those observed in the hippocampal replay, for any given start and goal location. Furthermore, it is capable to generalize to never experienced trajectories, as observed in rodents (Gupta et al., 2010). The second simulation addresses generic problem solving tasks consisting in finding short paths from start to goal nodes in a graph. Our results show that the GCML finds efficient heuristic solutions, approximating the well-known k-shortest path problem (Eppstein, 1998). Furthermore, the GCML can leverage noise to solve instances of the problem in which certain nodes are associated with rewards and penalties. Finally, when edges are removed from the graph, it is capable to stochastically sample trajectories that avoid them, without retraining the cognitive map – paralleling evidence of hippocampal replay trajectories around novel barriers (Widloski and Foster, 2022). Finally, the third simulation addresses a more challenging, compositional task, which requires deciding whether a novel 2D silhouette can be decomposed into a known set of building blocks. A challenge of this task is that it requires planning through never-encountered states in the cognitive map, including novel start and goal states. We show that the GCML model is capable to solve this computationally challenging (NP-hard) task using compositional cognitive maps and its intrinsic sense of direction – and to generate imagined sequences of removals of building blocks similar to those recorded in the human brain (Schwartenbeck et al., 2023).

Taken together, our simulations show that the GCML provides a surprisingly powerful and versatile heuristic method to solve computationally challenging problem solving tasks, with the help of learned cognitive maps and an algorithmic form of goal-directed imagination. Furthermore, the GCML provides a novel mechanistic hypotheses about how the brain could use cognitive maps to support goal-directed replay sequences towards known navigational goals (Pfeiffer and Foster, 2013), around novel barriers (Widloski and Foster, 2022), and towards novel goal configurations in compositonal tasks (Schwartenbeck et al., 2023).

Besides explaining previous findings, our perspective generates numerous novel empirical predictions that can be addressed in future studies. First, our perspective suggests that during goal-directed imagination, the brain predicts and represents all the intermediate states of the plan towards the goal. Importantly, while in our paper we focused on hippocampal sequences, various studies in humans, monkeys and rodents reported prospective representations of states in various cortical areas, such as the parietal and the prefrontal cortex, which might represent cognitive maps for abstract states (Dimakou et al., 2025; Mushiake et al., 2006; Saito et al., 2005; Schuck et al., 2016; Viganò et al., 2023). Future studies could align more directly the mechanisms of the GCML to the results of these and other studies.

A second prediction is that at every decision point during imagination, the brain represents the difference between the current and goal states. The assumption that goal-directed behavior is guided by (and aims to minimize) the difference between current and goal states is common to cybernetics (Powers, 1973; Rosenblueth et al., 1943; Wiener, 1961) and active inference (Parr et al., 2022). However, different from previous approaches, in the GCML the difference between current and goal states is used for action selection through a learned W matrix that estimates the utility of each action.

This leads to the third prediction; namely, that the difference between current and goal states is projected to (pre)motor areas, which have a dual role in both "virtual" and "actual" action selection. During goal-directed imagination, these neuronal populations are prevented from initiating the preferred action (possibly by downstream networks), but give rise to a competitive (winner-take-all, WTA) process, which selects the next imagined action. When action is disinhibited, the neural populations support the execution of the selected action (Cisek, 2007; Cisek and Kalaska, 2010; Pezzulo

and Cisek, 2016). If this perspective is correct, the variance of the imagined solution paths – or perhaps the level of fantasy and creativity – could be causally related to the level of noise in the WTA circuit for virtual action selection.

Fourth, our perspective predicts the speed of the generation of the next step of a multi-step solution plan to be largely invariant, both within the same task and across tasks (at least for well known tasks). The reason is that the duration of generation the next step in a plan is determined by the processing speed of a feedback circuit that routes activity from the implementation of a cognitive map for the task to motor areas and the implementation of the WTA mechanism for generating the next (imagined) action (see Figure S2 for a neural network implementation of the planning process).

Fifth, if the projection from motor areas where the imagined actions are selected to the structures (e.g., parietal cortex) where the task-relevant cognitive map is interrupted or neural activity in the cognitive map is interrupted, the imagination of a solution path is halted.

The model presented here has various limitations that need to be addressed in future studies. One limitation is that in our simulations, we provided the start and goal states as observations to the GCML. Addressing more general cases in which one or both are unknown is an open challenge. Another limitation is that in our simulations, the GCML only addresses a subset of the empirical findings related to hippocampal replay – in particular, those that appear more related to goal-directed planning and imagination (Pfeiffer and Foster, 2013; Widloski and Foster, 2022). Numerous experiments have shown that replay trajectories can be both forward and backward and show a different phenomenology in different conditions, such as when the animal is awake or sleeping, has recently collected a reward, is navigating a familiar or novel environment (Foster, 2017). It remains to be studied to what extent the GCML or its extensions can address the existing empirical findings about hippocampal replay. Similarly, future studies could assess to what extent the cognitive maps learned by the GCML resemble biological cognitive maps, such as those present in the hippocampal formation – and whether the GCML can help understand the diversity of cell types that compose the maps, such as place cells (O'Keefe, 1976), grid cells (Hafting et al., 2005), barrier cells (Rivard et al., 2004), object (Manns and Eichenbaum, 2009) or landmark vector cells (Deshmukh and Knierim, 2013), boundary vector cells (Lever et al., 2009; Solstad et al., 2008), and goal-vector cells (Sarel et al., 2017).

In addition to providing a novel hypothesis for goal-directed imagination in the brain, the GCML provides a novel approach to problem solving in brain-inspired AI. As exemplified by our simulations, an attractive feature of the GCML is that it is very flexible and can be used for creating imagined solutions for many different possible goals and start states. Importantly, it can address compositional tasks that are challenging for current AI methods and where the number of possible goal states to which paths can be imagined is much larger than the number of states that had been visited during learning of the underlying cognitive map. This inherent flexibility of the GCML provides a viable alternative to widely used AI methods for planning and problem solving, such as Monte-Carlo tree search (Gelly and Silver, 2011), which require a large number of rollouts. By the same token, it provides a viable alternative to reinforcement learning (RL) methods that typically require re-training and/or time-consuming calculations in order to create a plan for reaching a new goal (Sutton and Barto, 2018). Furthermore, we are not aware of studies showing generalization capabilities of RL approaches in compositional settings, where the number of possible goals is substantially larger than the total number of states visited during learning.

Finally, there are various potential applications of the GCML in neuromorphic computation and hardware. First, as already pointed out in (Stöckl et al., 2024), the shallow learning mechanism of the CML (and is extension GCML) makes this approach suitable for on-chip learning. Second, the action selection mechanism of the CML, that is also employed by the GCML for virtual action selection, is suitable for in-memory computing with memristor crossbars and can be carried out very fast, independently of the size of the environment and the number of options for action selection. This implies, that an in-memory computing implementation of the GCML supports very quick generation of an imagined solution path. Third, the GCML paves the way for creating edge devices that exhibit some first forms of goal-directed imagination or fantasy in order to find creative solutions for solving a hard problem. Future studies could explore the capabilities and limitations of the the GCML in challenging AI tasks and its implementation in neuromorphic hardware.

4 Methods

4.1 Mathematical description of the GCML

The starting point to develop the GCML is the CML methodology described in (Stöckl et al., 2024). Specifically, each observation $\mathbf{o}_t$ is mapped into a high-dimensional state space using a learned embedding matrix $\mathbf{Q} \in \mathbb{R}^{n_s \times n_o}$:

$$\mathbf{s}_t = \mathbf{Q}\mathbf{o}_t, \tag{2}$$

where n_s is the dimensionality of the state space, and n_o is the dimensionality of the observation $\mathbf{o}_t$. The target state (goal) is obtained as the embedding of a target observation:

$$\mathbf{s}^* = \mathbf{Q}\mathbf{o}^*. \tag{3}$$

Similarly, all possible actions $\mathbf{a}$ are embedded into the same high-dimensional space via a learned embedding matrix $\mathbf{V} \in \mathbb{R}^{n_s \times n_a}$, where n_a is the dimensionality of the action space. The objective of learning these embeddings is ensuring that the predicted next state $\hat{\mathbf{s}}_{t+1}$ is given by:

$$\hat{\mathbf{s}}_{t+1} = \mathbf{s}_t + \mathbf{V}\mathbf{a}_t. \tag{4}$$

$\hat{\mathbf{s}}_{t+1}$ should become a good approximation of the actual next state $\mathbf{s}_{t+1}$. The prediction error $(\mathbf{s}_{t+1} - \hat{\mathbf{s}}_{t+1})$, which is minimized by the local self-supervised learning rules for $\mathbf{Q}$ and $\mathbf{V}$ has therefore the following form:

$$\mathbf{\Delta V}_{t+1} = \eta_v \cdot (\mathbf{s}_{t+1} - \hat{\mathbf{s}}_{t+1})\mathbf{a}_t^{\mathrm{T}}, \tag{5}$$

$$\mathbf{\Delta Q}_{t+1} = \eta_q \cdot (\hat{\mathbf{s}}_{t+1} - \mathbf{s}_{t+1})\mathbf{o}_t^{\mathrm{T}}, \tag{6}$$

where η_v and η_q are the learning rates. These plasticity rules, known as Delta rules, approximate gradient descent to minimize the prediction error.

During trajectory generation, at step t, the delta vector represents the difference between the target state and the estimated current state:

$$\mathbf{\Delta}_t = \mathbf{s}^* - \hat{\mathbf{s}}_t, \tag{7}$$

where $\hat{\mathbf{s}}_t$ is the estimated current state, which equals the actual initial state $\mathbf{s}_0 = \mathbf{Q}\mathbf{o}_0$ when $t = 0$. The utility of an action $\mathbf{a}_t$ is computed as the dot product between its embedding $(\mathbf{V}\mathbf{a}_t)$ and the delta vector:

$$\mathbf{u}_t = \mathbf{V}^{\mathrm{T}}\mathbf{\Delta}_t. \tag{8}$$

Each dimension stores the utility of a specific action, forming a utility vector $\mathbf{u}_t$ of length n_a. The estimation of action utility is referred to as Principle 2 in CML (Stöckl et al., 2024).

In neuromorphic applications, obtaining a matrix transpose can be computationally demanding. To address this, we introduce a learnable matrix $\mathbf{W} \in \mathbb{R}^{n_a \times n_s}$ as a substitute for $\mathbf{V}^{\mathrm{T}}$. The utility calculation thus becomes:

$$\mathbf{u}_t = \mathbf{W}\mathbf{\Delta}_t. \tag{9}$$

The weights $\mathbf{W}$ can be learned via Hebbian learning:

$$\Delta\mathbf{W}_{t+1} = \eta_w \cdot \mathbf{a}_t(\mathbf{s}_{t+1} - \mathbf{s}_t)^{\mathrm{T}}, \tag{10}$$

where η_w is the learning rate. We consistently use the matrix $\mathbf{W}$ in all experiments throughout this paper.

In cases where not all actions are applicable, the utility values are masked by the learned affordance factor $\hat{\mathbf{g}}_t \in \mathbb{R}^{n_a}$, a one-hot vector that indicates which actions are executable (1 for executable, 0 otherwise). This gating factor prevents GCML from executing infeasible actions and is estimated using the affordance gating matrix $\mathbf{G} \in \mathbb{R}^{n_a \times n_s}$. The matrix is updated during the learning process according to the following rule:

$$\mathbf{\Delta G} = \eta_g \cdot (\mathbf{g}_t - \mathbf{G}\mathbf{s}_t)\mathbf{s}_t^{\mathrm{T}}, \tag{11}$$

where η_g represents the learning rate. The affordance information is either provided directly by the environment or learned through $\mathbf{G}$ during the imagination steps. The resulting vector, after applying the affordance gating, is termed the eligibility:

$$\mathbf{e}_t = \hat{\mathbf{g}}_t \odot (\mathbf{u}_t + \epsilon), \tag{12}$$

where ϵ is the inherent noise to encourage controlled exploration.

Finally, the selected action $\mathbf{a}_t$ is the one with the highest eligibility:

$$\mathbf{a}_t = \mathrm{WTA}(\mathbf{e}_t), \tag{13}$$

where $\mathrm{WTA}(\cdot)$ denotes the winner-take-all operator, selecting the action with the highest eligibility from all possible actions.

After selecting the action $\mathbf{a}_t$ at the estimated state $\hat{\mathbf{s}}_t$, the estimated next state for the following step, $t+1$, is calculated using the bootstrapping process:

$$\hat{\mathbf{s}}_{t+1} = \hat{\mathbf{s}}_t + \mathbf{V}\mathbf{a}_t. \tag{14}$$

By iteratively applying the trajectory generation process described earlier, a possible trajectory from $\mathbf{s}_0$ to $\mathbf{s}^*$ can be generated.

By repeatedly generating trajectories, and due to the inherent noise in Equ. 12, multiple trajectories are imagined. This results in a set of possible solutions from the initial state $\mathbf{s}_0$ to the goal $\mathbf{s}^*$.

One could select the final action sequence of a specific trajectory based on a given criterion. For example, when the environment contains some rewards associated with actions or observations, the estimated rewards $\hat{r}_t$ of each step are accumulated over the entire trajectory, and the sequential actions associated with the highest accumulated reward $\sum \hat{r}_t$ are selected as the final solution.

4.2 Details of Simulation 1

Paths were generated in a 30×30 2D grid, and movement was restricted to neighboring graph nodes with four possible actions. The impact of each action is initially unknown to the GCML.

The training dataset consisted of 600 randomly generated trajectories, each containing 16 steps of random exploration.

Due to limited exploration, some node transitions remained unvisited during training. In the planning phase, GCML navigates from a given starting node to a goal node by generating a set of possible trajectories, including transitions not encountered during training. To encourage controlled exploration, Gaussian noise is added to the utility values. The noise is sampled from the distribution $\epsilon \sim \rho \times \alpha_\epsilon \times \mathcal{N}(0,1)$, where ρ is dynamically set as the maximum utility value to maintain scale consistency across steps, and noise amplitude α_ϵ controls exploration intensity, with a default value of 0.6. Increasing α_ϵ encourages GCML to explore actions with lower utility values.

Actions $\mathbf{a}_t \in \mathbb{R}^4$ and observations $\mathbf{o}_t \in \mathbb{R}^{900}$ were represented using one-hot encoding. The dimension of the high-dimensional state space was set to 1,000, though smaller dimensions typically also perform adequately.

4.2.1 Details of the Implementation of GCML with Spiking Neurons

To align the imagination process of GCML more closely with biological data, an implementation using spiking neurons was developed. We applied this implementation specifically to the 2D path generation task of Figure 2b.

Details of the spiking neuron model.

A probabilistic spiking neuron model was used, with update equations for the relevant variables from time step t to $t+1$ given by:

$$\begin{cases} V_{t+1} &= g \times V_t + I + I_{\mathsf{ext}} \\ s_{t+1} &= \begin{cases} 1, \text{ if } V_{t+1} + \epsilon >= V_{\mathsf{threshold}} \text{ and } r_t \leq 0 \\ 0, \text{ otherwise.} \end{cases} \\ er_{t+1} &= \begin{cases} t_{\mathsf{rfr}}, & \text{if } S_{t+1} = 1 \\ r_t - 1, & \text{otherwise.} \end{cases} \end{cases} \tag{15}$$

Here, V_t and V_{t+1} represent the membrane potentials at time steps t and $t+1$, respectively, and $V_{\mathsf{threshold}} = 1$ is the neuron's firing threshold. I denotes the input current to the spiking neuron, and

r_t indicates the refractory state, determining whether the neuron is in a refractory period ($r_t > 0$) or not ($r_t \leq 0$). The parameter $t_{\mathsf{rfr}} = 4$ specifies the refractory period duration.

A spike is generated if the membrane potential V_{t+1}, with added random noise ϵ, reaches or exceeds the threshold $V_{\mathsf{threshold}}$, provided that the neuron is not in its refractory period. The spike indicator s_{t+1} is set to 1 if a spike occurs at time $t+1$ and 0 otherwise. After generating a spike, the membrane potential is reset to $V_{\mathsf{reset}} = 0$. The decay constant g is set to 0.01. The random noise ϵ is drawn from a uniform distribution $\mathcal{U}(0,1)$. The external current, set to $I_{\mathsf{ext}} = -0.7$, regulates the firing rate of spiking neurons.

For encoding with the spiking neuron network (SNN), we used rate coding based on the average spike count over a time interval $T_{snn} = 20$. A longer T_{snn} typically yields higher encoding precision.

To simplify notation, $\mathrm{SNN}(\mathbf{s}_t)$ denotes the entire process described in Eq. 15, where the input currents for n_s spiking neurons are defined by $\mathbf{I} = \mathbf{s}_t \in \mathbb{R}^{n_s}$. Thus, this formulation encodes the high-dimensional state $\mathbf{s}_t$ using spiking neurons. The function $\mathrm{FR}(\mathrm{SNN}(\mathbf{s}_t))$ denotes the rate-coded outputs of the n_s spiking neurons over the chosen time window $Tsnn$.

Details of training for the spike-based GCML. The training of spike GCML follows the same principles as the original GCML for the embedding matrices $\mathbf{Q}$ and $\mathbf{V}$, which were learned to minimize the prediction error of Principle I (see Equ. 5 and Equ. 6).

The weight matrix $\mathbf{W}$ is trained using a Hebbian learning rule, given by:

$$\Delta\mathbf{W} = \eta_w \cdot \mathbf{a}_t(\mathrm{SNN}(\mathbf{s}_{t+1}) - \mathrm{SNN}(\mathbf{s}_t))^{\mathrm{T}}, \tag{16}$$

where η_w is the learning rate.

After training, $\mathbf{Q}$ and $\mathbf{V}$ are scaled to ensure that all possible $\mathbf{s}_t = \mathbf{Q}\mathbf{o}_t$ values remain within the range [0,1], improving the encoding performance of the spike neurons. The utility value $\mathbf{u}_t$ is computed based on the rate coding:

$$\mathbf{u}_t = \mathrm{FR}(\mathbf{W}(\mathrm{SNN}(\mathbf{s}_{t+1}) - \mathrm{SNN}(\mathbf{s}_t))). \tag{17}$$

4.3 Details of Simulation 2

The setup consisted of a connected graph with 32 nodes, where the degree of each node was uniformly sampled between 2 and 5. Observations $\mathbf{o}_t$ corresponded to nodes, and actions $\mathbf{a}_t$ represented traversing edges in a specific direction, with each edge associated with two possible actions. Both $\mathbf{o}_t$ and $\mathbf{a}_t$ were encoded as unique one-hot vectors. Each node was assigned a reward drawn from a uniform distribution $\mathcal{U}(-5,5)$. The training dataset consisted of 200 randomly generated trajectories, each of length 32. During the planning process, GCML navigated from a given starting node to a target node, optimizing both path length and cumulative reward.

The mappings $\mathbf{G}$ transform the estimated high-dimensional state $\mathbf{s}_t$ into corresponding affordance factors. The affordance gating vector is a binary vector indicating available actions, where ones represent feasible actions and zeros indicate restricted actions. A linear transformation is sufficient in this setting, as $\mathbf{s}_t$ is approximately orthonormal.

Exploration was regulated by Gaussian noise $\epsilon \sim \rho \times \alpha_\epsilon \times \mathcal{N}(0,1)$, where ρ was set to the maximum utility value to ensure scale consistency, and the noise amplitude α_ϵ was set to a default value of 0.1.

Initial values for model parameters were drawn from Gaussian distributions: $\mathbf{Q} \sim \mathcal{N}(0,1)$ and $\mathbf{V}, \mathbf{W}, \mathbf{G} \sim \mathcal{N}(0, 0.1)$. Smaller initial values for $\mathbf{V}$ improved performance. Learning rates were set as $\eta_q = 0.1$ and $\eta_v = \eta_w = \eta_g = 0.01$. GCML demonstrates robustness to variations in learning ratesÄîsimilar to the original CMLÄîwith stable training performance for η_g in the range $[0.001, 0.05]$.

4.3.1 Details of Fig. 3c and d (top-k shortest paths).

To demonstrate that GCML provides a good approximation of the top-k shortest path problem, 40 trajectories were generated from a given starting node to a goal node under different noise scales using GCML. The number of distinct trajectories for each possible trajectory length (i.e., the number of edges in the path from the starting node to the goal node) was computed using a breadth-first search algorithm. For each trajectory length, the proportion of distinct trajectories generated by a given noise scale GCML relative to the total number of possible distinct trajectories serves as a measure of how well GCML approximates the top-k shortest path algorithm.

4.4 Details of Simulation 3

The silhouette decomposition problem involves finding a set of building blocks (BBs, or "tiles") that completely cover a given silhouette without gaps or overlaps. This process is called the decomposition of the given silhouette. GCML generates candidate decompositions through goal-directed sampling, similar to the previously described navigation tasks. In the simplest version of the problem, the goal node $\mathbf{o}^*$ is defined as the empty silhouette, while the starting node $\mathbf{o}_0$ is the silhouette to be decomposed. An action is defined as removing a BB from a particular position in the remaining silhouette. Details of this process are provided below.

Building Blocks. The training dataset consisted of eight predefined BBs. The first four BBs were categorized as *corners*, each with four possible orientations and a width of 1 pixel. The remaining four BBs were *bars*, including two longer types (1×3 and 3×1 pixels) and two shorter types (1×2 and 2×1 pixels). These eight BBs were derived from the human silhouette decomposition experiment of (Schwartenbeck et al., 2023), excluding the trivial 1×1 BB, which would excessively simplify the problem. GCML was tasked with imagining possible decompositions of given silhouettes, each composed exactly of five BBs.

Generation of Training and Test Datasets. Each silhouette in the dataset was created by randomly selecting five BBs and placing them within a 10×10 image at random positions. In total, 18,000 training samples and 2,000 test samples were generated. Any silhouette appearing in the training dataset was excluded from the test dataset, regardless of its position. Each sample represented a decomposition trajectory, in which BBs were sequentially removed from an initial silhouette. The silhouette generation process followed these rules: all BBs had to remain within image boundaries without overlapping. An initial BB was placed randomly within the image with equal probability for each position. Subsequent BBs were added adjacent to the existing silhouette, with at least one pixel adjacency to increase task difficulty. Longer adjacencies increased the number of possible removal actions, further challenging the model. In the training dataset, the five BBs composing each silhouette were removed sequentially in random order to generate training trajectories.

Details of Actions. In the silhouette decomposition problem, each action $\mathbf{a}_t$ corresponds to removing a BB from any valid position, resulting in 664 possible actions. Each action is uniquely encoded as a one-hot vector. This formulation aligns with human experiments (Schwartenbeck et al., 2023), where participants manipulate BBs using a mouse and keyboard to cover the silhouette. Since moving the same BB to different positions requires distinct movements, each BB at a different position is treated as a separate action.

Details of Observations. In the silhouette decomposition problem, the observation $\mathbf{o}_t \in \mathbb{R}^{10\times 10}$ directly represents the silhouette image, rather than being encoded as a one-hot vector as in previous experiments.

High-Dimensional Embedding Details. The state representation $\mathbf{s}_t = \mathbf{Q}\mathbf{o}_t$ is defined in a 100-dimensional space, matching the dimensionality of the observation space. The matrix $\mathbf{Q}$ is assumed to be the identity matrix $\mathbf{I}$, ensuring a direct mapping between the high-dimensional state and the observation. Each element in $\mathbf{s}_t$ functions as a grandmother neuron, activating only when the corresponding pixel in the observation space is black.

Affordance Details. The affordance gating mechanism in the silhouette decomposition problem comprises two components. The first component, $\mathtt{g}_1$, ensures the agent selects only BBs present in the silhouette. We assumed GCML could directly access $\mathtt{g}_1$ from the environment during all imagined steps. This assumption aligns with biological experiments (Schwartenbeck et al., 2023), where subjects directly observe the screen while solving the silhouette decomposition problem. Thus, affordance information is assumed available at all steps, requiring no inference from $\hat{\mathbf{s}}_t$.

The second component, $\mathtt{g}_2$, represents context-dependent affordance determined by the estimated state $\hat{\mathbf{s}}_t$. Each pixelÄôs affordance value is set as the number of non-empty adjacent pixels (top, bottom, left, and right). This is modeled as a convolution operation with the kernel:

$$\mathbf{k} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} \tag{18}$$

using a 3×3 kernel size, padding of 1, and stride of 1. The context-dependent affordance function g_2 transforms the delta vector $\mathbf{\Delta} = \mathbf{s}^* - \hat{\mathbf{s}}_t$ into:

$$\mathrm{g}_2(\mathbf{\Delta}) = (\mathbf{k} * (\mathbf{1} + \mathbf{\Delta})) \otimes (-\mathbf{\Delta}), \tag{19}$$

where $*$ denotes the convolution, and $\otimes$ represents the element-wise multiplication. The delta vector $\mathbf{\Delta}$ contains unremoved pixels with a value of -1 and all other pixels with a value of 0. The expression $(\mathbf{1} + \mathbf{\Delta})$ flips these values, converting -1 to 0 and 0 to 1, so the subsequent convolution counts the number of adjacent pixels with a value of 0. Finally, since only affordance values at unremoved pixels are relevant, an element-wise multiplication with the negative delta vector (where unsolved pixels have a value of 1) is performed.

Training details for GCML. The action embedding matrix $\mathbf{V}$ and the utility matrix $\mathbf{W}$ were trained according to Equ. 5 and 10, respectively. Both matrices were initialized using Gaussian distributions $\mathcal{N}(0, 0.001)$. The method remained effective across a broad range of initial standard deviations, from 0.0001 to 1. The learning rates were set to $\eta_v = \eta_w = 0.01$, with values between 0.0001 and 0.1 also demonstrating stable performance.

Trying to decompose a silhouette. This task was formulated as a navigation problem, where the embedding of the empty silhouette, $\mathbf{o}^* = \mathbf{0}$, represented the target node in the high-dimensional state space (cognitive map), while the embedding of the initial silhouette, $\mathbf{o}_0$, served as the starting node. In the presence of noise, GCML selected the action with the highest utility value from the set of currently feasible actions, determined by the environment's affordance factors $\mathrm{g}_1(\cdot)$ and $\mathrm{g}_2(\cdot)$. This selection process followed the directional intuition of the cognitive map:

$$\mathbf{a}_t = \mathrm{WTA}(\mathrm{g}_1(\mathbf{W}\mathrm{g}_2(\mathbf{\Delta})) + \epsilon), \tag{20}$$

where the noise term ϵ followed a Gaussian distribution $\mathcal{N}(0, 0.1)$.

After selecting an action $\mathbf{a}_t$, the next state was estimated as $\hat{\mathbf{s}}_{t+1} = \hat{\mathbf{s}}_t + \mathbf{V}\mathbf{a}_t$, and GCML proceeded to the next iteration to attempt the removal another BB. The algorithm terminated when no feasible actions remained. If all pixels were removed, GCML successfully identified a valid decomposition; otherwise, the attempt was deemed unsuccessful.

Details of performance evaluation. The performance of GCML was evaluated based on the proportion of test cases in which it successfully solves the given silhouette in test dataset. As baseline comparisons, we consider a random strategy that removes as many BBs as possible from the silhouette in a random manner, as well as the CML algorithm (Stöckl et al., 2024).

Decomposition to a given non-empty goal silhouette. In the non-empty goal silhouette decomposition problem, GCML removes BBs from a given silhouette $\mathbf{o}_0$ to transform it into various non-empty target silhouettes $\mathbf{o}^* \neq \mathbf{0}$. When generating the test dataset, the initial silhouettes contained six BBs, generated following the same method described earlier. Any initial silhouette that appeared in the training dataset was removed, regardless of its position. The goal silhouettes contained two BBs, generated by randomly removing four BBs from the six BBs used to generate the initial silhouettes. GCML could plan trajectories for non-empty goal silhouettes using Equ. 20 without requiring retraining. The performance was evaluated using 1,000 test silhouette pairs.

Combining sihouette decomposition and composition. To address the silhouette rebuilding problem, GCML began with a given silhouette and iteratively modified it by adding or removing BBs to reconstruct the goal silhouette. This task was solved using the same cognitive map learned from the silhouette decomposition problem introduced earlier, leveraging mathematical symmetry.

GCML followed a two-stage approach. In the first stage, it removed as many BBs as possible from the given silhouette using a straightforward method. Here, GCML adopted a sub-goal $\mathbf{s}^\dagger = \mathbf{0}$, and the delta vector was computed as $\mathbf{\Delta} = \mathbf{s}^\dagger - \mathbf{s}_t$. In the second stage, GCML reconstructed the target silhouette by sequentially adding BBs. In this phase, the goal was the final silhouette $\mathbf{s}^* = \mathbf{Q}\mathbf{o}^*$, and the delta vector was given by $\mathbf{\Delta} = \mathbf{s}^* - \mathbf{s}_t$. Since adding BBs was the inverse operation of removing them, GCML selected actions based on:

$$\mathbf{a}'_t = \mathrm{WTA}(\mathrm{g}'_1(\mathbf{W}\mathrm{g}'_2(\mathbf{\Delta})^{\mathrm{T}}) + \epsilon), \tag{21}$$

where the affordance factor $\mathbf{g}_1'$ ensured that GCML only added BBs contained within the goal $\mathbf{s}^*$. The function $\mathbf{g}_2'(\mathbf{\Delta}) = (\mathbf{k} * (\mathbf{1} - \mathbf{\Delta})) \otimes \mathbf{\Delta}$ was designed following the same principle as in Equ. 19. The negative transformation of $\mathbf{\Delta}$'s sign in Equ. 19 facilitated the addition of BBs as the inverse of their removal. Here, $\mathbf{a}_t'$ represented the action of adding a BB corresponding to $\mathbf{a}_t$, which removed a BB.

For the testing dataset, the initial and goal silhouettes were generated using the five BBs following the previously introduced generation process. To test the robustness of our model, a randomly selected noise-pixel was added to the same empty location in both the start and goal silhouettes. Any initial or goal silhouette that appears in the training dataset was removed, regardless of its position. A total of 1,000 initial and goal silhouette pairs were generated to test the performance.

References

Bakermans, J.J., J. Warren, J.C. Whittington, and T.E. Behrens. 2025. Constructing future behavior in the hippocampal formation through composition and replay. *Nature Neuroscience*: 1–12 .

Behrens, T.E., T.H. Muller, J.C. Whittington, S. Mark, A.B. Baram, K.L. Stachenfeld, and Z. Kurth-Nelson. 2018. What is a cognitive map? organizing knowledge for flexible behavior. *Neuron 100*(2): 490–509 .

Bottini, R. and C.F. Doeller. 2020. Knowledge across reference frames: Cognitive maps and image spaces. *Trends in Cognitive Sciences 24*(8): 606–619 .

Buesing, L., J. Bill, B. Nessler, and W. Maass. 2011. Neural dynamics as sampling: a model for stochastic computation in recurrent networks of spiking neurons. *PLoS computational biology 7*(11): e1002211 .

Bush, D., C. Barry, D. Manson, and N. Burgess. 2015. Using grid cells for navigation. *Neuron 87*(3): 507–520 .

Buzsáki, G. and E.I. Moser. 2013. Memory, navigation and theta rhythm in the hippocampal-entorhinal system. *Nature neuroscience 16*(2): 130–138 .

Cisek, P. 2007. Cortical mechanisms of action selection: the affordance competition hypothesis. *Philosophical Transactions of the Royal Society B: Biological Sciences 362*(1485): 1585–1599 .

Cisek, P. and J.F. Kalaska. 2010. Neural mechanisms for interacting with a world full of action choices. *Annual review of neuroscience 33*(1): 269–298 .

Davies, M., A. Wild, G. Orchard, Y. Sandamirskaya, G.A.F. Guerra, P. Joshi, P. Plank, and S.R. Risbud. 2021. Advancing neuromorphic computing with loihi: A survey of results and outlook. *Proceedings of the IEEE 109*(5): 911–934 .

Deshmukh, S.S. and J.J. Knierim. 2013. Influence of local objects on hippocampal representations: Landmark vectors and memory. *Hippocampus 23*(4): 253–267 .

Dimakou, A., G. Pezzulo, A. Zangrossi, and M. Corbetta. 2025. The predictive nature of spontaneous brain activity across scales and species. *Neuron* .

Eppstein, D. 1998. Finding the k shortest paths. *SIAM Journal on computing 28*(2): 652–673 .

Epstein, R.A., E.Z. Patai, J.B. Julian, and H.J. Spiers. 2017. The cognitive map in humans: spatial navigation and beyond. *Nature neuroscience 20*(11): 1504–1513 .

Foster, D.J. 2017. Replay comes of age. *Annual review of neuroscience 40*(1): 581–602 .

Geffner, H. and B. Bonet. 2013. *A concise introduction to models and methods for automated planning*. Morgan & Claypool Publishers.

Gelly, S. and D. Silver. 2011. Monte-carlo tree search and rapid action value estimation in computer go. *Artificial Intelligence 175*(11): 1856–1875 .

Gupta, A.S., M.A. Van Der Meer, D.S. Touretzky, and A.D. Redish. 2010. Hippocampal replay is not a simple function of experience. *Neuron 65*(5): 695–705 .

Habenschuss, S., Z. Jonke, and W. Maass. 2013. Stochastic computations in cortical microcircuit models. *PLoS computational biology 9*(11): e1003311 .

Hafting, T., M. Fyhn, S. Molden, M.B. Moser, and E.I. Moser. 2005. Microstructure of a spatial map in the entorhinal cortex. *Nature 436*(7052): 801–806 .

Hills, T.T., P.M. Todd, D. Lazer, A.D. Redish, and I.D. Couzin. 2015. Exploration versus exploitation in space, mind, and society. *Trends in cognitive sciences 19*(1): 46–54 .

Jonke, Z., S. Habenschuss, and W. Maass. 2016. Solving constraint satisfaction problems with networks of spiking neurons. *Frontiers in neuroscience* 10: 118 .

Kurth-Nelson, Z., T. Behrens, G. Wayne, K. Miller, L. Luettgau, R. Dolan, Y. Liu, and P. Schwartenbeck. 2023. Replay and compositional computation. *Neuron 111*(4): 454–469 .

Lansink, C.S., P.M. Goltstein, J.V. Lankelma, B.L. McNaughton, and C.M. Pennartz. 2009. Hippocampus leads ventral striatum in replay of place-reward information. *PLoS biology 7*(8): e1000173 .

Lever, C., S. Burton, A. Jeewajee, J. O'Keefe, and N. Burgess. 2009. Boundary vector cells in the subiculum of the hippocampal formation. *Journal of Neuroscience 29*(31): 9771–9777 .

Liu, C., R. Todorova, W. Tang, A. Oliva, and A. Fernandez-Ruiz. 2023. Associative and predictive hippocampal codes support memory-guided behaviors. *Science 382*(6668): eadi8237 .

Manns, J.R. and H. Eichenbaum. 2009. A cognitive map for object memory in the hippocampus. *Learning & memory 16*(10): 616–624 .

Moore, C. and J.M. Robson. 2001. Hard tiling problems with simple tiles. *Discrete & Computational Geometry* 26: 573–590 .

Moser, E.I., E. Kropff, and M.B. Moser. 2008. Place cells, grid cells, and the brain's spatial representation system. *Annu. Rev. Neurosci. 31*(1): 69–89 .

Mushiake, H., N. Saito, K. Sakamoto, Y. Itoyama, and J. Tanji. 2006. Activity in the lateral prefrontal cortex reflects multiple steps of future events in action plans. *Neuron 50*(4): 631–641 .

Newell, A., H.A. Simon, et al. 1972. *Human problem solving*, Volume 104. Prentice-hall Englewood Cliffs, NJ.

O'Keefe, J. 1976. Place units in the hippocampus of the freely moving rat. *Experimental neurology 51*(1): 78–109 .

O'Keefe, J. and L. Nadel. 1978. *The hippocampus as a cognitive map*. Oxford University Press, New York.

Parr, T., G. Pezzulo, and K.J. Friston. 2022. *Active inference: the free energy principle in mind, brain, and behavior*. MIT Press.

Pennartz, C., R. Ito, P. Verschure, F. Battaglia, and T. Robbins. 2011. The hippocampal–striatal axis in learning, prediction and goal-directed behavior. *Trends in neurosciences 34*(10): 548–559 .

Pezzulo, G. and P. Cisek. 2016. Navigating the affordance landscape: feedback control as a process model of behavior and cognition. *Trends in cognitive sciences 20*(6): 414–424 .

Pezzulo, G., M.A. Van der Meer, C.S. Lansink, and C.M. Pennartz. 2014. Internally generated sequences in learning and executing goal-directed behavior. *Trends in cognitive sciences 18*(12): 647–657 .

Pfeiffer, B.E. and D.J. Foster. 2013. Hippocampal place-cell sequences depict future paths to remembered goals. *Nature 497*(7447): 74–79 .

Poggio, T. 2012. The levels of understanding framework, revised. *Perception 41*(9): 1017–1023 .

Powers, W.T. 1973. *Behavior: The control of perception.* Aldine Chicago.

Raju, R.V., J.S. Guntupalli, G. Zhou, C. Wendelken, M. Lázaro-Gredilla, and D. George. 2024. Space is a latent sequence: A theory of the hippocampus. *Science Advances 10*(31): eadm8470 .

Redish, A.D. 2016. Vicarious trial and error. *Nature Reviews Neuroscience 17*(3): 147–159 .

Rivard, B., Y. Li, P.P. Lenck-Santini, B. Poucet, and R.U. Muller. 2004. Representation of objects in space by two classes of hippocampal pyramidal cells. *Journal of General Physiology 124*(1): 9–25 .

Rosenblueth, A., N. Wiener, and J. Bigelow. 1943. Behavior, purpose and teleology. *Philosophy of science 10*(1): 18–24 .

Russell, S.J. and P. Norvig. 2016. *Artificial intelligence: a modern approach.* pearson.

Saito, N., H. Mushiake, K. Sakamoto, Y. Itoyama, and J. Tanji. 2005. Representation of immediate and final behavioral goals in the monkey prefrontal cortex during an instructed delay period. *Cerebral Cortex 15*(10): 1535–1546 .

Sarel, A., A. Finkelstein, L. Las, and N. Ulanovsky. 2017. Vectorial representation of spatial goals in the hippocampus of bats. *Science 355*(6321): 176–180 .

Schuck, N.W., M.B. Cai, R.C. Wilson, and Y. Niv. 2016. Human orbitofrontal cortex represents a cognitive map of state space. *Neuron 91*(6): 1402–1412 .

Schwartenbeck, P., A. Baram, Y. Liu, S. Mark, T. Muller, R. Dolan, M. Botvinick, Z. Kurth-Nelson, and T. Behrens. 2023. Generative replay underlies compositional inference in the hippocampal-prefrontal circuit. *Cell 186*(22): 4885–4897 .

Solstad, T., C.N. Boccara, E. Kropff, M.B. Moser, and E.I. Moser. 2008. Representation of geometric borders in the entorhinal cortex. *Science 322*(5909): 1865–1868 .

Stöckl, C., Y. Yang, and W. Maass. 2024. Local prediction-learning in high-dimensional spaces enables neural networks to plan. *Nature Communications 15*(1): 2344 .

Stoianov, I., D. Maisto, and G. Pezzulo. 2022. The hippocampal formation as a hierarchical generative model supporting generative replay and continual learning. *Progress in Neurobiology* 217: 102329 .

Sutton, R.S. 1992. Adapting bias by gradient descent: An incremental version of delta-bar-delta. In *AAAI*, pp. 171–176. Citeseer.

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

Tavares, R.M., A. Mendelsohn, Y. Grossman, C.H. Williams, M. Shapiro, Y. Trope, and D. Schiller. 2015. A map for social navigation in the human brain. *Neuron 87*(1): 231–243 .

Tsoar, A., R. Nathan, Y. Bartan, A. Vyssotski, G. Dell'Omo, and N. Ulanovsky. 2011. Large-scale navigational map in a mammal. *Proceedings of the National Academy of Sciences 108*(37): E718–E724 .

Ujfalussy, B.B. and G. Orbán. 2022. Sampling motion trajectories during hippocampal theta sequences. *Elife* 11: e74058 .

Van Der Meer, M., Z. Kurth-Nelson, and A.D. Redish. 2012. Information processing in decision-making systems. *The Neuroscientist 18*(4): 342–359 .

Viganò, S., R. Bayramova, C.F. Doeller, and R. Bottini. 2023. Mental search of concepts is supported by egocentric vector representations and restructured grid maps. *Nature Communications 14*(1): 8132 .

Widloski, J. and D.J. Foster. 2022. Flexible rerouting of hippocampal replay sequences around changing barriers in the absence of global place field remapping. *Neuron 110*(9): 1547–1558 .

Wiener, N. 1961. *Cybernetics or Control and Communication in the Animal and the Machine*, Volume 25. MIT press.

Wittkuhn, L., S. Chien, S. Hall-McMaster, and N.W. Schuck. 2021. Replay in minds and machines. *Neuroscience & Biobehavioral Reviews* 129: 367–388 .

Xiao, Z., X. Wang, J. Zhang, J. Ou, L. He, Y. Qu, X. Hu, T. Behrens, and Y. Liu. 2025. Human hippocampal ripples predict the alignment of experience to a grid-like schema. *bioRxiv*: 2025–01 .

Supplementary Information for

Neural sampling from cognitive maps produces goal-directed imagination

A Possible implementation of GCMLs by neural networks

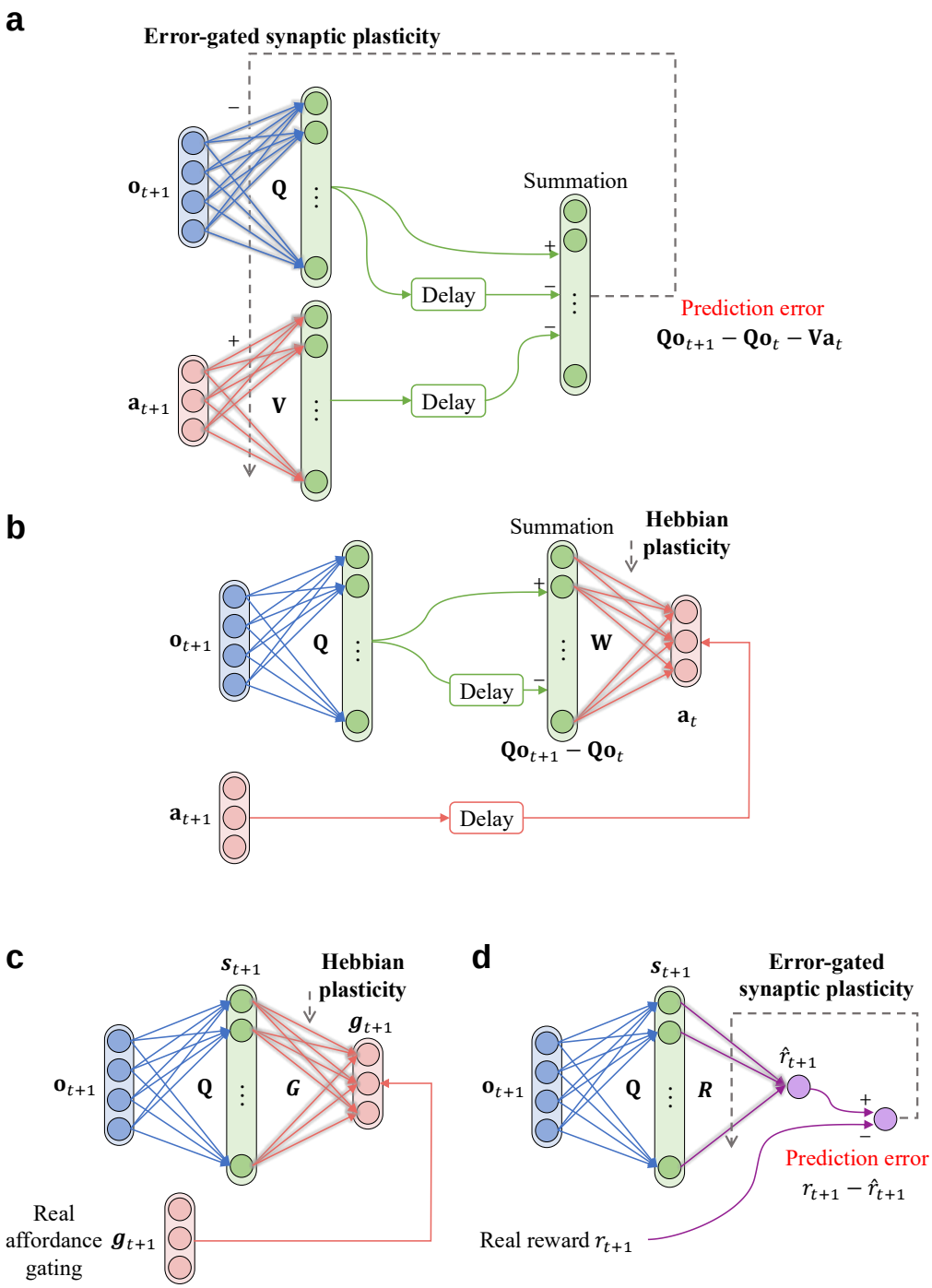

Fig. S1: **Neural network implementation of the GCML's training process.** **a** Network structure for cognitive map learning. The prediction error, computed by the population of linear units on the right, gates synaptic plasticity between neuron populations representing observations, actions, and internal states of GCML. **b** Network structure for learning the utility weight matrix $\mathbf{W}$ using Hebbian plasticity. Some signals pass through inhibitory interneurons (not shown), indicated by a negative sign at the corresponding synaptic connections. These signals are assumed to be delayed by one time step. **c** Network structure for learning the affordance gating matrix $\mathbf{G}$ based on the real affordance vector $\mathbf{g}_{t+1}$, applying a Hebbian synaptic update rule. **d** Network structure for learning the reward matrix $\mathbf{R}$ using the reward value at observation $\mathbf{o}_{t+1}$. The prediction error update rule is applied. The network associated with the reward estimation is served as an outside observer.

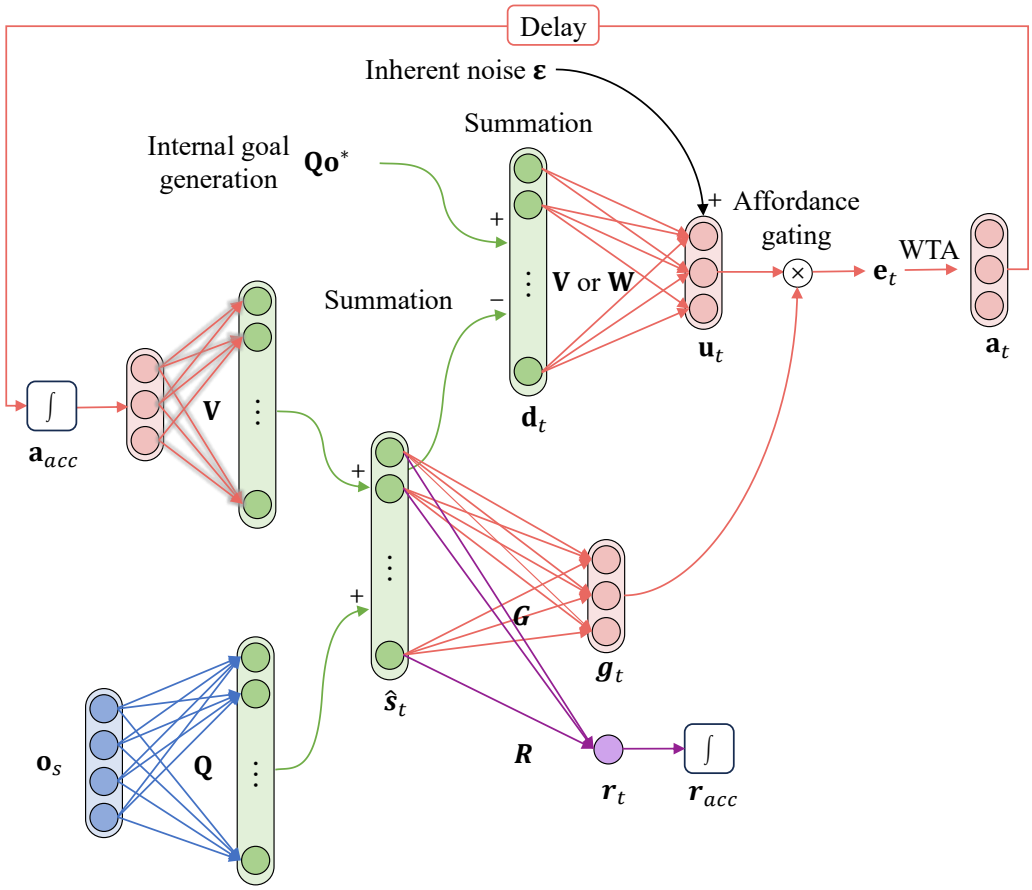

Fig. S2: **Neural network implementation of the GCML's planning process.** Given a target observation $\mathbf{Qo}^*$, GCML plans a trajectory starting from observation $\mathbf{o}_s$. The difference between the target and the estimated current state is computed as $\mathbf{d}_t = \mathbf{Qo}^* - \hat{\mathbf{s}}_t$, where $\hat{\mathbf{s}}_t = \mathbf{Va}_{acc} + \mathbf{Qo}_s$. Here, $\mathbf{a}_{acc}$ represents the accumulated actions from $\mathbf{a}_0$ to $\mathbf{a}_t$. At each step, the utility values for all possible actions are computed as $\mathbf{u}_t = \mathbf{Wd}_t + \epsilon$, where ϵ represents inherent noise. The affordance gating vector $\mathbf{g}_t$, provided by the environment at the first step, is computed as $\mathbf{G}\hat{\mathbf{s}}_t$. The eligibility of each action is then determined by gating the utility with the affordance via elementwise multiplication $e_t = \mathbf{g}_t \otimes \mathbf{u}_t$. The action $\mathbf{a}_t$ (a one-hot vector) is selected using a winner-take-all mechanism, $\mathrm{WTA}(\mathbf{e}_t)$. Once selected, $\mathbf{a}_t$ is accumulated into $\mathbf{a}_{acc}$ with a time delay to prepare for the next planning step. The reward at each step is computed as $r_t = \mathbf{R}\hat{s}_t$, and the total reward is accumulated as $r_{acc} = \sum_t r_t$.

GCML provides a framework for online planning and problem-solving by leveraging neural sampling from cognitive maps, structured according to the conceptual and algorithmic levels of Marr hierarchy (Poggio, 2012). This framework can be implemented using biological or neuromorphic neural networks, with one such implementation is depicted in Fig. S1 and Fig. S2.

The training process of GCML involves several populations of neurons. Peripheral neurons encode observations (blue in Fig. S1), while action-encoding neurons use a one-hot representation (red in Fig. S1). Affordance gating neurons (also red) regulate action feasibility. Reward neurons (purple) represent value signals. High-dimensional state-space neurons (green) encode and process state information.

To facilitate temporal dynamics, the network incorporates delay modules that introduce one-time-step latencies through inhibitory relays. Such delay modules are readily available in neuromorphic hardware platforms, including Spinnaker and Intel's Loihi chip (Davies et al., 2021), as well as in-memory computing systems.

As illustrated in Fig. S1, GCML learns the parameters $\mathbf{Q}$, $\mathbf{V}$, and $\mathbf{G}$ through minimizing the prediction error, while $\mathbf{W}$ and $\mathbf{R}$ are updated via Hebbian plasticity.

The planning process, shown in Fig. S2, begins when a target observation $\mathbf{o}^*$ is specified. Initially, GCML starts from an observation $\mathbf{o}_s$, and computes the difference between the target and the current state as $\mathbf{d}_t = \mathbf{s}^* - \hat{\mathbf{s}}_t$. The estimated state at time t is given by $\hat{\mathbf{s}}_t = \mathbf{V}\mathbf{a}_{acc} + \mathbf{Q}\mathbf{o}_s$, where $\mathbf{a}_{acc}$ represents the accumulated sequence of actions from $\mathbf{a}_0$ to $\mathbf{a}_t$, which is in consistent with the bootstrapping process.

At each step, the utilities of all possible actions are computed as $\mathbf{u}_t = \mathbf{W}\mathbf{d}_t + \epsilon$, where ϵ accounts for inherent noise. The affordance gating vector $\mathbf{g}_t$ is obtained from the environment at the first step and subsequently estimated as $\mathbf{G}\hat{\mathbf{s}}_t$. The eligibility of each action is determined by the element-wise product $\mathbf{e}_t = \mathbf{g}_t \otimes \mathbf{u}_t$. The action $\mathbf{a}_t$, represented as a one-hot vector, is then selected using a winner-take-all mechanism: $\mathbf{a}_t = \mathrm{WTA}(\mathbf{e}_\mathrm{t})$.

Once an action is selected, it is accumulated into $\mathbf{a}_{acc}$ with a temporal delay, preparing for the next step in the planning process. The reward at each step is computed as $r_t = \mathbf{R}\hat{\mathbf{s}}_t$, and the total reward is accumulated as $r_{acc} = \sum_t r_t$. The accumulated reward serves as the criterion for the outside observer to select the optimal trajectory from the set of generated trajectories.

B Algorithm details

Since GCML's planning process is complex, we provide detailed implementation example algorithms in three different environments to enhance understanding. Algorithm 1 outlines the implementation of trajectory generation in the 2D navigation task (Simulation), Algorithm 2 describes the implementation of the reward-based graph task (Simulation 2), and Algorithm 3 presents the approach for the silhouette decomposition task (Simulation 3).

Algorithm 1 Generating Trajectories in a 2D Navigation Task (Simulation 1)

Require: Initial observation $\mathbf{o}_0$, goal observation $\mathbf{o}^*$, number of trajectories n_θ, trajectory length n_{traj}, trained CML

```
Compute target state: s* ← Qo*
for θ = 0 to n_θ − 1 do
    Initialize current state: ŝ_t ← Qo_0
    for t = 0 to n_traj − 1 do
        if ŝ_t ≈ s* then
            break
        end if
        if t = 0 then
            Set affordance vector ĝ using real affordance from o_0
        else
            Estimate affordance vector: ĝ ← Gô_t
        end if
        Compute delta vector: Δ ← s* − ŝ_t
        Compute utility with noise: u ← WΔ + ε
        Compute eligibility: e ← ĝ ⊗ u
        Select next action: a_t ← WTA(e)
        Predict next state: ŝ_{t+1} ← ŝ_t + Va_t
    end for
end for
return a set of generated trajectories
```

Algorithm 2 Generating Trajectories in the Reward-Based Graph Task ((Simulation 2))

Require: Initial observation $\mathbf{o}_0$, target observation $\mathbf{o}^*$, number of trajectories n_θ, trajectory length n_{traj}, trained GCML

Compute target state: $\mathbf{s}^* \leftarrow \mathbf{Q}\mathbf{o}^*$
for $\theta = 0$ to $n_\theta - 1$ **do**
  Initialize total reward: $r_\theta \leftarrow 0$
  Initialize current state: $\hat{\mathbf{s}}_t \leftarrow \mathbf{Q}\mathbf{o}_0$
  for $t = 0$ to $n_{traj} - 1$ **do**
    if $\hat{\mathbf{s}}_t \approx \mathbf{s}^*$ **then**
      break
    end if
    if $t = 0$ **then**
      Set affordance factor $\hat{\mathbf{g}}$ using real affordance from $\mathbf{o}_0$
    else
      Estimate affordance vector: $\hat{\mathbf{g}} \leftarrow \mathbf{G}\hat{\mathbf{s}}_t$
    end if
    Compute delta vector: $\mathbf{\Delta} \leftarrow \mathbf{s}^* - \hat{\mathbf{s}}_t$
    Compute utility with noise: $\mathbf{u} \leftarrow \mathbf{W}\mathbf{\Delta} + \epsilon$
    Compute affordance factor: $\hat{\mathbf{g}}_t = \mathbf{G}\hat{\mathbf{s}}_t$
    Compute eligibility: $\mathbf{e} \leftarrow \hat{\mathbf{g}}_t \otimes \mathbf{u}$
    Select next action: $\mathbf{a}_t \leftarrow \mathrm{WTA}(\mathbf{e})$
    Predict next state: $\hat{\mathbf{s}}_{t+1} \leftarrow \hat{\mathbf{s}}_t + \mathbf{V}\mathbf{a}_t$
    Compute reward: $r \leftarrow \mathbf{R}\hat{\mathbf{s}}_{t+1}$
    Accumulate trajectory reward: $r_\theta \leftarrow r_\theta + r$
  end for
end for
return a set of trajectories and the accumulated reward for each trajectory.

Algorithm 3 Generating a Set of Possible Decompositions in the Silhouette Task (Simulation 3)

Require: Initial silhouette $\mathbf{o}_0$, target observation $\mathbf{o}^* = \mathbf{0}$, maximum trajectory length n_{traj}, trained cognitive map

Initialize state: $\hat{\mathbf{s}}_0 \leftarrow \mathbf{Q}\mathbf{o}_0$
for $t = 0$ to $n_{traj} - 1$ **do**
  if $\hat{\mathbf{s}}_t \approx \mathbf{s}^*$ **then**
    Found a valid decomposition for $\mathbf{o}_0$
    break
  end if
  Compute delta vector: $\mathbf{\Delta} \leftarrow \mathbf{Q}\mathbf{o}^* - \hat{\mathbf{s}}_t$
  Apply context-dependent affordance factor g_2 (see Equ. 19):
    $\mathrm{g}_2(\mathbf{\Delta}) = (\mathbf{k} * (\mathbf{1} + \mathbf{\Delta})) \otimes (-\mathbf{\Delta})$
  if no valid action remains **then**
    No valid decomposition found.
    break
  end if
  Compute eligibility with noise: $\mathbf{e}_t \leftarrow \mathrm{g}_1(\mathbf{W}\mathrm{g}_2(\mathbf{\Delta})) + \epsilon$ (see Equ. 20)
  Select action: $\mathbf{a}_t \leftarrow \mathrm{WTA}(\mathbf{e}_t)$
  Compute next state: $\hat{\mathbf{s}}_{t+1} = \hat{\mathbf{s}}_t + \mathbf{V}\mathbf{a}_t$
end for
return a set of trajectories

C Supplementary figures for Simulation 1

Fig. S3 illustrates the progressive unfolding of the cognitive map during learning. Principal Component Analysis (PCA) was used to project the high-dimensional representation of $\mathbf{Q}\mathbf{o}$ for all possible $\mathbf{o}$

onto a two-dimensional space. Initially, before training, the nodes were randomly distributed in the high-dimensional space. As training progresses, they gradually organized into a structured 30×30 two-dimensional mesh.

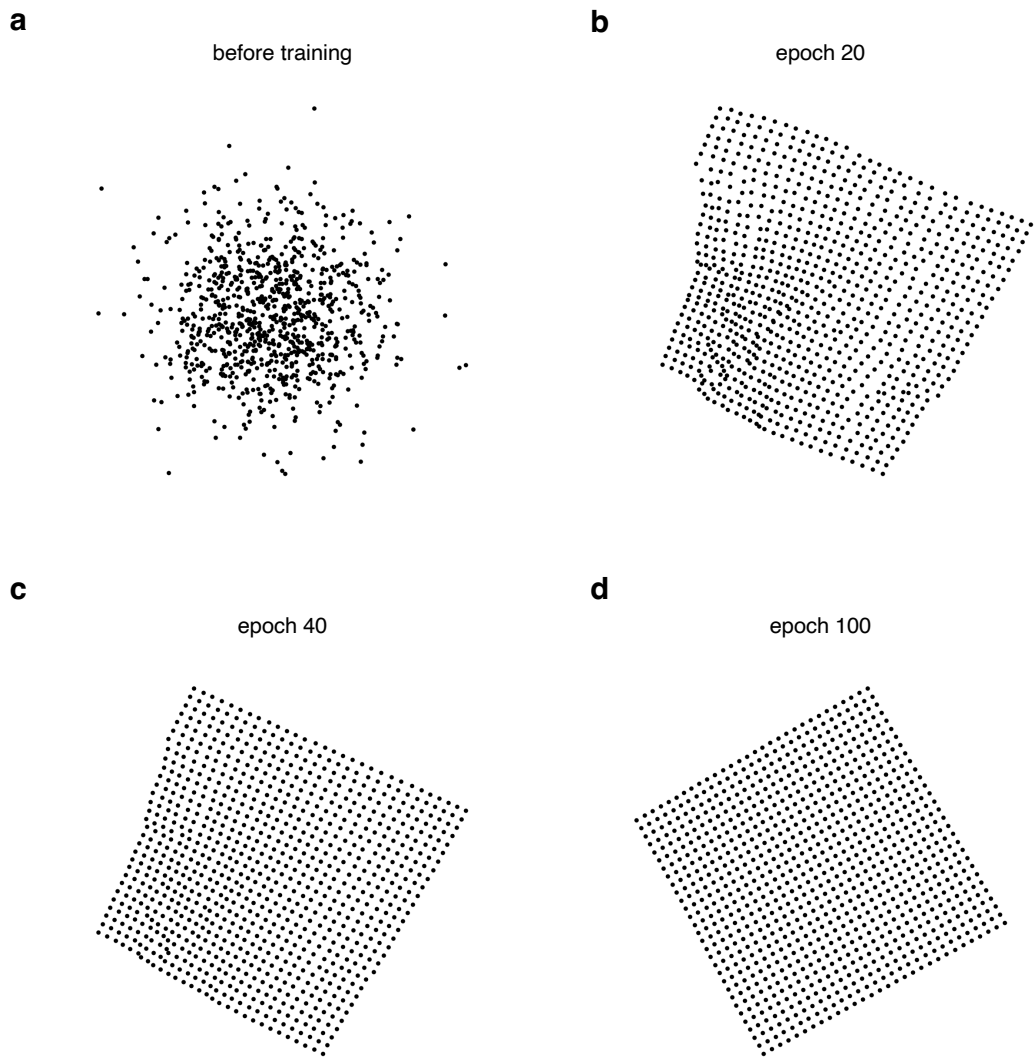

Fig. S3: **Unfolding of the cognitive map during learning of Fig.2.** (a) Cognitive map before training. The high-dimensional states of **Qo** were distributed in an unordered manner within the high-dimensional space. (b-d) Cognitive map during training at epochs 20, 40, and 100, respectively. The structure gradually organized into a 30×30 grid within the high-dimensional space.

D Supplementary figures for Simulation 2

Fig. S4 shows the top five shortest paths generated by GCML with a noise scale of 0.15. These five trajectories represent all possible shortest paths of length five between the given start and goal nodes, demonstrating that GCML provides a good approximation of the top-k shortest path algorithm.

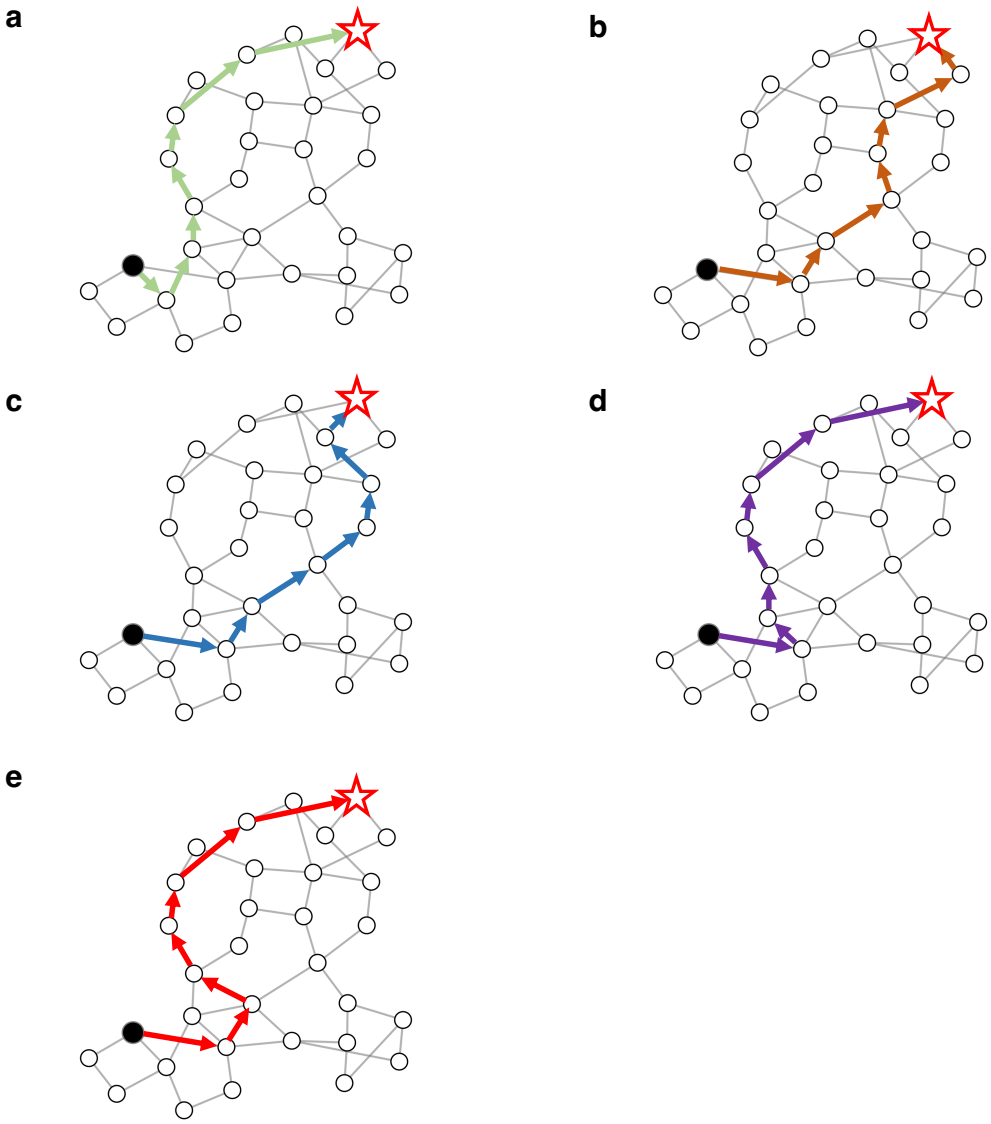

Fig. S4: **Top five shortest trajectories generated by GCML in the abstract graph environment.** These five trajectories are the shortest possible paths of length five between the given start (shown in black dot) and goal nodes (shown in red star).

D.1 Foresight ability of GCML

By generating a set of possible trajectories, GCML can adapt to environmental changes without retraining the cognitive map. For instance, in Fig. S5, two red edges were removed during the planning process to mimic the environmental changes. Without retraining the cognitive map, the original CML algorithm would lead to a detour, as shown by the blue trajectories. In contrast, the GCML generates a set of candidate trajectories and chose one that successfully avoids the detour, as demonstrated by the green trajectory.

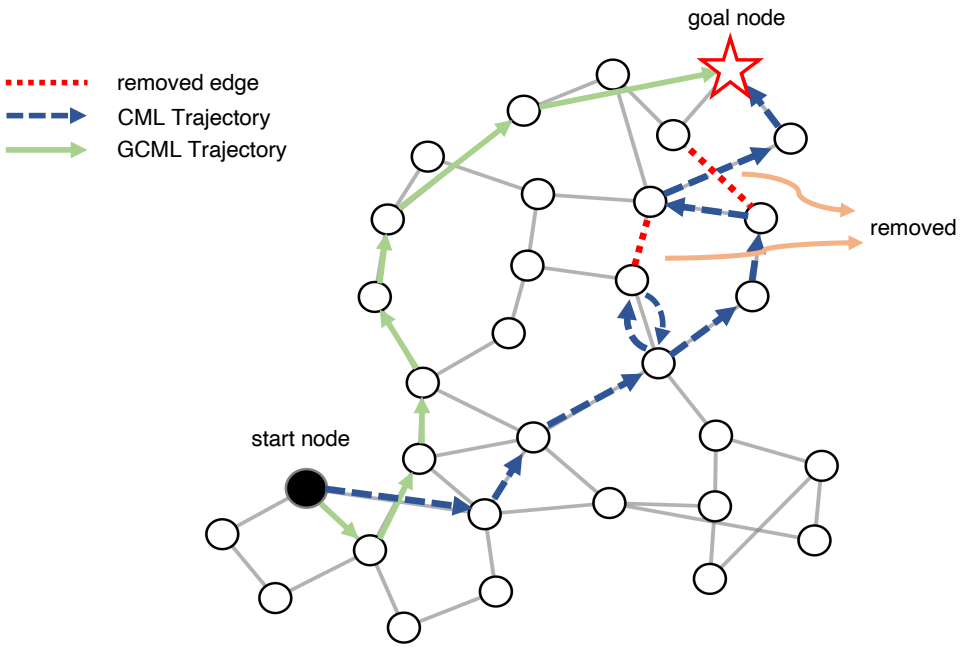

Fig. S5: **GCML demonstrates foresight in handling environmental changes.** The red dashed edges in the cognitive map represent connections removed during planning but retained during training. Without retraining the cognitive map, CML would take a detour, as shown by the blue arrow trajectory. In contrast, GCML avoids this detour by selecting an alternative shortest path, indicated by the green arrow trajectory.

D.2 Experimental understanding of the relationship between the top-K shortest path and GCML algorithm

In the abstract graph environment, GCML is asked to generate a set of possible trajectories from a given source node $\mathbf{o}_1$ to the goal node $\mathbf{o}^*$. Because trajectories are generated by adding noise to the utility $u(\mathbf{a})$ of a specific action $\mathbf{a}$. So the key point to understand why GCML is a good approximation of K-shortest path algorithm is the relationship between the shortest path length associated with action $\mathbf{a}$ and its utility value.

Consider a shortest path from $\mathbf{s}_t$ to $\mathbf{s}^*$ consisting of L actions. If the agent indeed reaches the goal from the current state in exactly L steps, then in the embedding space, after the prediction error is minimized to zero, satisfies:

$$\mathbf{\Delta} = \sum_{i=1}^{L} \mathbf{V}\,\mathbf{a}_i, \tag{22}$$

where $\mathbf{a}_i$ is the ith action taken from a given starting node.

We define the *cosine utility* of an action $\mathbf{a}$ (embedded as $\mathbf{Va}$) at the current state $\mathbf{s}_t$ as $\mathbf{u}_{cos}(\mathbf{a})$, which is computed by the cosine similarity

$$\mathbf{u}_{cos}(\mathbf{a}) = \frac{\mathbf{\Delta}^{\mathrm{T}}\,(\mathbf{V}\,\mathbf{a})}{\|\mathbf{\Delta}\|_2 \times \|\mathbf{Va}\|_2} = \frac{(\mathbf{s}^* - \mathbf{s}_t)^{\mathrm{T}}\,(\mathbf{V}\,\mathbf{a})}{\|\mathbf{s}^* - \mathbf{s}_t\|_2 \times \|\mathbf{Va}\|_2}. \tag{23}$$

This cosine utility normalizes our previously defined utilities to the range $[-1, 1]$, providing a more intuitive interpretation of its value. Due to high-dimensional properties, the norms of action embeddings remain approximately equal, a property largely preserved after learning (see (Stöckl et al., 2024)). Thus, normalization does not alter the relative ranks of action utilities.

We analyzed three representative graph structures—random graphs (Fig. S6a), small-world graphs (Fig. S6b), and dead-end graphs (Fig. S6c)—by examining the relationship between the first action's

embedding $\mathbf{Va}_1$ and the shortest path length L. The utility values for each graph are shown in the bottom panels of Fig. S6a–c.

For instance, in Fig. S6b, actions $(\mathbf{a})_0$ and $(\mathbf{a})_1$, originating from the initial node, both have a shortest path length of 5 to the goal node, resulting in similar utility values. However, action $(\mathbf{a})_2$, with a shortest path length of 7, shows significantly lower utility compared to actions $(\mathbf{a})_0$ and $(\mathbf{a})_1$. This pattern is also observed in both Fig. S6a and c. In Fig. S6a, although both actions $(\mathbf{a})_0$ and $(\mathbf{a})_2$ have a shortest path length of 7, the utility of $(\mathbf{a})_0$ is slightly lower than that of $(\mathbf{a})_2$. We suspect that this is because action $(\mathbf{a})_0$ leads to a single shortest path to the goal, whereas action $(\mathbf{a})_2$ offers four possible shortest paths, as illustrated in Fig. S4a. In Fig. S4c, both actions $(\mathbf{a})_1$ and $(\mathbf{a})_2$ have the shortest path length of 5, resulting in similar cosine utilities relative to the delta vector toward the goal. In contrast, the other two actions, which lead to dead ends, have cosine utilities close to zero or negative.

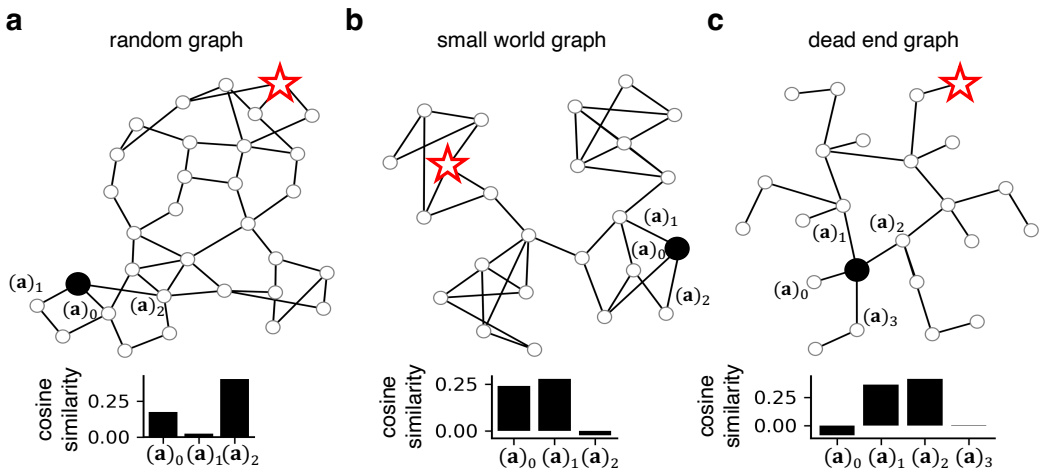

Fig. S6: **Relationship between utility value and shortest path length.** (a-c) The top panels show three different graph types: random graph, small-world graph, and dead-end graph. The bottom panels of (a-c) display the utility values (i.e., cosine similarity) of the first actions corresponding to the source node.

In summary, when multiple shortest paths both have length L, the utility of their first-step actions remains similar. However, actions leading to longer paths result in lower utility, making them less likely to be selected in a greedy decision-making process focused on maximizing utility. Therefore, if we allow noise in each action's utility and sample multiple independent trajectories (each time re-drawing noise), then sometimes the agent will deviate from the single highest-utility route. As a result:

- **Shortest path(s):** They dominate because at each step their corresponding action typically has the highest true utility.
- **Near-shortest paths:** They still have a non-trivial chance of being selected occasionally due to noise, so repeated sampling will discover them with non-zero probability.
- **Far-suboptimal paths:** With significantly lower stepwise utility, they are less likely to appear in repeated trials.

By collecting all distinct sampled trajectories and ranking them by actual cost (or negative total utility), this procedure recovers the top-K solutions—those that are at most the K-th best path. Thus, the noisy sampling effectively approximates a K-shortest-path enumeration while being driven by the learned GCML utility.

E Supplementary figures for Simulation 3

E.1 Measure the difficulty of the training dataset

The *conditional probability* was introduced to evaluate the difficulty of the tiling problem. The challenge was assessed by calculating the conditional resolvable rate (RR) at each step, based on the previous step. The RR at step i was defined as the ratio between the number of resolvable silhouettes (RS) and the total number of possible silhouettes after randomly removing i building blocks (BB) from a resolvable silhouette at step $i-1$:

$$\mathrm{RR}_i = \frac{n_s}{n_a}. \tag{24}$$

A silhouette was considered resolvable (as a RS) if all pixels in it could be removed to solve the silhouette.

For $i = 1, \ldots, 5$, the average conditional RR for all silhouettes in the test dataset is shown in Fig. S7(a). At each step, removing one BB resulted in approximately a 50% chance of the remaining silhouette being resolvable.

Considering a random removal strategy, the probability of finding a resolvable silhouette at step i was represented as the product of the average RR values from step 1 to step i, reflecting the *global probability* derived from conditional probabilities. This global probability is illustrated in Fig. S7(b), showing a very low chance of finding a solution using a random removal strategy at step 5 in the current test dataset.

In contrast, after training, the GCML method (regardless of whether the same silhouette was removed) achieved a resolvable rate of approximately ∼95% (as shown in Fig. 4e), indicating that GCML effectively learned to solve the tiling problem through the training process.

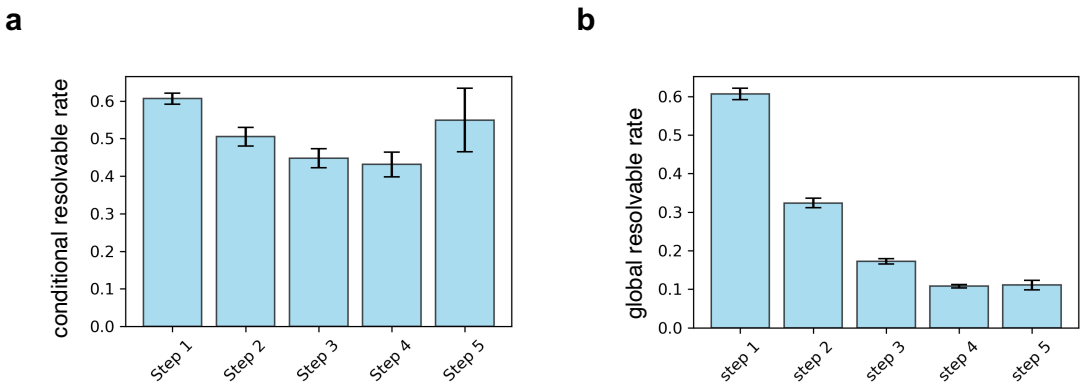

Fig. S7: **Measuring the Difficulty of the Tiling Problem.** (a) shows the conditional resolvable rate (RR) at each step. (b) represents the global resolvable rate, computed as the product of the RR at each step, indicating the cumulative probability of identifying a resolvable silhouette using a random removal strategy. The error bars in (a) and (b) denote the standard error over 1,000 test silhouettes.

E.2 Generated solutions

Fig. S8 shows four example trajectories generated by GCML for a given initial silhouette $\mathbf{o}_t$. The first three trajectories successfully decomposed $\mathbf{o}_t$, while the last trajectory failed to find a solution.

Fig. S8: **Four example trajectories generated by the GCML to address the silhouette decomposition problem.** The initial silhouette is shown in the left. The four generated trajectories are placed on the right of the initial silhouette. The colors represent $\mathbf{g}_2(\mathbf{s}^* - \mathbf{s}_t)$, the delta vector that went through the context-dependent affordance function. The noise scale is set to 0.1.

E.3 Cognitive map for more complicated silhouette

Fig. S9 presents cognitive maps similar to Fig. 4d but for more complex silhouettes. Panels Fig. S9a and Fig. S9b depict the cognitive maps of silhouettes containing two BBs, while Fig. S9c illustrates the cognitive map of a silhouette with three BBs. Despite the increased complexity of the cognitive map structure, GCML successfully maintained a meaningful cognitive map, providing reliable directional guidance even for these intricate shapes. This foundation enables GCML to generalize to more complex tasks, such as solving silhouettes with a greater number of BBs.

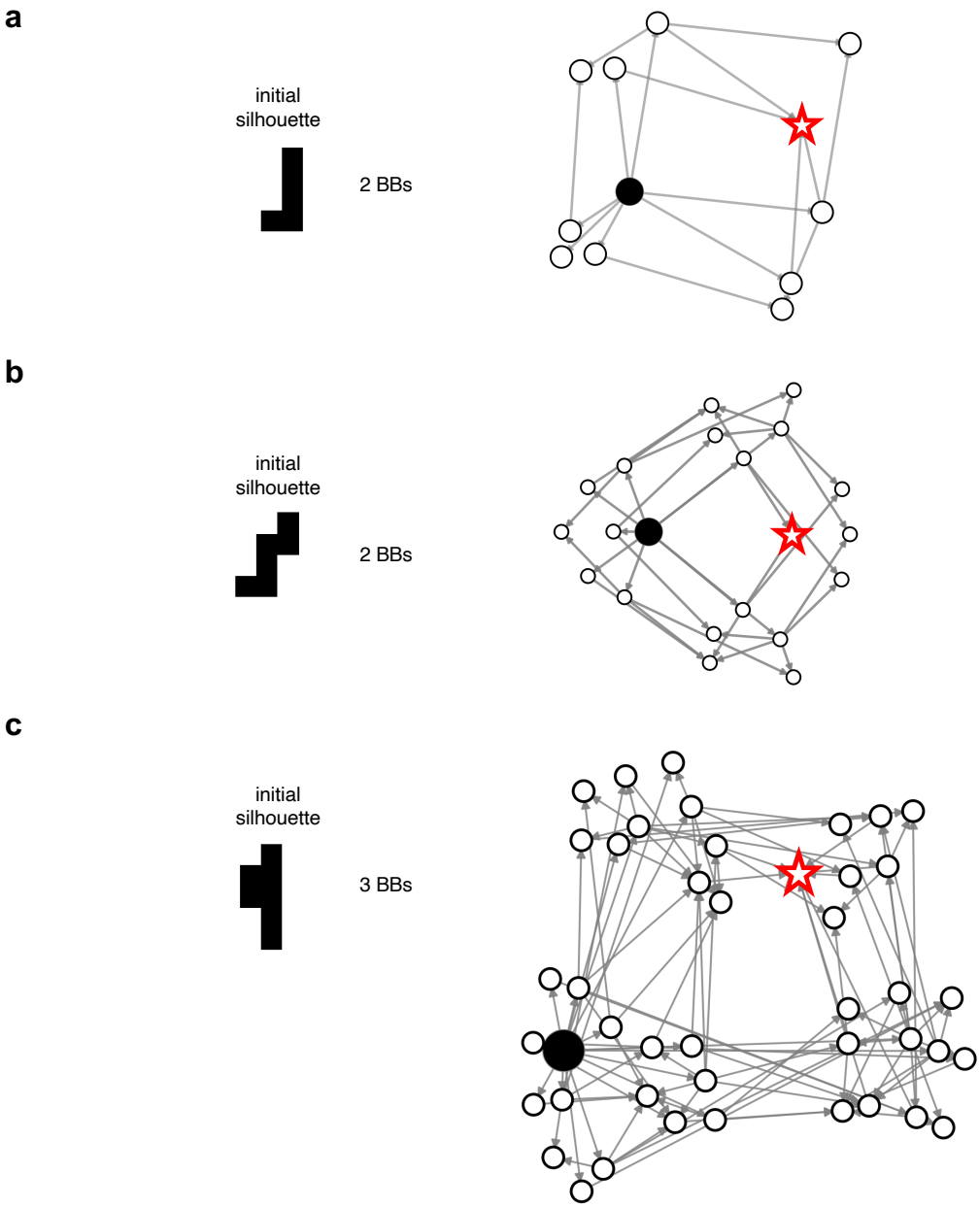

Fig. S9: **Cognitive map of GCML for the tiling problem.** Panels (a,b) show the cognitive map of a silhouette containing two BBs, while panel (c) shows the cognitive map of a silhouette containing three BBs. The left panels display the initial silhouette, and the right panels present their corresponding cognitive maps. The black node represents the high-dimensional state $\mathbf{s}_0$ of the initial silhouette, and the red star denotes the high-dimensional state $\mathbf{s}^*$ of the empty goal silhouette. All 2-D projections are generated via t-SNE (PCA initialized).

E.4 Generalization ability of GCML

The GCML strategy remained effective even when the test process involved silhouettes containing eight BBs. As shown in Fig. S10, GCML was still capable of identifying potential decompositions for these more complex silhouettes. The silhouettes with eight BBs were generated according to the following rules: all BBs had to remain within image boundaries and could not overlap; an initial BB was placed randomly within the image, with each position having equal probability; subsequent BBs

were added adjacent to the existing silhouette, requiring at least one pixel of adjacency to increase task complexity. None of the test silhouettes appeared in the training dataset.

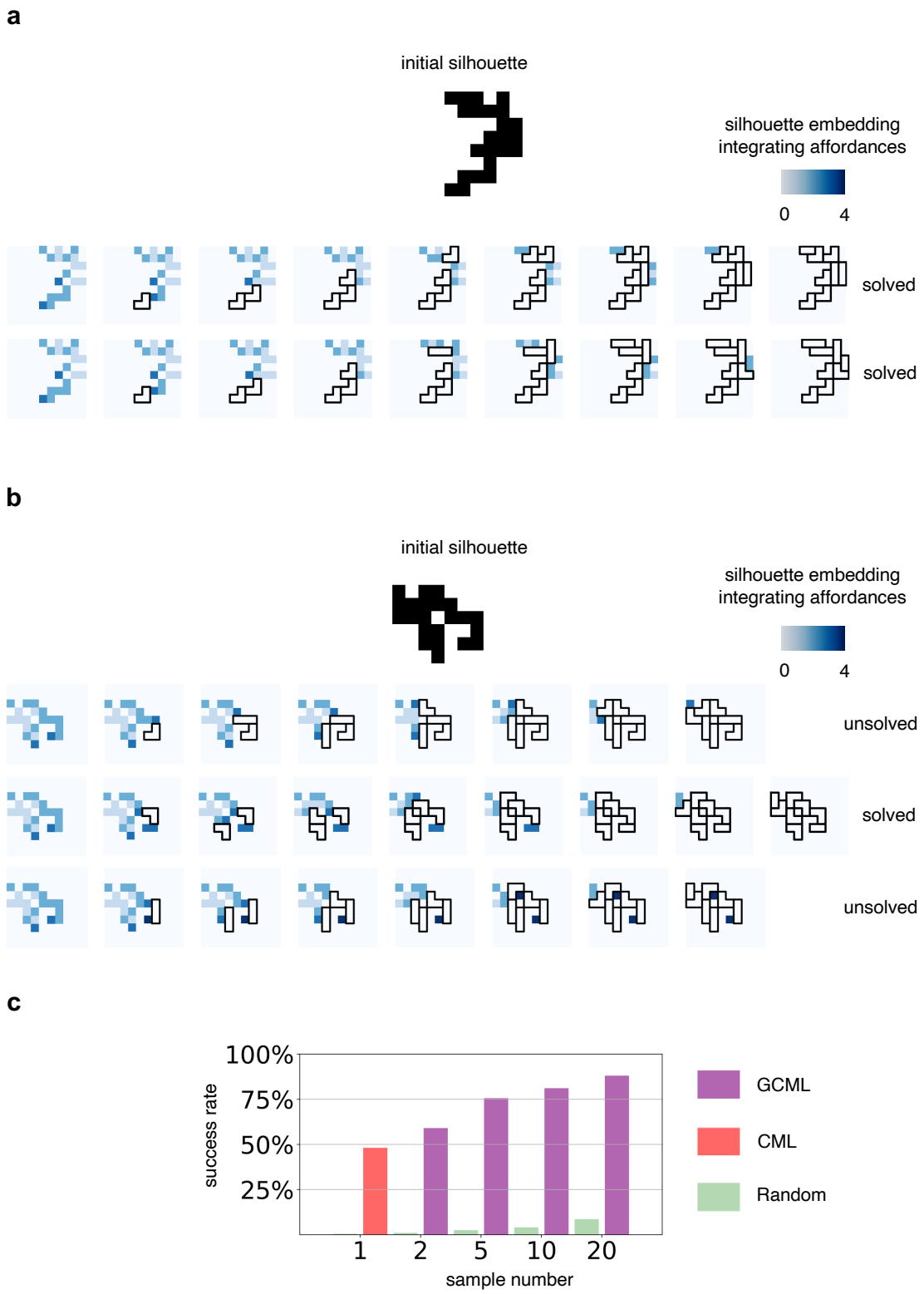

Fig. S10: **Tiling problem with more BBs in the test process.** GCML was still able to imagine the decomposition of the test silhouette when it contains eight BBs. Panels (a,b) show the trajectories generated by GCML to solve two different initial silhouettes, each containing eight BBs. The colors represent $\mathbf{g}_2(\mathbf{s}^* - \mathbf{s}_t)$, the delta vector that went through the context-dependent affordance function. The noise scale is 0.1. (c) shows the performance of GCML when the test silhouette contains eight BBs. The performance was evaluated using 1,000 test silhouettes.