Chapter 13: Simulation and planning with mental models

The pinnacle of flexibility is achieved through the use of mental models that support simulation and planning. This chapter discusses evidence for mental models in the brain. Building on the last few chapters, we formalize the function of mental models in terms of model-based solutions to sequential decision problems. Some model-based algorithms use offline simulation to provide synthetic data for training model-free algorithms. Other model-based algorithms use online simulation to evaluate different courses of action at decision time. The brain appears to implement both kinds of algorithms. Unifying these two approaches is the idea that the brain can imagine answers to "what if?" questions, liberating itself from the prison of pure experience.

The predictive maps introduced in the last chapter (the successor representation and its feature-based generalization) confer some flexibility upon agents, consistent with patterns of behavior such as latent learning, and their underlying neural mechanism in the hippocampus. However, predictive maps are still fundamentally constrained in certain ways. They compile the detailed transition structure of the environment into an actionable format that permits linear value computation. In doing so, they sacrifice the ability to use the transition structure to support an even greater degree of flexibility.

This chapter will review evidence that animals are capable of flexibility beyond the reach of predictive maps. We will focus on spatial navigation, which provides the most well-studied examples, and which carries ecological significance for many species. Experiments indicate that even animals with relatively simple nervous systems, such as ants, are capable of tracking their spatial location and using this information for charting a path toward specific landmarks. In some cases, animals are able to use their knowledge of the environment to plan detour and shortcut routes. We will consider what kinds of neural mechanisms could support these computations.

1 *Spatial navigation*

The desert ant *Cataglyphis*, despite its small brain (less than 1 million neurons), is capable of incredible navigational feats. While foraging, the ant will typically follow a circuitous path until it finds food. Rather than retracing its steps, it orients toward its nest and follows

Compare with around 70-100 million neurons in a mouse brain.

a more or less straight path home—a hole 1 mm in diameter, which might be over 100 m away (Figure 1).

A clever experiment illuminates one aspect of the ant's navigation algorithm. If the ant is displaced far from its nest after finding food (so that it can't rely on visual landmarks), it will orient in the direction where its nest would have been had it not been displaced, and follows a straight-line path that terminates close to the counterfactual location of the nest (Wehner and Srinivasan, 1981). Moreover, ants trained with consistent food locations follow straight-line outbound paths from the nest back to those locations (Collett et al., 1999). These "food vector" memories appear to last the ant's entire lifetime.

It has been asserted (controversially) that desert ants, as well as certain foraging bees and wasps, accomplish these navigational feats by using a "cognitive map" (or mental model) of space similar to the kind of cartographic map used by human navigators (Gallistel, 1990). According to this interpretation, ants keep track of their 2D location relative to the nest, and then use this to chart a "homing" vector back to the nest. The hypothesized algorithm for keeping track of position—*path integration*—is essentially identical to the *dead reckoning* algorithm used by human sailors for thousands of years. The core idea is that position is the integral of velocity over time, so that a position vector can be computed by adding up movements over time. The same algorithm is used by many animals, including some mammals (Etienne and Jeffery, 2004).

Path integration exemplifies an elementary form of model-based reasoning: what happens to my location when I move? In terms of the sequential decision framework studied in the last few chapters, we can identify this knowledge with a representation of the transition function, with locations as the state and movement vectors as the actions. In the next section, we describe path integration formally, and then turn to how this might work in mammalian brains. Despite its usefulness, path integration is also limited in certain ways: it applies specifically to spatially organized state spaces, where movement velocity has a unique effect on state transitions, and it relies on idiothetic (self-generated) cues rather than more abstract knowledge of environment structure. Despite these limitations, access to a path integration system can be harnessed by more powerful algorithms as a forward simulator even in the absence of overt movement (i.e., a form of imagination), opening the door to online planning and offline learning abilities. As we'll see, path integration can sometimes be applied to more general "conceptual" state spaces that can be represented in an approximately Euclidean form (i.e., a low-dimensional manifold in a metric space).

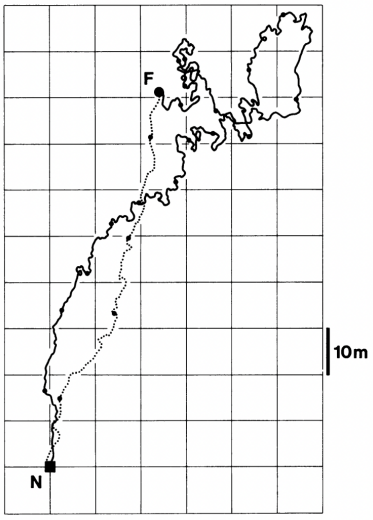


Figure 1: **Foraging (solid line) and return (dotted line) trajectories of the desert ant *Cataglyphsis*.** F = food, N = nest. Reproduced from Müller and Wehner (1988).

Small errors accumulate over time, so animals need to correct their positional estimates using landmarks and other sources of information.

1.1 Path integration

Let $s_t \in \mathbb{R}^2$ denote the agent's 2D spatial position at time t. The agent's action $a_t \in \mathbb{R}^2$ is a velocity vector, which specifies the instantaneous change in position:

$$\dot{s}_t = a_t. \tag{1}$$

If the position variable is initialized to $[0, 0]$ at time 0, the path integral of velocity yields position in the reference frame centered at the initial position:

$$s_t = \int_0^t a_{\tilde{t}} d\tilde{t}. \tag{2}$$

For the foraging ant, the initial position is the nest, so that the integrated position is represented in nest-centered coordinates. Once food is found, the ant can chart a path home by aligning its movement direction with the homing vector, $-s_t$. Once the homing vector is close to $[0, 0]$, the ant knows that it's in the vicinity of the nest (up to noise in the integrator). In addition, it can store the food location in memory, so that it can return to it from the nest by aligning its movement direction with the vector pointing at the stored location.

1.2 Continuous attractor network

While the preceding section provides a useful abstraction, the brain needs to implement these computations with neurons. A standard implementation uses a network of recurrently connected neurons with population activity x and recurrent weight matrix W. Its continuous-time firing rate dynamics are given by:

Our treatment is similar to previous continuous attractor models (e.g., Burak and Fiete, 2009).

To keep the notation light, we've dropped the time index here.

$$\tau \dot{x} = -x + \phi(Wx + Ga), \tag{3}$$

where $\phi(\cdot)$ is a non-linearity (e.g., a rectified linearity, to prevent firing rates below 0), τ is a time constant, and a is the velocity drive acting through the feedforward matrix G, which implicitly encodes the preferred movement direction θ_i for each neuron i. Each neuron also has a preferred position $\bar{s}_i$, which governs the "difference of Gaussians" connectivity profile:

You can think of g_i, the ith row of G, as a vector pointing in the preferred movement direction for neuron i.

$$W_{ij} = K(\bar{s}_i - \bar{s}_j - g_j), \tag{4}$$

$$K(s) = \exp[-\alpha \|s\|^2] - \exp[-\beta \|s\|^2], \tag{5}$$

where α and β control the relative strength of excitation and inhibition. With this connectivity profile, neurons that prefer nearby positions excite one another. The inhibitory term produces surround suppression, whereby neurons with preferred positions just outside

the neighborhood of a given neuron send inhibitory input. The shift term g_j skews the flow of activity in the direction that the agent is moving.

This network will form stable "bumps" of activity at particular positions. Stability means that the bumps will be restored following small perturbations of the activity state—the bumps are "attractors" in dynamical systems terminology. The set of attractors forms a continuous manifold in position space: smooth changes in position generate locally Euclidean changes in neural activity. Thus, this kind of network is referred to as a *continuous attractor network*.

We can think of the network's activity as organized on a sheet, where neurons are placed topographically based on their preferred location. Each patch of the sheet contains neurons covering all preferred directions. When the animal is stationary at a particular location, a periodic grid of neurons is activated on the sheet (Figure 2), similar to the entorhinal grid cells introduced in the last chapter. The periodicity comes from the center-surround interactions between nearby neurons on the sheet. To avoid edge effects, some models assume periodic boundary conditions, where neurons near the edge of the sheet wrap around to the opposite edge (Figure 3).

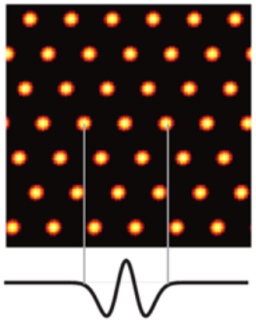


Figure 2: **Activity on the neural sheet**. The center-surround kernel, $K(x)$, is shown on the bottom. Reproduced from Burak and Fiete (2009).

Note that the sheet does not correspond to the true anatomical arrangement of neurons in the brain; it's a conceptual device to facilitate visualization.

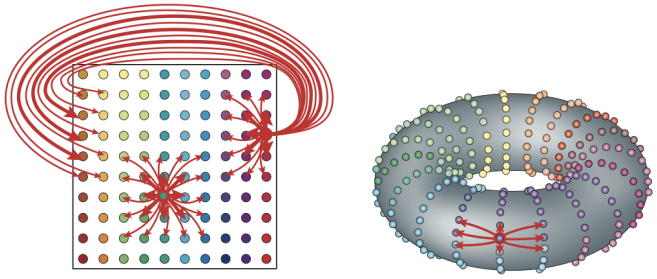


Figure 3: **Neural connectivity with periodic boundary conditions**. (Left) Neural connectivity wraps around the neural sheet. (Right) The connectivity is topologically equivalent to a torus. Reproduced from McNaughton et al. (2006).

A non-zero velocity drive causes the activity bump to translate on the sheet. For grid cells, this means that individual neurons fire periodically as an animal traverses space, while maintaining their phase relationships (as observed empirically; Yoon et al., 2013). The critical question is whether these activity dynamics carry a stable representation of the animal's position that is updated correctly based on the animal's motion. In other words, can downstream neurons accurately decode position from the pattern of activity? One way to do this is using the phase of neural activity relative to the starting point of the traversal: position can be decoded linearly from phase up to the period of the grid. Geometrically, this corresponds to identifying a position on a ring wrapping around the torus representation of the sheet (see Figure 3). Using this readout, Burak and Fiete (2009) showed with simulations that the total positional error accumulated

Phase here refers to the phase of the 2D center of the bump on the sheet (i.e., the attractor phase). The activity of the population does not itself have a fixed phase, since the neurons do not fire synchronously.

over 20 minutes (covering around 260 meters) is less than 15 centimeters.

At first glance, it seems strange to use a periodic representation of space to keep track of position. If position can only be identified up to the grid period, this representation will be useless for any environments that are larger than the grid period. Critically, the entorhinal cortex contains multiple modules, as described in the last chapter. Each module has a different period, giving rise to a multi-scale representation of space. By combining the modules, positional information can be tracked over very large spaces (Fiete et al., 2008). To understand why, let's examine a simplified problem: representation of 1-D position.

Let λ_k denote the period of module k. The attractor phase in module k can be written as $\theta_k = \hat{s} \mod \lambda_k$, where $\hat{s}$ is the animal's position estimate and "mod" refers to the modulo operator. The phase vector $\theta = (\theta_1, \ldots, \theta_K)$ collects together the phases from K different modules. This phase code is an example of a *residue number system*, which can uniquely represent a number of positions that is exponential in the number of modules. In the (admittedly contrived) setting where the periods are whole numbers and share no common factors (i.e., they are co-prime), then one can apply the *Chinese remainder theorem* to prove that any number in the range 0 to $\prod_k \lambda_k - 1$ can be uniquely represented with the phase code (Fiete et al., 2008). For example, over 1 million positions could be represented with only 5 modules. The vast capacity of this representation means that any excess capacity could be utilized for error-correcting redundancy, thereby conferring robustness in the face of spiking noise.

Another computationally useful property of this representation is that each phase can be updated in parallel when the animal moves; no interaction is needed between the modules. This allows distributed tracking of position.

In summary, this section has shown how path integration can be implemented in a velocity-driven continuous attractor network, thought to exist in the entorhinal cortex. Returning to our main theme, the purpose of this exercise was to show how an ecologically significant form of model-based knowledge (what happens to my position when I move?) can be implemented in biologically plausible neural circuitry.

Further evidence that the entorhinal cortex is important for path integration comes from the finding that lesions to this area impair the ability of rats to find their way back to a starting refuge using only self-motion information (Parron and Save, 2004).

1.3 Connection to predictive maps

In the last chapter, we suggested that grid cells might provide a low-dimensional basis (specifically, an eigendecomposition) of the predictive map (successor representation) in the hippocampus, possibly in the service of regularization. While this proposal seems fundamentally different from the idea that the grid cells encode a transition model for path integration, the transition function and the successor representation share the same eigenvectors.

Technically, this requires the transition matrix to be diagonalizable, a condition satisfied when the Markov chain is reversible: $T^\pi(s'|s)\mu^\pi(s) = T^\pi(s|s')\mu^\pi(s')$, where μ^π is the stationary distribution of the Markov chain.

When the velocity drive is close to 0, or random exploration generates isotropic velocity signals (i.e., the agent moves in any direction with equal probability), the neural activity "relaxes" towards the eigenvectors. Thus, a low-dimensional predictive representation, as described in the last chapter, is intrinsic to the network dynamics. When the velocity drive is non-zero/anisotropic, the population activity bump tracks position by integrating velocity, mirroring movement-driven transition dynamics. To summarize, the same network has different modes of operation: exploration-driven (low-dimensional predictive representation) and velocity-driven (path integration).

1.4 Shortcuts and detours

Having access to a spatial model enables more than tracking position and computing homing vectors. An agent can also plan shortcuts and detours, by using the same path integration mechanism in its imagination rather than during actual movement. Before describing how such *mental navigation* can be implemented, let's look at a few empirical examples.

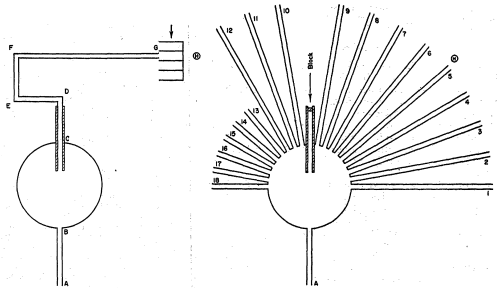


Figure 4: **Tolman shortcut experiment**. (Left) Training apparatus. (Right) Test apparatus. "H" indicates the goal location. Adapted from Tolman et al. (1946).

In a classic demonstration of shortcutting behavior, Tolman et al. (1946) first trained rats to take an indirect route from a circular platform to a goal location containing food. After this initial training, rats were placed back on the circular platform, but this time they could choose from several alleys radiating from the platform (Figure 4). The key finding was that rats preferred the alley leading directly to the food. This finding is closely related to the homing behavior discussed above: essentially, the rats are "homing" toward the goal location.

Unfortunately, Tolman's experiment had two major problems. First, a light bulb was suspended above the goal location throughout training and testing; the rats could have used this as a beacon, rather than relying on a cognitive map of space. A similar confound also afflicted many later studies of shortcut behavior. Second, the shortcut doesn't require any flexible planning or spatial reasoning; the rats simply have to run in a single direction until they reach the food.

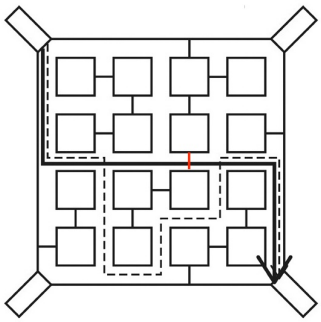


Figure 5: **Sato shortcut experiment**. Red line shows the wall that was removed in the shortcut test. The dashed black line shows the path rats took during training; the solid black line shows the path rats took during the shortcut test. Adapted from Sato et al. (2018).

Later work addressed these problems. Roberts et al. (2007) showed that rats still take shortcuts in enclosed mazes, where they can't make use of distal cues as beacons. Sato et al. (2018) developed a "lattice maze" where rats could take circuitous paths to a goal. After training, a shortcut was created by removing one of the walls in the lattice (Figure 5). Rats were more likely to take the shortcut than a longer detour path, demonstrating that they are capable of flexible shortcutting behavior.

A fascinating (and under-appreciated) example of shortcutting behavior comes from a study by Zanforlin and Poli (1970), who took advantage of rats' natural tendency to burrow underground tunnels. They first trained rats to follow an S-shaped tube between boxes, and then observed as the rats burrowed tunnels (Figure 6). Strikingly, of the rats that successfully reached the goal box, the paths they burrowed were shorter than the S-shaped training tube.

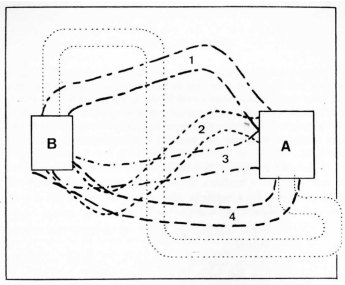


Figure 6: **Burrowing rats (sometimes) take shortcuts**. Out of 10 rats in the experiment, 5 burrowed from box A to box B, and one missed it by 10 cm. Four of these tunnels are shown, along with the S-shaped tube used during training. Reproduced from Zanforlin and Poli (1970).

Another example of flexible mental navigation is the ability to adapt in response to novel obstacles. Tolman and Honzik (1930) studied a maze with multiple possible paths to a goal (Figure 7). When they blocked one of these paths, rats tended to take the shortest path that would get them to the goal.

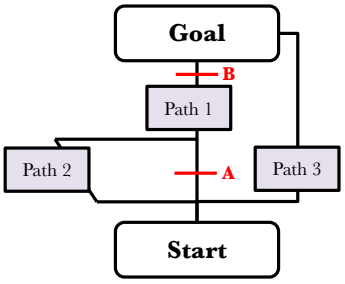


Figure 7: **Schematic of Tolman detour maze**. Red lines show two block locations. Based on apparatus described in Tolman and Honzik (1930).

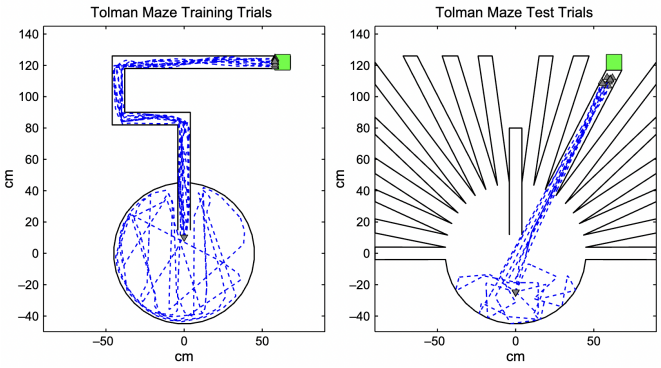
We now turn to the computational question: how does the brain do it? It's not enough to compute the direction of the goal, because in many of these cases animals have to move *away* from the goal in order to take the shortest path. What's needed is the ability to look ahead in a cognitive map, until a sufficiently short path is found. Tolman (1948) pointed out that rats often pause at choice points (intersections) in a maze, looking in each direction before moving again. He dubbed this behavior *vicarious trial and error*, based on the hypothesis that the rats are mentally simulating different paths (sometimes to unseen goals), testing whether each path is successful until a successful one is found. Many years after this hypothesis was formulated, Johnson and Redish (2007) provided direct neural evidence for such simulation, showing that place cells swept ahead of the rat's actual position at choice points (Figure 8).

Figure 8: **Decoded spatial position from hippocampal ensembles at a choice point**. Note that the rat is not moving; thus, the decoded position is imagined. Reproduced from Johnson and Redish (2007).

Hippocampal sweeps are temporally organized by the theta rhythm (6-10 Hz), starting a little behind the rat's current position and ending slightly ahead of it about 100-160 milliseconds later, at the end of a theta cycle (Foster and Wilson, 2007). These sweeps preferentially project in the direction of the rat's current goals, extending

farther for more distant goals (Wikenheiser and Redish, 2015), and alternating between different possible paths on consecutive theta cycles (Kay et al., 2020). With more experience, rats stop exhibiting vicarious trial and error, but hippocampal sweeps continue to be observed, except now only projecting in the direction of the animal's eventual destination—mirroring the animal's stereotyped behavior.

Mechanistically, a path integration system could implement these forward sweeps by applying velocity drive to the integrator. In a cluttered environment, the velocity drive would need to point not directly at the goal but rather at locally accessible subgoals. Indeed, hippocampal sweeps appear to be chunked based on significant landmarks or choice points (Figure 9). One proposal for how this might work (Erdem and Hasselmo, 2012) uses head direction cells (neurons tuned to specific head directions) to supply the directional component of velocity drive. The resulting sweeps produce a form of locally linear look-ahead; by chaining these sweeps together, the simulation can eventually reach a goal (Figure 10).



The question remains how the head direction cells know what direction to activate. Erdem and Hasselmo (2012) proposed that the head direction cells are guided by a representation of reward signals at particular locations; critically, this signal diffuses through the cognitive map, such that a short linear look-ahead can contact it and subsequently project further look-aheads in that direction. This provides the key mechanism for shortcut discovery, since the diffusion will tend to concentrate along shorter paths.

The "reward map" is thought to be implemented in the prefrontal cortex, where neurons tuned to both location and reward have been identified. The relatively large place fields of these neurons (compared to hippocampal place cells) may lend themselves to reward diffusion (Hok et al., 2005). Prefrontal place cells are coordinated with the hippocampal theta rhythm during navigation (Tang et al., 2021), and are predictive of subsequent hippocampal place cell activation

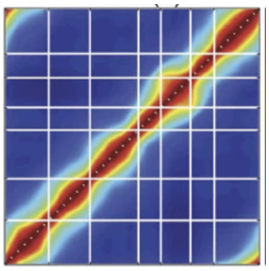


Figure 9: **Cross-correlation between positional probabilities decoded from hippocampal activity**. White lines indicate landmarks in the maze. The "pinches" in the cross-correlation indicate that positional probabilities are more strongly correlated within than between landmarks. Reproduced from Gupta et al. (2012).

Entorhinal neurons tuned to speed (speed cells) might provide the speed component of velocity drive (Kropff et al., 2015).

Figure 10: **Linear look-ahead in the Tolman shortcut maze**. Dashed lines show simulated trajectories. Reproduced from Erdem and Hasselmo (2012).

during deliberation (Hasz and Redish, 2020).

The linear look-ahead + reward diffusion model has some intriguing similarities with modern planning algorithms, which have been instrumental to the design of high-performance artificial intelligence systems. For example, AlphaGo (Silver et al., 2016) uses a value function approximator to guide look-ahead. The values are estimated with a form of temporal difference learning (see Chapters 10 and 11). In an environment with a single goal, this produces a diffusion-like propagation of information. An important difference is that general planning algorithms do not typically assume that the state space is Euclidean (an assumption necessary for linear look-ahead). Below we will discuss algorithms applicable to more general state spaces. But first we will look at examples where non-Euclidean state spaces can be treated in a Euclidean way.

In 2015, AlphaGo became the first program to beat a professional Go player without a handicap. A few years later, it beat the top-ranked Go player in a 3-game match.

1.5 Mental navigation in non-Euclidean state spaces

A Euclidean state space is geometrically self-consistent in the sense that if you add together a sequence of translations and end up where you started, then the sum of the translations must equal 0—a *loop closure* property. This is the basis of the desert ant's homing behavior: following the homing vector ensures that its path integrator returns to 0, coinciding with arrival at its nest. Non-Euclidean state spaces don't necessarily obey loop closure. For example, the Earth is only locally Euclidean (the ground looks flat over short distances); over longer distances, the curvature of the Earth starts to matter, and the homing vector will not necessarily get the ant back to its nest, even if the path integrator returns to 0.

The brain needs to plan in many state spaces that are not obviously Euclidean (e.g., imagine you're adjusting the knobs in a shower to control water temperature and pressure). The question is whether we can represent these spaces in a way that is at least approximately Euclidean. If we can, then we can harness the path integration capability of the entorhinal cortex for planning. This might be the case for some "conceptual" state spaces. Constantinescu et al. (2016) used functional MRI to identify brain areas in humans that exhibited grid-like periodicity in their responses to changes in a conceptual space (the neck and leg lengths of a bird silhouette). By associating particular points in this "bird space" with unique outcomes (Christmas symbols), humans could be trained to plan morphs that achieve specific outcomes (Figure 11). The entorhinal cortex was among several areas that exhibited grid-like periodicity over bird space.

Another example comes from a study of monkeys performing a 1-D mental navigation task (Figure 12). They were trained to move

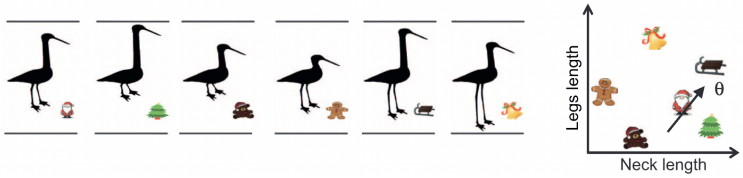


Figure 11: **A conceptual space and associated outcomes**. (Left) Example birds and outcomes. (Right) Euclidean representation. Reproduced from Constantinescu et al. (2016).

through an abstract space of landmarks using a joystick, and could eventually perform this task with high accuracy. When they moved the joystick, movement through the space was invisible (they couldn't see the sequence of landmarks intervening between the start and end points); nonetheless, neurons in entorhinal cortex exhibited periodic firing patterns, with bumps that aligned to the expected timing of landmark arrivals.

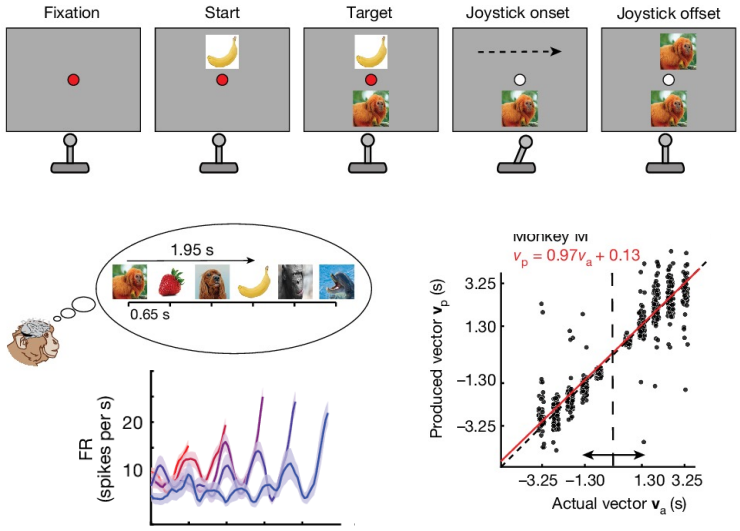


Figure 12: **Signatures of mental navigation in an abstract space**. Monkeys used a joystick to navigate between visual "landmarks" (images) arranged on a 1-D line, illustrated in the schematic. Monkeys learned to produce the correct translation vector that would bring them to a target landmark. As the translation was unfolding invisibly, entorhinal cells fired periodically, with a period matched to the inter-landmark interval. Adapted from Neupane et al. (2024).

These studies suggest that the periodic structure of grid cells in entorhinal cortex extend beyond spatial navigation, reflecting a common underlying principle. One way to formalize this principle is to optimize a recurrent neural network that transforms neural state representations by running the dynamics forward (Whittington et al., 2020), as in Eq. 3. By mapping the state representations to observations, the network can be trained to match its predictions with these observations. In open-field environments this produces grid-like periodic representations. These representations form an abstract structural code in the sense that they can be reused in any task that shares the same underlying spatial structure.

Strictly speaking, the forward dynamics of recurrent neural network do not necessarily correspond to path integration, because there's no guarantee that the loop closure property will be satis-

fied. Iyer et al. (2024) addressed this by training neural networks with an explicit loop closure loss (penalizing them for violations of this property). This resulted in networks that were geometrically self-consistent, such that we could more properly interpret their dynamics as a form of approximate path integration. Moreover, the networks reproduced grid-like representations for several different domains (an example is shown in Figure 13). The training procedure is admittedly artificial, but perhaps points the way toward the kinds of inductive biases that the brain might use to discipline its learned representations.

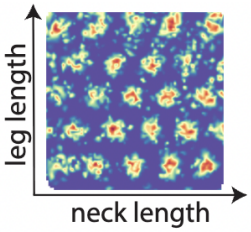


Figure 13: **Grid-like representation learned for the bird space**. Reproduced from Iyer et al. (2024).

2 *General planning algorithms*

We now turn to general planning algorithms that don't make Euclidean assumptions about the structure of the state space.

2.1 *Dynamic programming*

The key idea behind dynamic programming is the decomposition of a complex problem into a series of sub-problems by harnessing the recursive structure of the complex problem. In a Markov decision process (see Chapter 11), the recursive structure is represented by the Bellman optimality equation:

$$Q^*(s,a) = R(s) + \gamma \sum_{s'} T(s'|s,a) \max_{a'} Q^*(s',a'), \tag{6}$$

Here we have assumed discrete states, but an analogous equation applies to continuous states.

which decomposes the optimal value function Q^* (i.e., the value function under the optimal policy) into the sum of the immediate reward and the expected reward at the next state. This optimality equation can be used to define a simple dynamic programming algorithm known as *value iteration* (Sutton and Barto, 2018). Start by initializing estimates $\hat{Q}^*(s,a)$ arbitrarily, and then iterate over states, applying a "Bellman backup" each time:

$$\hat{Q}^*(s,a) \leftarrow R(s) + \gamma \sum_{s'} T(s'|s,a) \max_{a'} \hat{Q}^*(s',a'). \tag{7}$$

This algorithm will converge to the true optimal values.

Friedrich and Lengyel (2016) developed a biologically plausible implementation of dynamic programming, similar to value iteration. Here we present their model in a slightly simplified form. They posited a population of neurons tuned to different state-action pairs, (s_i, a_i), where i indexes neurons. Using a rate-based approximation, the membrane potential dynamics are modeled as linear and recurrent:

A similar approach can be used with function approximation, but we omit this for brevity (see Friedrich and Lengyel, 2016).

$$\tau \dot{\mu}_i = -\mu(t) + \sum_j W_{ij} x_j(t) + I_i(t), \tag{8}$$

where τ is the membrane time constant, $\mu_i(t)$ is the membrane potential of neuron i, $x_i(t) = [\mu - \theta]_+$ is the firing rate of neuron i (a rectified linear function of the membrane potential, with threshold θ), and $I_i(t) = R(s_i)$ is the input current representing the expected reward for the state encoded by neuron i. The recurrent weights are parametrized as the sum of excitatory and inhibitory components:

$[x]_+ = x$ if $x > 0$, otherwise $[x]_+ = 0$.

Friedrich and Lengyel express the expected reward as a function of the state-action pair, but for consistency we use $R(s)$ instead of $R(s, a)$.

$$W_{ij} = W_{ij}^{\text{exc}} + W_{ij}^{\text{inh}} \tag{9}$$

$$W_{ij}^{\text{exc}} = \gamma T(s' = s_j | s = s_i, a = a_i) \tag{10}$$

$$W_{ij}^{\text{inh}} = \mathbb{I}[i = j] - \mathbb{I}[s_i = s_j]. \tag{11}$$

The excitatory component encodes the transition function (with post-synaptic neurons representing the current state and pre-synaptic neurons representing the next state). The inhibitory component encodes the mutual exclusivity between different actions for the same state: if the actions are different and the states are the same, this component is negative. Friedrich and Lengyel showed that the firing rates of the population converge to a fixed point that will (under suitable conditions) correspond to the optimal value function, up to a constant that depends only on the firing threshold θ and the discount factor γ:

The summation is taken over all neurons coding for state s.

$$V^*(s) = \sum_{i:s_i=s} x_i - \frac{\theta}{1-\gamma}, \tag{12}$$

where $V^*(s) = \text{argmax}_a Q^*(s, a)$. The optimal policy is to choose the action coded by any of the currently active neurons that correspond to the current state, for example by choosing the action coded by the most active neuron: $a^*(s) = \text{argmax}_i x_i \mathbb{I}[s_i = s]$.

Figure 14 illustrates the model behavior on a sequential movement selection task studied by Sohn and Lee (2007). Monkeys were trained to execute a hand movement (selecting a visual target with a cursor) in order to move to the next state. If they made no mistakes in the sequence, they were given a juice reward; otherwise, they had to start over. Although in principle the monkeys could solve this task in an online manner (paying attention only to the current stimulus, without planning), their response times suggested that they were planning ahead. In particular, their response times were longest when the number of remaining movements (NRM) was large.

Figure 14 shows activity recorded from the pre-supplementary area (pre-SMA), a frontal cortical region known to be involved in motor planning (Nachev et al., 2008). For example, neurons in pre-SMA respond when a monkey needs to change the direction of a forthcoming arm movement (Matsuzaka and Tanji, 1996). Disruption

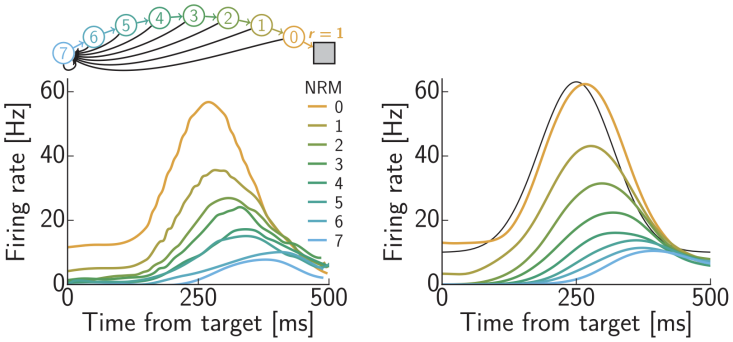


Figure 14: **Neural activity and model predictions during sequential movement selection**. (Left) Activity in the presupplementary motor area (pre-SMA) as a function of the number of remaining movements (NRM). The inset shows the state transition structure: colored lines denote correct actions, black lines denote incorrect actions, and the grey square denotes the goal state. The numbers on each state indicate the corresponding NRM. (Right) Firing rates of value neurons generated by the model. The black line shows the reward input, which was fitted to the data. Reproduced from Friedrich and Lengyel (2016). The neural data is taken from Sohn and Lee (2007).

of pre-SMA activity impairs the ability to alter upcoming action sequences (Kennerley et al., 2004). Thus, pre-SMA is a plausible site of goal-directed value computation. Sohn and Lee found that neurons respond earlier and more strongly when the NRM is small, a pattern recapitulated by the Friedrich and Lengyel model. The effect of NRM on neural response latency arises in this model from spreading activation, which is needed to propagate value information through the network. The effect of NRM on response amplitude arises from discounting. Friedrich and Lengyel also showed that the model could closely mirror the NRM-dependent response times by assuming that movements are generated whenever the relative activity of the corresponding neurons crosses a threshold.

2.2 *Tree search*

Dynamic programming is a powerful and widely applied family of algorithms, but it suffers from a major limitation. Because it's designed to compute optimal values for every state-action pair, it becomes intractable for large state-action spaces. This is why modern AI systems like AlphaGo, which are designed to work on the extremely large state spaces of games like Go, do not use dynamic programming. Instead, they use "rollouts" of simulated state-action sequences initiated from the current state. These rollouts can be understood as search through the decision tree rooted at the current state, hence the name *tree search* for this general family of algorithms. Unlike dynamic programming, tree search algorithms estimate the value function locally, so they don't suffer from unfavorable scaling with the size of the state space. On the other hand, this locality means that action policies are not guaranteed to be globally consistent over the entire state space. Moreover, the stochastic nature of the rollouts introduces finite sampling errors when the transitions and rewards are not deterministic, and the estimates can also be biased due to finite truncation of the rollout when the planning horizon is

The linear look-ahead algorithm described above can be viewed as a kind of rollout specialized for spatial tasks.

See Daw and Dayan (2014) for further discussion of different trade-offs involved in the design of tree search algorithms.

infinite (i.e., the task doesn't have a terminal state).

The trick to making tree search algorithms efficient is designing the rollout policy to selectively search promising parts of the decision tree (or equivalently pruning unpromising parts; see Figure 15). A simple heuristic is to myopically prune sub-trees (rooted at the current state) whenever an unfavorable outcome is encountered. There is some evidence that humans follow this heuristic, relinquishing high-payoff paths that require traversing a large loss (Huys et al., 2012). A more sophisticated, non-myopic heuristic uses a learned value function (e.g., via TD learning) to guide the rollouts. This is a key design feature of systems like AlphaGo. Evidence from experiments in which human subjects are asked to externalize their rollouts (by trying out paths before committing to one) suggests that they can adopt non-myopic heuristics, taking into account both the long-term value and their own uncertainty when selecting rollouts (Fan et al., 2025).

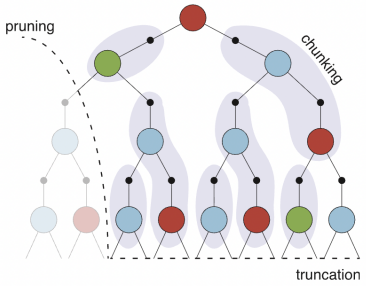


Figure 15: **Tree search heuristics**. Each node represents a state, and each edge represents an action. Reproduced from Mattar and Lengyel (2022).

Monte Carlo tree search algorithms (Browne et al., 2012), widely used in modern AI, construct rollouts stochastically by following an exploration policy.

Another trick that exploits a value function estimate is to execute limited-depth rollouts (truncation, as shown in Figure 15) and then add the accumulated reward to the value estimate of the final state. This works because the Bellman equation can be expanded in the following way:

$$V^\pi(s) = \mathbb{E}[r_1 + \gamma r_2 + \cdots \gamma^{K-1} r_K + \gamma^K V^\pi(s_{K+1}) | s_1 = s], \qquad (13)$$

This is also known as *partial evaluation* in computer science.

where K is the rollout depth. In other words, the value function for depth $K+1$ "completes" the partial value estimate at the current state estimated using the rollout. Humans typically plan 3-6 steps ahead (at least on games such as chess, although professionals can sometimes plan more deeply; De Groot, 1978); planning depth can vary adaptively with task demands (Eluchans et al., 2025) and expertise (Van Opheusden et al., 2023). Which plans humans ultimately select depends on both their depth-limited rollout and the estimated value at the end of the rollout (Keramati et al., 2016).

Chunking (Figure 15) is a heuristic that simplifies the planning problem by abstracting away the fine structure of the decision tree. This is familiar to anyone planning a complex activity like a vacation: planning initially unfolds at a high level (e.g., how to rent a car or buy plane tickets), before figuring out the low-level details (e.g,. what muscles to move in order to open a door or type on a computer). One way that chunks form is through repetitive action or state sequences (Dezfouli and Balleine, 2013; Tomov et al., 2020). These sequences may get chunked together into a single unit; a hierarchical planning algorithm can operate on this unit rather than on the low-level states/actions. One consequence of this chunking is that humans forego optimal paths if they can reuse a chunk in a good but

suboptimal path (Huys et al., 2015).

3 *Using simulation for learning*

We've now seen several ways in which "model-free" algorithms (like TD learning) can assist "model-based" algorithms like tree search—by guiding rollouts and completing partial evaluations. This form of interplay is based on the idea that the model-based planner ultimately controls behavior, whereas the model-free learning algorithms operate in the background and provide input to the planner. We could alternatively flip this on its head, handing control to the model-free algorithm and allowing it to call upon a model-based system for assistance. This is the essential idea underlying architectures like Dyna (Sutton, 1991), which uses a model to simulate synthetic data that is fed into a model-free learning system (Figure 16). All interactions with the environment are controlled by the model-free system.

The same idea appears in modern AI systems like Dreamer (Hafner et al., 2025).

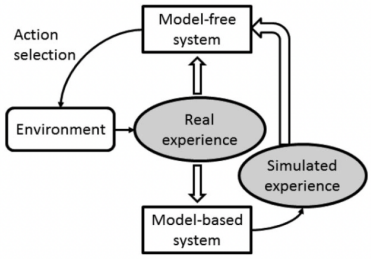


Figure 16: **The Dyna architecture**. Reproduced from Gershman et al. (2014).

The principal advantage of this architecture is that action selection is very efficient, since it doesn't require any planning. Instead, model-based knowledge is compiled (via simulation in the background) into an efficiently actionable form such as cached values or a computationally cheap function approximator. One implication of this architecture is that the model-based system can generate synthetic data which the model-free system has never experienced—i.e., a form of learning from imagination.

Consider for example the experimental design shown in Figure 17. Human subjects were first taught a simple one-step transition model (Phase 1), and then taught about rewards at each terminal state (Phase 2). In Phase 3, the state space was expanded, with the formerly terminal states now serving as initial states. Finally, in Phase 4, subjects were returned to the original initial states and asked to judge which state is better. By comparing these judgments to the same judgments after Phase 2, we can calculate a "revaluation score" that quantifies how much Phase 3 training altered their value estimates. The critical feature of this design is that subjects never experience an unbroken trajectory through the state space—what a conventional TD learning algorithm would require to correctly estimate values. Yet they were still able to update their values based on the Phase 3 training. However, if they were placed under cognitive load during Phase 3 (simultaneously performing a secondary task), the revaluation score was significantly attenuated. This is consistent with the hypothesis that they were running simulations during Phase 3.

Even more direct evidence for offline simulation comes from another experiment, using the same setup, where a rest interval is interposed between Phases 3 and 4 (subjects sat quietly in a room listen-

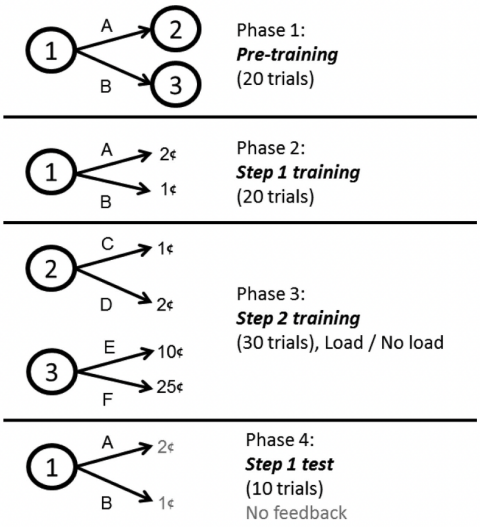


Figure 17: **Experimental design for studying simulation-based learning**. Reproduced from Gershman et al. (2014).

ing to music, without any specific task). This rest interval rescued the revaluation score from its attenuation under load, putatively by giving people more opportunity for offline simulation. Importantly, the subjects in these experiments were probably not using planning at decision time, because cognitive load during Phase 4 did not impair their performance.

Other experiments have shown that cognitive load selectively impairs model-based computation (Otto et al., 2013).

Another source of evidence for offline simulation is a functional MRI study using a similar design (Momennejad et al., 2018). This revealed reactivation of state representations in the hippocampus during a rest interval, specifically in a condition where values needed to be updated (significantly less reactivation was observed in a control condition where values were unchanged). The degree of reactivation was significantly correlated with the revaluation score, again only in the condition where values need to be updated.

A variation of the learning-by-simulation idea was explored by Jensen et al. (2024). They trained a recurrent neural network to maximize cumulative reward by outputting actions conditional on a set of inputs which included the most recent state, action, and reward. The key innovation was to endow the agent with a "cognitive" action—simulating a rollout using a learned world model. These rollouts were then fed back into the network as additional inputs, thereby influencing the policy. Because the simulation action is treated in the same way as all the other actions, the agent can learn to adaptively decide when and how much to simulate. Like humans, the agent spent more time simulating when it was farther from a goal and before the first action of a trial (Figure 18).

Jensen et al. (2024) used the same model to capture aspects of hippocampal place cell sequences (using data from Widloski and Foster,

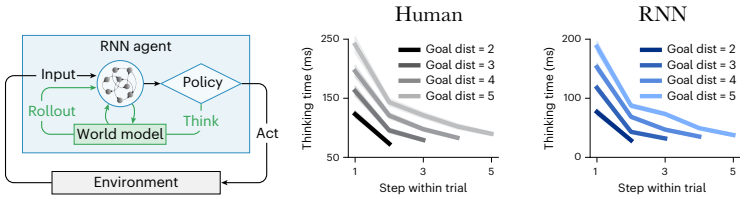


Figure 18: **Adaptive simulation**. (Left) Agent architecture. (Middle) Human thinking time on a maze navigation task. (Right) Recurrent neural network (RNN) thinking time (number of rollouts x 120 ms/rollout). Adapted from Jensen et al. (2024).

2022). Consistent with the data reviewed above, hippocampal sequences (forward sweeps) are interpreted as rollouts of the transition model under the learned policy. This interpretation is supported by the several facts: sequences tend to (i) avoid passing through walls; (ii) reach the goal; (iii) predict the next physical movement specifically when the sequence reaches the goal; and (iv) increase the rate at which the goal is reached over multiple sequences.

4 *Conclusion*

Model-based reasoning is often thought to be the pinnacle of cognition, underlying our most impressive feats of flexibility. Elements of an internal world model can be seen not only in humans but even in much simpler creatures like ants and wasps. This chapter reviewed several ways in which an internal model can be used: for homing behavior (in spatial navigation tasks), for goal-directed planning, and for mental simulation. We have not exhausted all the uses of internal models, but we focused on these because we know the most about their underlying neural mechanisms. In particular, we showed how path integration exemplifies an elementary form of model-based reasoning, how it can be implemented in a continuous attractor network, and how it can be repurposed for conceptual state spaces with approximately Euclidean topology. Going beyond path integration, we saw how forms of dynamic programming and tree search could be implemented neurally. It's possible that the brain has evolved to use all of these algorithms in different situations, though we are just at the beginning of understanding how the "meta-control" problem of choosing algorithms is solved neurally.

Study questions

1. What are other ways that model-free and model-based systems might interact?
2. What do you expect would happen to sequential choice behavior if the hippocampus is lesioned?

3. What are some computational trade-offs between online planning vs. offline simulation?

References

Browne, C. B., Powley, E., Whitehouse, D., Lucas, S. M., Cowling, P. I., Rohlfshagen, P., Tavener, S., Perez, D., Samothrakis, S., and Colton, S. (2012). A survey of Monte Carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in games*, 4:1–43.

Burak, Y. and Fiete, I. R. (2009). Accurate path integration in continuous attractor network models of grid cells. *PLoS Computational Biology*, 5:e1000291.

Collett, M., Collett, T. S., and Wehner, R. (1999). Calibration of vector navigation in desert ants. *Current Biology*, 9:S1.

Constantinescu, A. O., O'Reilly, J. X., and Behrens, T. E. (2016). Organizing conceptual knowledge in humans with a gridlike code. *Science*, 352:1464–1468.

Daw, N. D. and Dayan, P. (2014). The algorithmic anatomy of model-based evaluation. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 369:20130478.

De Groot, A. D. (1978). *Thought and Choice in Chess*. Walter de Gruyter.

Dezfouli, A. and Balleine, B. W. (2013). Actions, action sequences and habits: evidence that goal-directed and habitual action control are hierarchically organized. *PLoS Computational Biology*, 9:e1003364.

Eluchans, M., Lancia, G. L., Maselli, A., D'Alessandro, M., Gordon, J. R., and Pezzulo, G. (2025). Adaptive planning depth in human problem-solving. *Royal Society Open Science*, 12:241161.

Erdem, U. M. and Hasselmo, M. (2012). A goal-directed spatial navigation model using forward trajectory planning based on grid cells. *European Journal of Neuroscience*, 35:916–931.

Etienne, A. S. and Jeffery, K. J. (2004). Path integration in mammals. *Hippocampus*, 14:180–192.

Fan, H., Callaway, F., and Gershman, S. J. (2025). Uncertainty-driven exploration during planning. *Decision*.

Fiete, I. R., Burak, Y., and Brookings, T. (2008). What grid cells convey about rat location. *Journal of Neuroscience*, 28:6858–6871.

Foster, D. J. and Wilson, M. A. (2007). Hippocampal theta sequences. *Hippocampus*, 17:1093–1099.

Friedrich, J. and Lengyel, M. (2016). Goal-directed decision making with spiking neurons. *Journal of Neuroscience*, 36:1529–1546.

Gallistel, C. R. (1990). *The Organization of Learning*. The MIT Press.

Gershman, S. J., Markman, A. B., and Otto, A. R. (2014). Retrospective revaluation in sequential decision making: A tale of two systems. *Journal of Experimental Psychology: General*, 143:182–194.

Gupta, A. S., Van Der Meer, M. A., Touretzky, D. S., and Redish, A. D. (2012). Segmentation of spatial experience by hippocampal theta sequences. *Nature Neuroscience*, 15:1032–1039.

Hafner, D., Pasukonis, J., Ba, J., and Lillicrap, T. (2025). Mastering diverse control tasks through world models. *Nature*, 640:1–7.

Hasz, B. M. and Redish, A. D. (2020). Spatial encoding in dorsomedial prefrontal cortex and hippocampus is related during deliberation. *Hippocampus*, 30:1194–1208.

Hok, V., Save, E., Lenck-Santini, P., and Poucet, B. (2005). Coding for spatial goals in the prelimbic/infralimbic area of the rat frontal cortex. *Proceedings of the National Academy of Sciences*, 102:4602–4607.

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

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

Iyer, A., Chandra, S., Sharma, S., and Fiete, I. (2024). Flexible mapping of abstract domains by grid cells via self-supervised extraction and projection of generalized velocity signals. *Advances in Neural Information Processing Systems*, 37:85441–85466.

Jensen, K. T., Hennequin, G., and Mattar, M. G. (2024). A recurrent network model of planning explains hippocampal replay and human behavior. *Nature Neuroscience*, 27:1340–1348.

Johnson, A. and Redish, A. D. (2007). Neural ensembles in CA3 transiently encode paths forward of the animal at a decision point. *Journal of Neuroscience*, 27:12176–12189.

Kay, K., Chung, J. E., Sosa, M., Schor, J. S., Karlsson, M. P., Larkin, M. C., Liu, D. F., and Frank, L. M. (2020). Constant sub-second cycling between representations of possible futures in the hippocampus. *Cell*, 180:552–567.

Kennerley, S. W., Sakai, K., and Rushworth, M. (2004). Organization of action sequences and the role of the pre-SMA. *Journal of Neurophysiology*, 91:978–993.

Keramati, M., Smittenaar, P., Dolan, R. J., and Dayan, P. (2016). Adaptive integration of habits into depth-limited planning defines a habitual-goal–directed spectrum. *Proceedings of the National Academy of Sciences*, 113:12868–12873.

Kropff, E., Carmichael, J. E., Moser, M.-B., and Moser, E. I. (2015). Speed cells in the medial entorhinal cortex. *Nature*, 523:419–424.

Matsuzaka, Y. and Tanji, J. (1996). Changing directions of forthcoming arm movements: neuronal activity in the presupplementary and supplementary motor area of monkey cerebral cortex. *Journal of Neurophysiology*, 76:2327–2342.

Mattar, M. G. and Lengyel, M. (2022). Planning in the brain. *Neuron*, 110:914–934.

McNaughton, B. L., Battaglia, F. P., Jensen, O., Moser, E. I., and Moser, M.-B. (2006). Path integration and the neural basis of the 'cognitive map'. *Nature Reviews Neuroscience*, 7:663–678.

Momennejad, I., Otto, A. R., Daw, N. D., and Norman, K. A. (2018). Offline replay supports planning in human reinforcement learning. *elife*, 7:e32548.

Müller, M. and Wehner, R. (1988). Path integration in desert ants, cataglyphis fortis. *Proceedings of the National Academy of Sciences*, 85:5287–5290.

Nachev, P., Kennard, C., and Husain, M. (2008). Functional role of the supplementary and pre-supplementary motor areas. *Nature Reviews Neuroscience*, 9:856–869.

Neupane, S., Fiete, I., and Jazayeri, M. (2024). Mental navigation in the primate entorhinal cortex. *Nature*, 630:704–711.

Otto, A. R., Gershman, S. J., Markman, A. B., and Daw, N. D. (2013). The curse of planning: dissecting multiple reinforcement-learning systems by taxing the central executive. *Psychological Science*, 24:751–761.

Parron, C. and Save, E. (2004). Evidence for entorhinal and parietal cortices involvement in path integration in the rat. *Experimental Brain Research*, 159:349–359.

Roberts, W., Cruz, C., and Tremblay, J. (2007). Rats take correct novel routes and shortcuts in an enclosed maze. *Journal of Experimental psychology. Animal Behavior Processes*, 33:79–91.

Sato, N., Fujishita, C., and Yamagishi, A. (2018). To take or not to take the shortcut: Flexible spatial behaviour of rats based on cognitive map in a lattice maze. *Behavioural Processes*, 151:39–43.

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

Sohn, J.-W. and Lee, D. (2007). Order-dependent modulation of directional signals in the supplementary and presupplementary motor areas. *Journal of Neuroscience*, 27:13655–13666.

Sutton, R. S. (1991). Dyna, an integrated architecture for learning, planning, and reacting. *ACM Sigart Bulletin*, 2:160–163.

Sutton, R. S. and Barto, A. G. (2018). *Reinforcement Learning: An Introduction*. MIT Press.

Tang, W., Shin, J. D., and Jadhav, S. P. (2021). Multiple time-scales of decision-making in the hippocampus and prefrontal cortex. *Elife*, 10:e66227.

Tolman, E. and Honzik, C. (1930). "insight" in rats. *University of California Publications in Psychology*, 4:215–232.

Tolman, E., Ritchie, B., and Kalish, D. (1946). Studies in spatial learning. I. Orientation and the short-cut. *Journal of Experimental Psychology*, 36:13–24.

Tolman, E. C. (1948). Cognitive maps in rats and men. *Psychological Review*, 55:189–208.

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

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

Wehner, R. and Srinivasan, M. V. (1981). Searching behaviour of desert ants, genus Cataglyphis (Formicidae, Hymenoptera). *Journal of Comparative Physiology*, 142:315–338.

Whittington, J. C., Muller, T. H., Mark, S., Chen, G., Barry, C., Burgess, N., and Behrens, T. E. (2020). The Tolman-Eichenbaum machine: unifying space and relational memory through generalization in the hippocampal formation. *Cell*, 183:1249–1263.

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

Wikenheiser, A. M. and Redish, A. D. (2015). Hippocampal theta sequences reflect current goals. *Nature Neuroscience*, 18:289–294.

Yoon, K., Buice, M. A., Barry, C., Hayman, R., Burgess, N., and Fiete, I. R. (2013). Specific evidence of low-dimensional continuous attractor dynamics in grid cells. *Nature Neuroscience*, 16:1077–1084.

Zanforlin, M. and Poli, G. (1970). *The burrowing rat: a new technique to study place learning and orientation*, volume 82. Societa cooperativa tipografica.