Active Tree Search in Large POMDPs

Domenico Maisto\textsuperscript{1}, Francesco Gregoretti\textsuperscript{1}, Karl Friston\textsuperscript{2}, Giovanni Pezzulo\textsuperscript{3,*}

1) Institute for High Performance Computing and Networking, National Research Council, Via P. Castellino, 111, Naples 80131, Italy

2) The Wellcome Centre for Human Neuroimaging, Institute of Neurology, University College London, London, WC1N 3AR UK

3) Institute of Cognitive Sciences and Technologies, National Research Council, Via San Martino della Battaglia 44, Rome 00185, Italy

* Corresponding author:

Giovanni Pezzulo
ISTC-CNR
Via San Martino della Battaglia 44, 00185 Rome, Italy
Email: giovanni.pezzulo@istc.cnr.it
Phone: +39 6 44595206
Abstract

The ability to plan ahead efficiently is central for both living organisms and artificial systems. Model-based planning and prospection are widely studied in both cognitive neuroscience and artificial intelligence (AI), but from different perspectives—and with different desiderata in mind (biological realism versus scalability) that are difficult to reconcile. Here, we introduce a novel method to plan in large POMDPs—Active Tree Search—that combines the normative character and biological realism of a leading planning theory in neuroscience (Active Inference) and the scalability of Monte-Carlo methods in AI. This unification is beneficial for both approaches. On the one hand, using Monte-Carlo planning permits scaling up the biologically grounded approach of Active Inference to large-scale problems. On the other hand, the theory of Active Inference provides a principled solution to the balance of exploration and exploitation, which is often addressed heuristically in Monte-Carlo methods. Our simulations show that Active Tree Search successfully navigates binary trees that are challenging for sampling-based methods, problems that require adaptive exploration, and the large POMDP problem ‘Rocksample’. Furthermore, we illustrate how Active Tree Search can be used to simulate neurophysiological responses (e.g., in the hippocampus and prefrontal cortex) of humans and other animals that contain large planning problems. These simulations show that Active Tree Search is a principled realisation of neuroscientific and AI theories of planning, which offers both biological realism and scalability.

Keywords: Active Inference, Monte-Carlo Tree Search, model-based planning, POMDP
Introduction

Model-based planning problems have received substantial attention in a variety of disciplines, which include AI, machine learning, robotics, cognitive science and neuroscience. The interdisciplinary exchanges between these disciplines have been numerous [1–6], but yet we still lack a theoretical synthesis that unites their desiderata (e.g., biological realism in neuroscience versus efficiency in AI). Here, we take a step in this direction, by showing that a prevalent theory of model-based control and planning in neuroscience—Active Inference [7]—can be extended in a straightforward manner to address large-scale planning problems, using Monte Carlo methods. Our novel proposed approach—Active Tree Search—bridges the requirements of computational neuroscience and AI, by uniting the normative character and biological realism of Active Inference with the scalability and efficiency of Monte Carlo planning methods.

Active Inference is an increasingly popular framework in computational and systems neuroscience that characterizes perception, action and planning in terms of approximate (variational) Bayesian inference under a generative model [7–10]. Active Inference is related to a family of recent approaches to solving POMDP problems in machine learning and AI, which exploit a general duality between control and inference problems [11], and which include control as inference [12,13], planning as inference [14,15], risk-sensitive and KL control [16]. A peculiarity of Active Inference is that it implements a principled form of model-based planning: it infers a posterior over action sequences (or policies), by considering an expected free energy functional, which effectively balances exploration and exploitation. Previous studies have established that the computations underlying Active Inference are biologically plausible and can reproduce a variety of findings in brain anatomy, neuronal dynamics and behaviour [9,17–19] as well as furnishing sophisticated forms of inference under hierarchical and temporally deep generative models [10,17,19–23].

However, the framework of Active Inference has been developed with cognitive and biological realism in mind, not scalability or efficiency. Its current implementations require the exhaustive evaluation of all the possible policies and hence can only address small-scale POMDP problems. Here, we develop an extension of Active Inference that can address large POMDPs. Our novel Active Tree Search (AcT) algorithm retains the key aspects of Active Inference, such as the use of expected free energy to infer the posterior probability of policies, but relaxes the exhaustive evaluation of all the possible policies, using instead Monte-Carlo planning [24].

Monte-Carlo planning exploits sampling based, look-ahead search over a planning tree, which is expanded during planning, from the root node (i.e., the state where planning starts) to the leaves. It samples rewards obtained by following a given branch of the tree (corresponding to a given course of action) and stores its statistics in the tree nodes. In most practical cases, the planning tree cannot be expanded or sampled exhaustively and hence various heuristic procedures have been proposed to decide what actions to consider next and how to expand the planning tree, in order to converge to an almost-optimal course of action, using bounded resources [25,26]. Interestingly, Active Inference can contextualize these heuristic methods within a normative (and biologically realistic) approach, based on free energy minimization.
Hence, the main contribution of this article is a proof of principle that the novel Active Tree Search method—that combines Active Inference and Monte-Carlo planning—can benefit both approaches. On the one hand, using Monte-Carlo planning permits scaling up the biologically grounded approach of Active Inference to large-scale problems. On the other hand, Active Inference provides a principled approach to dissolve the exploration-exploitation dilemma, which is only addressed heuristically in Monte-Carlo methods.

In the following Sections, we firstly review existing Monte-Carlo and Active Inference methods. We then introduce Active Tree Search formally and validate it using three simulations, showing that it can handle (i) deceptive binary trees (that are challenging for sampling-based methods), (ii) problems that require adaptive exploration, and (iii) large POMDP problems (i.e., Rocksample). Finally, to highlight the potential of Active Tree Search for the study of biological phenomena, we use the scheme to simulate neuronal responses in humans (and other animals) that solve large planning problems.

**Methods: technical background**

*Partially observed Markov decision problems (POMDP)*

Several real world and biological problems can be cast as sequential decisions under uncertainty. Formally, they can be treated as extensions of Markov Decision Process, where the observed action outcomes provide only partial information about the state of the environment; this corresponds to the framework of Partially Observed Markov Decision Process (POMDP) [4].

A POMDP can be defined as a tuple $(S, A, T, Z, O)$ where:

- $S$ denotes the set of the environment states that generate information for the agent to accomplish a task;
- $A$ is the set of actions potentially executable by the agent;
- $T : S \times A \times S \rightarrow [0,1]$, such that $T(s,a,s') = Pr(s'|s,a)$, is the transition probability of being in a state $s'$ after performing the action $a$ from state $s$;
- $Z$ denotes the set of observations.
- $O : S \times A \times Z \rightarrow [0,1]$. such that $O(s',a,z) = Pr(z|a,s')$, is the probability of observing $z \in Z$ by performing an action $a$ in state $s'$;

In Reinforcement Learning (RL) or similar frameworks [5], an additional element of the tuple is necessary: a reward function $R : S \times A \rightarrow R$, where $R(s,a)$ is the reward obtained by taking the action $a$ from a particular state. Here, we dispense with this element (and absorb reward into prior beliefs about all observable outcomes $Z$).
In a POMDP, the state of the environment cannot be observed directly, but can be inferred based on the partial observations that the agent solicits through action. Since the agent’s state information can be noisy or incomplete, it is useful for an agent to consider a probability distribution over the states it could be in. This probability distribution, called (Bayesian) belief, is defined as the posterior 
\[ b_t(s) = Pr(s_t = s|h_t,b_0) \]
given the initial belief \( b_0 \) and a complete sequence or history \( h_t = \{a_0,z_1, ..., z_{t-1}, a_{t-1}, z_t\} \) of past actions and observations. At any time \( t \), it is possible to write down the belief state \( b_t \) as a Bayesian update \( \tau(b_{t-1}, a_{t-1}, z_t) \) of the previous belief state \( b_{t-1} \), given the action \( a_{t-1} \) and the current observation \( z_t \).

Generally, solving a POMDP problem means finding a plan or policy (i.e., sequence of actions), by predicting the situations the agent could encounter in the future, conditioned on the actions it executes. One can specify a policy as a function \( \pi : \mathcal{B} \rightarrow \mathcal{A} \) that associates belief states \( b \in \mathcal{B} \) to actions \( a \in \mathcal{A} \). In Reinforcement Learning (RL), the value function \( V_\pi(b) \) of a policy \( \pi \), evaluated starting from an initial belief \( b \), corresponds to the expected total discounted reward \( V_\pi(b) = \mathbb{E}[\sum_{t=0}^{T} \gamma^t R(s_t, \pi(b_t))] \), where \( \gamma \in [0,1] \) is a discount factor and \( T \) is a finite (or infinite) value if the POMDP problem has a finite (or infinite) time horizon. Following the RL setup, a POMDP plan or policy is optimal when \( \pi^* \) maximises the value function \( V_{\pi^*}(b) \).

**Online POMDP planning**

There are two main approaches to POMDP problems: offline and online. In offline methods [27][28][29][30], the policy is computed before execution by considering every possible belief state. Offline methods achieve good results for small-size scenarios but are not suitable for large POMDP problems. In online methods, there is an alternation between policy construction, whose goal is discovering a good short-path policy (often a single action) for the current belief; and execution, where the selected policy is executed. These methods scale up to large POMDP problems but result in policies that are usually suboptimal, as they are computed based on a subset of beliefs. The two methodologies are complementary and can coexist. For example, online methods can be complemented by initial approximations acquired via some offline algorithm. However, in general, online methods are more widely used given their scalability.
Figure 1. AND-OR tree for a POMDP with 2 observations \(\{z_1, z_2\}\) and 2 actions \(\{a_1, a_2\}\). Triangular (OR) nodes represent belief states, whereas circular (AND) nodes represent actions. The numerical values shown on the edges that stem from OR-nodes represent rewards \(R(b,a)\), whereas the numerical values shown on the edges that stem from AND-nodes represent conditional probabilities \(P(z|b,a)\).

Ross et al [4] established a general scheme for online planning algorithms. In their scheme, every online algorithm can be understood as a procedure in which policy construction is a routine implementing a predefined set of steps: 1) visit, 2) expansion and 3) estimation of an AND-OR tree; with the OR nodes representing beliefs and AND nodes corresponding to actions (Figure 1). The algorithm starts by setting the current belief as the root node of a tree; and then builds new belief nodes generated by action nodes. Every time a new belief node is allocated, it is evaluated, and its value is transmitted up to the belief ancestors—up to the root—to update the value of the corresponding policy (i.e., action sequence corresponding to a specific branch of the tree).

The most popular online-planning approaches are Heuristic Search, Branch-and-Bound pruning and Monte Carlo sampling [31]. In Heuristic Search methods [32][33], a routine explores the belief tree using a heuristic to detect relevant nodes to branch out (frequently, for a single forward step), and successively updates the heuristic value associated to its ancestors (which differs between heuristic search algorithms). However, this procedure can be computationally expensive, hence reducing the effectiveness of the heuristic-based node selection. Branch-and-bound approaches instead rely on a general search technique that constrains the search tree expansion by pruning suboptimal branches [34]. They assign every belief tree node an upper and a lower bound of a quality value function. If a branch leads to a node with an upper bound—that is lower than the lower bound of another node of a different branch—then the first node is labelled as the root of a suboptimal subtree that can be pruned. Finally, Monte Carlo algorithms randomly sample a subset of observations at each time. This constrains the branching expansion of the belief tree and the depth of the search.

While Monte Carlo algorithms follow the same general procedure, they sample outcomes in different ways [35][36]. One of the earliest Monte-Carlo-sampling-based algorithm for solving
POMDP—the Sparse Sampling algorithm of Kearns et al [26]—builds a tree search of fixed depth in one stage (i.e., from the root to the leaves) using a "black-box" simulator (a generative model) for modelling state transitions and simulating reward returns. To improve the performance of the Sparse Sampling algorithm, Kocsis and Szepesvári introduced the Upper Confidence Tree (UCT) algorithm [19], which introduced two essential novelties. First, it uses Monte Carlo Tree Search [37]: a rollout-based Monte Carlo planning that is inspired by game strategy searches but builds the belief tree progressively and iteratively. Second, it selects actions during the planning phase in a stochastic way, rather than by drawing from a uniform distribution (which is consistent with theoretical results on sequential decision making under uncertainty [38]).

**Related works**

UCT [19] is an online decision-making algorithm that works by constructing a tree of simulated histories $h$ by expansion from an initial belief state $b_0$ cast as root, and alternating state and action nodes, eventually drawn by a generative model, i.e., a probabilistic model that statistically describes the POMDP distributions $T$ and $Z$. To select which node (and branch corresponding to some history $h$) to expand next, the algorithm uses the value function $V_v(h)$ and evaluates the expected return from the initial belief $b_0$ following the policy $\pi$ furnished by $h$. A peculiarity of UCT, as of every MCTS algorithm, is the way the value function $V_v(h)$ is calculated: instead of bootstrapping $V_v(h)$ using dynamic programming, it is estimated by Monte Carlo sampling, where multiple stochastic rollouts approximate a mean value. The computed value is then propagated back to each node of the branch and averaged with contributions from other histories branching off from the same node. Concurrently, a visitation count $N(oha)$ is updated, such that $N(h) = \sum_{sa} N(oha)$ is the number of simulations run through the node representing $s$. In UCT, statistics recording node visitations are used effectively: node selection is seen as a Multi-armed Bandit problem for which the optimal choice use the Upper Confidence Bound (UCB) $V_v(h) + c\sqrt{\log N(h)/N(oha)}$ that augments the value function with an exploration term that favours the less-visited nodes [38].

A UCT-based planning algorithm that has received attention in the last decade is POMCP [39]. POMCP can handle POMDPs with large state spaces by adopting MCTS to generate an AND-OR belief tree, where the AND nodes (actions) are selected through the UCB algorithm, and the OR nodes represent a set of sampled states (not a full probability distribution), which are iteratively maintained by a particle filter. Although it can handle problems of considerable size, POMCP has some implicit limitations. By representing the POMDP problem as a belief tree, POMCP needs to visit every potential observation related to a belief state at least once; and furthermore, as it uses the UCB heuristic, its worst-case is quite computationally challenging [40].

DESPOT [41] is another state-of-art MCTS-based algorithm that tries to overcome (at least theoretically) the above limitations by operating on a sparse belief tree generated on a subset of sampled observations. As in POMCP, the nodes of such a reduced tree—called DESPOT tree—that approximates distributions over belief states using particles. An MCTS planning routine progressively constructs the DESPOT tree by iterating the following three stages: a forward search that traverses the tree, until it encounters a leaf node according to the heuristic values (which
includes a pre-computed regularisation term, to prevent overfitting); a leaf initialisation, where a
Monte Carlo sampling estimates the upper and lower bounds of the selected leaf node; and a
backup, that passes back through the path tracked in the forward search and updates the upper and
lower bounds of each visited node, according to the Bellman optimality principle. These three
stages are analogous to the selection, expansion, and backpropagation phases of POMCP. However,
they are iterated until the difference between upper and lower bounds of the belief root state is
sufficiently small (as in a Branch-and-Bound methods).

The above schemes typically require long computation times and are generally not applicable in
real-world scenarios. Consequently, there have been several attempts to develop new approaches to
online POMDP planning, by parallelising extant methods [42][43] or using deep learning to extract
and aggregate relevant information from the environment, in order to speed up and improve the
policy inference [44][45]. More recently, applied research in decision making for autonomous
urban vehicles has engendered novel solutions to the online POMDP and approximate solutions
[46][47][48].

**Active Inference**

Active Inference is a formal framework that integrates the cybernetic concepts of feedback and
error control [16][51] with a Bayesian inferential scheme [17,52,53]. In Active Inference
(henceforth ActI), perception and action (or policy) selection form a closed-loop process, whose
execution can be cast in terms of approximate Bayesian inference [15,54–56], which is rendered
tractable using a variational approximation under the free-energy minimization principle; i.e., a
variational principle of least action [57]. The ActI scheme has been proposed in many variants and
its biological plausibility is under investigation [9][58].

In its essence, ActI is a theory of decision-making under uncertainty (i.e., “[…] a formal treatment
of choice behaviour based on the premise that agents minimize the expected free energy of future
outcomes.” [8]). Formally, ActI can be described as a Partially Observed Markov Decision Process
(POMDP) represented by a tuple \((S, O, U, \gamma, R, P, Q)\) where:

- \(S\) is the set of agent’s hidden states \(s\) by which the agent infers the environmental state.
  Where a sequence of hidden states is denoted by \(\hat{s} = (s_0, ..., s_T)\);

- \(O\) is a finite set of outcomes \(o\), and \(\hat{o} = (o_0, ..., o_T)\);

- \(U\) is a finite set of control states \(u\) executable by the agent to control the environment. A
  sequence of control states \([u_t, ..., u_T]\) is called *policy* and denoted as \(\pi\). Thus, \(\pi =
  [u_t, ..., u_T] = [\pi^{(t)}, ..., \pi^{(T)}]\);
\( \gamma \in \mathbb{R} \), is a supplemental variable denoted as *precision*, introduced to self-tune the control-state selection process adaptively;

\( R(\bar{\delta}, \bar{s}, \bar{u}) \) is a *generative process* that generates probabilistic outcomes from hidden states and actions;

\( P(\bar{\delta}, \bar{s}, \bar{u}, \gamma | \Theta) \) is a *generative model* with parameters \( \Theta = \{A, B, C, D, \alpha, \beta\} \), over outcomes, hidden states, control states and precision;

\( Q(\bar{s}, \bar{u}, \gamma) \) is an approximate posterior distribution over states, control states and parameters, with expectations \( (s^0_0, ..., s^T_0, \pi, \gamma) \).

It is worth noting that there is a fundamental difference between the *generative process* and *generative model*. The *generative process* describes transitions between states of the environment as a function of the agent’s actions and generates observed outcomes. The *generative model* describes the agent’s beliefs about the world and encodes states and policies as expectations. In other words, in ActI, an agent adopts an internal *generative model* to understand its observations and how they may be generated by external, environmental dynamics (*generative process*).

A second important difference is between *control states* (or policies), which are part of the generative model, and *actions*, which are part of the generative process. This formulation permits casting action in terms of beliefs about policies and to convert an optimal control problem into an optimal inference problem; a.k.a., planning as inference [22][59].

**Generative models for Active Inference**

As shown in Fig. 2, the generative model used in ActI includes hidden *states* \( s \) as causes of the observed outcomes \( o \). Hidden states move forward in time under a policy \( \pi \) that depends on the precision \( \gamma \). A series of marginalisations permit writing down the model’s joint density as:

\[
P(\bar{\delta}, \bar{s}, \bar{u}, \gamma | \Theta) = P(\gamma | \Theta)P(\pi | \gamma, \Theta) \prod_{t=0}^{T} P(o_t | s_t, \Theta)P(s_{t+1} | s_t, \pi, \Theta)
\]

where:

\( P(o_t | s_t, \Theta) = A \)
\[ P(s_{t+1}|s_t, \pi, \Theta) = B(u_t = \pi^{(t)}) \quad \text{and} \quad P(s_0|\Theta) = D \]  \hspace{1cm} (2)

\[ P(\pi|\gamma, \Theta) = \sigma(\ln E - \gamma \cdot G_\pi|\Theta) \]

\[ P(\gamma|\Theta) \sim \Gamma(\alpha, \beta) \]

In Equations (2), the matrix \( A \) encodes the likelihood of observations given a hidden state, while \( C \) represents their prior distribution. State transitions are specified by \( B \), the prior distribution of the initial state is given by \( D \), and \( E \) is the prior expectation of each policy. Finally, \( \alpha \) and \( \beta \) correspond to the shape and the rate parameters of the gamma density, which underlies the \( \gamma \)-distribution, respectively.

Figure 2. Graphical model for Active Inference (ActI). See the main text for explanation.

The quantity \( G_\pi \) is a score of the “quality” of a generic policy and can be viewed as the log prior of any given the policy, conditioned on the future state and observations, together with the preferred outcomes (see below for more details).

Active Inference uses a variational approximation to for Bayesian inference [60]. This implies two key things. First, ActI uses an approximate posterior over hidden states and parameters \((\tilde{s}, \tilde{u}, \gamma)\) which can be described in factorised form (known as a mean field approximation):

\[ Q(\tilde{s}, \tilde{u}, \gamma) = Q(\pi)Q(\gamma) \prod_{t=0}^{T} Q(s_t|\pi); \]

\hspace{1cm} (3)

where the sufficient statistics are encoded by the expectations \( \mu = (\tilde{s}^\pi, \pi, \gamma) \), with \( \tilde{s}^\pi = s_0^\pi, \ldots, s_T^\pi \).
Second, ActI performs a minimization of variational free energy of the generative model with respect to the sufficient statistics $\mathbf{\mu}$ of its approximate posterior $Q(\mathbf{s}, \mathbf{u}, \gamma)$. By exploiting some mathematical identities, the variational free energy function can be defined as follows, where $\mathbb{E}_Q[\cdot]$ denotes an expected value under $Q$, $D_{KL}[\cdot || \cdot]$ is the Kullback-Leibler divergence,

\[
F(\bar{o}, \bar{s}^\pi, \mathbf{\pi}, \gamma) = \mathbb{E}_Q[\ln Q(\bar{s}^\pi, \mathbf{\pi}, \gamma) - \ln P(\bar{o}, \bar{s}^\pi, \mathbf{\pi}, \gamma|\Theta)] \\
= D_{KL}[Q(\bar{s}^\pi, \mathbf{\pi}, \gamma)||P(\bar{s}^\pi, \mathbf{\pi}, \gamma|\Theta)] - \ln P(\bar{o}|\Theta) \\
\geq - \ln P(\bar{o}|\Theta)
\]

and, finally, $-\ln P(\bar{o}|\Theta)$ (i.e., the negative logarithm of the model evidence $P(\bar{o}|\Theta)$) is called self information, surprisal or, more simply, surprise. When $Q(\bar{s}^\pi, \mathbf{\pi}, \gamma)$ converges on the posterior $P(\bar{s}^\pi, \mathbf{\pi}, \gamma|\Theta)$, the variational free energy decreases. If they match exactly, and their divergence is zero, free energy becomes surprise. Therefore, one could summarize the variational inference as minimizing free energy to approximate the posterior, while, at the same time, evaluating a band on logarithm of model evidence (often called an evidence bound in machine learning).

Note that the variational approach transforms inference (namely, calculating posterior from prior beliefs) into an optimization problem (namely, finding sufficient statistics $\mathbf{\mu}$ such that the corresponding free energy is minimum). It is possible to demonstrate (see Appendix B of [29]) that such a condition is satisfied when the sufficient statistics at any time $t$ are:

\[
\mathbf{s}_t^\pi \approx \sigma(\ln \mathbf{A} \cdot \mathbf{o}_t + \ln(\mathbf{B}(\pi^{(t-1)}) \cdot \mathbf{s}_{t-1}^\pi)) \\
\mathbf{\pi} = \sigma(\ln \mathbf{E} - \mathbf{\gamma} \cdot \mathbf{G}_\pi) \\
\mathbf{\gamma} = \frac{\alpha}{\beta - \mathbf{G}_\pi}
\]

Here, we use the symbol “$\cdot$” to denote the inner product, defined as $\mathbf{A} \cdot \mathbf{B} = \mathbf{A}^T \mathbf{B}$, where $\mathbf{A}$ and $\mathbf{B}$ are two arbitrary matrices. The first equation defines the expected hidden state and corresponds to the part of Active Inference that implements perception. The second equation derives (as the expected hidden state) from a Boltzmann distribution of the policies’ quality values. The expected value of $\gamma$ is the sensitivity (or inverse temperature parameter) of the distribution: it adjusts the tendency to select a policy with greater or lesser confidence. The last equation tunes the value of the expected precision on the base of the values of the policy quality (in a nonbiological setting, this precision is usually set to 1; especially for policies that only look ahead).
The term $G_\pi$ is the *expected free energy* of the policies and is used to score the quality of a generic policy with respect to the future outcomes and states that are expected under such policies. The expected free energy $G_\pi$ occurs in each of the three above equations. In the first equation, it controls the optimism bias. In the second equation, it determines the choice of policies. Finally, in the third equation, it nuances the confidence an agent has about action selection: with greater differences among the values of $G_\pi$, the precision is greater—and the agent is more confident about what to do next (in a non-biological setting, the policy selected as the policy with the smallest expected free energy).

One can evaluate $G_\pi$ by integrating the expected free energy—under the policy $\pi$—from the current instant $t$ to some horizon $T$:

$$G_\pi = \sum_{\tau=t}^{T} G(\pi, \tau)$$ (6)

where

$$G(\pi, \tau) = F_\tau(\pi)$$

$$= \mathbb{E}_{\tilde{q}}[\ln Q(s_\tau|\pi) - \ln P(o_\tau, s_\tau|\pi, C)]$$ (7)

$$\geq \mathbb{E}_{\tilde{q}}[\ln Q(s_\tau|\pi) - \ln Q(s_\tau|o_\tau, \pi)] - \mathbb{E}_{\tilde{q}}[\ln P(o_\tau|C)]$$

$$= \mathbb{E}_{\tilde{q}}[\ln Q(o_\tau|\pi) - \ln Q(o_\tau|s_\tau, \pi)] - \mathbb{E}_{\tilde{q}}[\ln P(o_\tau|C)]$$

$$= -D_{\text{KL}}[Q(o_\tau|\pi)||P(o_\tau)] - \mathbb{E}_{\tilde{q}}[H[P(o_\tau|s_\tau)]]$$
Here, $\mathbb{E}_\tilde{Q}[\cdot]$ is the expected value under the predicted posterior distribution $\tilde{Q} = Q(o_\tau, s_\tau | \pi) \triangleq P(o_\tau | s_\tau)Q(s_\tau | \pi)$ over hidden states and their outcomes under a specific policy $\pi$. The final identity in Equation (7) provides an interpretation of the expected free energy as a sum of two terms. The former is the Kullback-Leibler divergence between (approximate) posterior and prior over the outcomes; it constitutes the \textit{pragmatic} (or utility-maximizing) component of the quality score, which favours policies that realise outcomes that are expected under the generative model. The latter is the expected entropy under the posterior over hidden states; it represents the \textit{epistemic} component of the quality score, which favours policies that realise unambiguous outcomes [29]. In other words, the former (pragmatic) term represents the \textit{risk} that the anticipated outcomes $Q(o_\tau | \pi)$ diverge from prior preferences $P(o_\tau)$, while the latter (epistemic) minimises \textit{ambiguity}. In summary, risk measures the difference between predicted and preferred outcomes $o_\tau^\pi$ in the future, while ambiguity quantifies to what extent expecting to be in a future state $s_\tau^\pi$ diminishes uncertainty about future outcomes. From a machine learning perspective, this would be equivalent to say that $G_\pi$ embodies a “regularisation” term, which balances between exploitive (pragmatic) and exploratory (epistemic) behaviour.

It is possible to obtain another expression for $G_\pi$ that can be evaluated effectively, by considering the equations for the free energy minimising sufficient statistics above, together with the generative model:

$$G(\pi, \tau) = o_\tau^\pi \cdot (\ln o_\tau^\pi - \ln P(o_\tau)) + s_\tau^\pi \cdot H \tag{8}$$

with

$$s_\tau^\pi = B(u_\tau = \pi^{(\tau)}) \cdot s_\tau$$

$$o_\tau^\pi = A \cdot s_\tau^\pi \tag{9}$$

$$\ln P(o_\tau) = \ln C$$

$$H = -\text{diag}(A \cdot \ln A)$$
where $\ln P(o_t)$ is the log-vector of preferred outcomes, and $H$ is the entropy matrix pertaining to future outcomes.

Crucially, standard implementations of Active Inference assume that during action selection, an agent evaluates all its policies $\pi$, for any possible future state an agent could be in. This means computing the expected free energy $G_{\pi}$ of each policy $\pi$. Once every plausible (i.e., allowable) policy has been scored (and its associated “quality” value is evaluated), the agent uses a Boltzmann distribution to select the best policy from which the next action is sampled. Hence, in Active Inference, an action is the result of an inferential process that scores possible futures and selects the most likely policy, under prior beliefs about the consequences of action.

Active Inference has been used to address a variety of cognitive phenomena, including decision-making [18,61], habitual behaviour, salience and curiosity-driven planning [58,62–64], and in general to develop a process theory for neural computation [65,66]. However, the requirement to evaluate all possible policies exhaustively makes standard approaches to Active Inference intractable in the setting of large POMDPs.

**Active Tree Search**

A straightforward method to overcome the limitations of Active Inference in large POMDPs is to elude the exhaustive evaluation of all allowable policies, using a heuristic procedure [67] [22]. The efficacy of such a proposal needs to be assessed by considering the quality of the approximation and the gains in terms of computational costs and tractability. Here, we evaluate a specific scheme to render Active Inference in large POMDPs tractable, which is inspired by Monte-Carlo Tree Search [24]. For this, we recast each stage of MCTS in terms of Active Inference, thus deriving a novel algorithm that we call Active Tree Search (AcT).

Active Tree Search (AcT) uses random simulations to build a planning tree: an abstract structure where nodes $v_t$ corresponds to beliefs $x_t$ about the state of the particular POMDP problem at hand, where the branches represent possible actions to take to reach future states. Its goal is to estimate the posterior over control states $P(u)$ from which the best action $a_t$ can be sampled. This estimate is obtained through Monte-Carlo simulations [68], which start from the current state $s_t$ (with a related observation $o_t$) and proceed forward, through specific branches of the decision tree, corresponding to a series of histories $h_t = (v_t, u_t, v_{t+1}, u_{t+1}, \cdots, v_T)$. The simulations approximate, statistically, the expected free energy values $G$ of the policies $\pi \equiv (u_t, u_{t+1}, u_{t+2}, \cdots)$; obviously, the larger the number of Monte-Carlo simulations, the more reliable the approximation of $G$. This planning process is iterated until one or more halting conditions are satisfied (see below).

Note that AcT is an algorithm to build a planning tree, not to select actions. Here, we assume that after the planning tree has been built, the agent selects an action by sampling from the distribution of control states inferred at the root node; and executes it. At this point, the agent makes a transition to a new state and receives a new observation—and can start planning again.
The four stages of Active Tree Search

Algorithm 1 reports the pseudocode of Active Tree Search as function of the parameters $\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D}$ usually used in Active Inference [8][9], and of the discount factor $\delta$, commonly used in Monte-Carlo Tree Search [25][69]. The Active Tree Search algorithm comprises four successive stages—Variational Inference, Expansion, Prediction, and Path Integration—applied iteratively at each time step $t$ (Figure 3). In the following, we examine each stage in detail.

![Diagram of the four stages of Active Tree Search](image)

Figure 3. The four stages of the Active Tree Search algorithm.

First stage: Variational Inference

The goal of the first (variational inference) stage is to select the next non-terminal leaf node $v_\tau$ of the tree to expand. From the root—and recursively until reaching an expandable node of the planning tree—this stage samples an action over a Boltzmann distribution $\sigma(\varepsilon \ln \mathbf{E} - \gamma_\tau \cdot G(\pi_\tau, v_\tau))$ that depends on three terms: 1) the expected free energy $G(\pi_\tau, v_\tau) = (\mathbf{A} \cdot \mathbf{x}_\tau) \cdot (\ln \mathbf{A} \cdot \mathbf{x}_\tau - \ln \mathbf{C}) + \mathbf{x}_\tau \cdot \left( \sum_i \mathbf{A}_{ij} \ln \mathbf{A}_{ij} \right)_j$ of the policy $\pi_\tau$ assembled so far, 2) the precision $\gamma_\tau$ computed at each depth of the tree visit, and 3) the policy prior beliefs $\mathbf{E}$ [70].

Taken together, these three terms define the estimated quality of a policy and consider 1) the divergence between the a-priori beliefs $\mathbf{C}$ and the expected outcomes $\mathbf{A} \cdot \mathbf{x}_\tau$, and expected entropy of observations (respectively, first and second terms of $G(\pi_\tau, v_\tau)$), 2) a modulation of the policy quality distribution that controls the stochasticity of action selection, and 3) a confidence bound that regulates exploration.

The latter term ($\mathbf{E}$) is modulated by a factor $\varepsilon$ named exploration factor and is closely related to the UCB1 algorithm for the Multi-armed Bandits [38][71]. It takes the form of probabilistic distribution $\mathbf{E} \sim \sqrt{2 \ln N(v')/N(v''')}$, where $v'$ denotes a child node, $N(v')$ denotes the number of visits of $v'$, and $N(v)$ denotes the number of visits of the parent node $v$. Given this definition of $\mathbf{E}$, the probability of every child node decreases if it is visited frequently (i.e., with high $N(v')$) and increases when its number of visits is sufficiently lower than that of the other children (i.e., a large ratio
\[ \ln N(v)/N(v') \]. Therefore, analogous to the UCT algorithm [25][69], the Variational Inference stage may select every node with a probability different from zero, and which increases in time for less visited states. In active inference, this extra parameter is usually read as encoding prior beliefs about policies that have become habitual (i.e., habitual priors that are combined with empirical priors furnished by the expected free energy).

**Second stage: Expansion**

The goal of the second (expansion) stage is to expand the non-terminal leaf node \( v_\tau \) selected during the former (variational inference) stage. Expansion of a leaf node \( v_\tau \) corresponds to instantiating a new child node \( v' \) by implementing a random action \( u' \) among those previously unused. Each of these children stands for a future state \( x' \) that an agent can visit, according to the transitions defined in matrix \( B \). By adopting a Bayesian terminology, expanding could mean defining new predictable events over the space of future policies, thereby expanding the horizon of possible events. Note that both the first (variational inference) and the second (expansion) stages return the same output—a node—but the former stage selects a node whereas the latter creates a node. It would be also possible to merge these two stages into a unique stage, analogous to TreePolicy [37] in MCTS. In Bayesian statistics this is not unlike the procedures implicit in nonparametric Bayes, based upon stick-breaking processes that allow for an expansion of latent states [72].

**Third stage: Prediction**

The goal of the third (prediction) stage is to roll out one or more random (or default [37]) policies starting from the node returned by the second (expansion) stage, and evaluate the nodes it traverses. Specifically, a policy (or a series of policies) \( \pi_\Delta \) is randomly generated and “rolled out” until it reaches a terminal node of the tree (alternative schemes involve a predefined horizon time determined by the parameter \( \rho \) or some computational bound). Then, the expected free energy \( G(\pi_\Delta, v_\tau) \) of the policy is computed, as in the Variational Inference stage. The resulting \( G \) value becomes the value of the tree node \( v_\tau \) from which the roll out originated; but it is discounted by a factor that depends on a parameter \( \delta \) and on the depth of the tree node \( v_\tau \). Note that while we use random policies, there are alternative ways to generate default policies that employ (for example) prior knowledge.

**Fourth stage: Path Integration**

The goal of the fourth (path integration) stage is to adjust the \( G \) values of the tree nodes up to the root node \( v_t \) by considering the new values obtained during the third (prediction) stage. The value accumulated during the prediction stage is “backpropagated” recursively through the nodes \( v_\tau, \ldots, v_t \) of the planning tree selected during the TreePolicy procedure, to update their statistics: their quality \( G \) and the number of visits \( N \). Such updating is the statistical analogue of “path-integration” formulations in Active Inference [8], where one sums up the expected free energy at each time step in the future.
The four stages are repeated iteratively, until a criterion is met, furnishing estimates of the $G$ values of a subset of the tree nodes.

Algorithm 1: Active Tree Search algorithm with subroutines

function $\text{AcT}(A, B, C, D, \delta)$

$t \leftarrow 0$

while halting conditions are not satisfied do

$x_t \leftarrow$ update expected state belief $x_{t-1}$ by using $s_t, o_t, a_{t-1}, A, B, D$

create a node $v(x_t, u_{t-1})$

while computational bounds are not reached do

$v_t \leftarrow \text{TreePolicy}(v, B)$

$G_{\Delta} \leftarrow \text{Prediction}(v_t, A, B, C, \delta)$

PathIntegration($v_t, G_{\Delta}$)

End

$(s_{t+1}, o_{t+1}, x_t, a_t) \leftarrow$ extract information saved in $v$

$t \leftarrow t + 1$

end

end

function $\text{TreePolicy}(v)$

while $v$ is nonterminal do

if $v$ not fully expanded then

return Expansion($v, B$)

else

$v \leftarrow \text{VariationalInference}(v)$

end

end

return $v_t \leftarrow v$

end

function $\text{Expansion}(v, B)$

draw randomly an unused action $u'$ on $v$

for the parent $v$, generate a new child $v'(x', u')$ with $x' = B(u') \cdot x$

return $v'$

end

function $\text{VariationalInference}(v)$

build the distribution $E$ via the probability mass function $\sqrt{\frac{2 \ln N(v)}{N(v')}}$

$u' \sim \sigma(\epsilon \ln E - \gamma \cdot G_{\Delta}(v'))$

return $v'(u')$


function Prediction\( (v_\tau, A, B, C, \delta) \)

\[ \text{while stopping criteria are not verified do} \]

append control states uniformly generated at random to a policy \( \pi_\Delta \equiv (u'_1, u'_2, ...) \)

end

evaluate the expected free energy \( G(\pi_\Delta, v_\tau) \) through \( A, B, C \)

\[ \text{return } G_\Delta = \delta^{\text{depth}(v_\tau)} \cdot G(\pi_\Delta, v_\tau) \]

end

function PathIntegration\( (v_\tau, G_\Delta) \)

\[ \text{while } v_\tau \text{ is not } v \text{ do} \]

\[ N(v_\tau) \leftarrow N(v_\tau) + 1 \]

\[ G(v_\tau) \leftarrow G(v_\tau) + \frac{1}{N(v_\tau)} (G_\Delta - G(v_\tau)) \]

\[ v_\tau \leftarrow \text{parent of } v_\tau \]

end

Computational resources required by Active Tree Search

Most on-line planning algorithms employ a belief tree like that shown in Figure 1, to encode the POMDP problem in a manageable form. The on-line algorithms implement multiple lookahead visits on the tree to plan the next action to execute. A belief tree of depth \( D \) contains \( O(|U|^D |Z|^D) \) nodes where \( |U| \) and \( |Z| \) are the cardinalities of the action and observation set, respectively; as a consequence, the tree size influences the performance of every algorithm relying on belief tree visits in their planning phase. The famous POMCP algorithm [39], for instance, the ancestor of the algorithms based on the tree visit sampling, is prone to this complexity. The R-DESPOT algorithm [73] generates a subtree of the belief tree of size \( O(|U|^D K) \) that encompasses the executions of all policies by \( K \) abstract simulations called scenarios. In contrast, AcT works on trees with \( O(|U|^D) \) nodes—of considerably less complexity—but uses lossless probabilistic representations for beliefs and observations. This requires greater memory resources and longer updating operations during the run. On the other hand, as noted in [74], POMCP and R-DESPOT, that approximate belief states and outcomes with a particle filter, suffer problems with large observation spaces because beliefs could collapse into single particles, causing convergence to suboptimal policies.

Experimental Results

We tested the Active Tree Search on a series of problems. First, we considered two problems that are often discussed in the literature to highlight the fact that UCT tends to be stuck in \textit{traps}, a notion introduced in adversarial games strategy search [75] to indicate those states of a game whose
instantaneous utility is deceptive, with respect to the future outcomes that they portend. Secondly, we applied Active Tree Search to a POMDP benchmark problem, the RockSample [32], to evaluate the effectiveness and scalability of the algorithm, as the problem complexity increases.

**Active Tree Search avoids traps in deceptive binary trees**

Ramanujan et al. [76] noted that the performance of the UCT algorithm is limited in games where the best proximal decisions do not necessarily correspond to winning strategies [13]. This is the case, for example, in Chess, where exhaustive search (e.g., minimax) yields better performance than sampling-based approaches. This property is due to the presence of particular game configurations in which an unfortunate move leads unavoidably to a defeat. These game states are called *traps* because, as noticed above, their instantaneous utility is deceptive with respect to the future outcomes that they lead to. Being able to escape from traps is a crucial feature of successful planning algorithms.

Coquelin & Munos [40] introduced an example challenging problem for sampling-based planning algorithms. This problem has just two actions, $2^D + 1$ states parameterised by $D \in \mathbb{N}$, and deterministic rewards. At each time step $d$, one can get a reward of $(D - d)/D$ by choosing the action 2: alternatively, one can move forward by choosing action 1. At time $D - 1$, action 1 corresponds to an absorbing state with maximum reward 1 and action 2 to another absorbing state with reward 0. Intuitively, the state space of the problem can be described as a binary tree of depth $D$ (Figure 4). The optimal plan involves always selecting action 1, to move along all the levels $d$ of the branch and reach the final (maximum) reward. Finding this solution is challenging for sampling methods, as the suboptimal action 2 is much more rewarding in the proximity of the root—and this immediate reward influences planning following the first moves.

![Figure 4. A binary tree representing the state space of a challenging problem for sampling-based planning methods. From the root (left node) toward the deepest level $D$, action 2 at each level leads to a deceptive leaf node with reward $(D - d)/D$. The optimal policy involves always selecting action 1, which yields reward 1. Figure is adapted from [48].](image)

Coquelin & Munos proved by induction that UCT has a hyper-exponential dependency concerning the depth $D$ of the binary tree and, by considering the worst case, it takes $\Omega(\exp(...(\exp(1))...))$—composed by $D - 1$ exponential functions to get the reward 1. To check whether AcT suffers the
same sort of limitation, we compared UCT, AcT and a reduced version of Act, called FE, which does not use the policy prior beliefs $E$ during the exploration stage (or, analogously, with $\varepsilon = 0$). To render this problem suitable for AcT we reformulated it as an MDP problem with $2D + 1$ hidden states with a corresponding set of observations (consequently $A$ is diagonal), action “1” and “2” to move from a state to another (reported in $B$) and a vector $C$ where the rewards are spread over observations by a probabilistic distribution encoding preferences.

We used a fixed number of simulations (5000) for every algorithm tested and considered problem cases with $D = 10$ and 100. We used a discount factor of $\delta = 0.9$. The results are shown in Figure 5, by plotting—for each algorithm—the state occupied as a function of simulations, the occupancy frequency, and the failure rate defined as the relative difference between the depth $d$ of the visited state and $D$. Our results show that UCT experiences problems by $D = 10$, and selects the first deceptive states. Conversely, both algorithms using Active-Inference reached the deepest state, despite performance decreases as the depth increases. FE displays a more pronounced greedy behaviour, while AcT keeps exploring, due to the prior distribution $E$. 

\[ D=10 \]
Figure 5. Experimental results in the deceptive binary tree of [40] for the depths $D = 10, 100$ (top and bottom panels, respectively). Each panel plots state occupation (top), occupation frequency (middle), and failure rate.

Active Tree Search evinces an adaptive level of exploration when finding the global maximum of a non-Lipschitzian function

The above problem can be considered illustrative of a whole class of MDP domains on which sampling algorithms manifest shortcomings [77]. These problems are all characterised by the lack of smoothness of the objective or value function, where the notion of “smoothness” corresponds to a well-behaved analytic or continuous value function. Formally, this condition can be expressed through the Lipschitz continuity, according to which a value function $V(s)$ defined over the state-space $S$ is $M$-Lipschitz continuous if $\forall s_1, s_2 \in S, |V(s_1) - V(s_2)| \leq M \|k(s_1) - k(s_2)\|$, where $M$ is a constant and $k(\cdot)$ is a mapping from $S$ to some normed vector space [78]. The challenge for any optimisation scheme is to find the global maximum of a non-Lipschitzian function. The function:

$$g(x) = \begin{cases} 
0.5 + 0.5|\sin \frac{1}{x^5}|, & 0 < x < 0.5 \\
0.35 + 0.5|\sin \frac{1}{x^5}|, & 0.5 \leq x \leq 1 
\end{cases} \quad (10)$$
introduced as a test in [49], has two distinct behaviours over its domain (see panel A in Figure 6). In the (left) interval [0,0.5], there exist numerous global optima, but its functional form is quite rough, whereby in the (right) interval [0.5,1] the function is smooth, but the extrema are suboptimal. In this case, an effective search algorithm should explore every region of the domain. However, because UCT has the aforementioned limitation, it visits the smoother (suboptimal) half of the function more frequently.

As for the binary-tree test used before, we cast this optimization problem as MDP problem: each state represents some interval \([a, b]\) within this unit square, with the starting state representing \([0, 1]\). We assume that there are two available actions at each state, the former resulting in a transition to the new state \([a, (a+b)/2]\) and the second resulting in a transition to \([(a+b)/2, b]\). For example, at the starting state the agent has the choice between a "left" action to make a transition to the state \([0,0.5]\) and a "right" action to make a transition to the state \([0.5,1]\). After it selects the left action, it has a choice between another "left" action to make a transition to \([0,0.25]\) or a "right" action to \([0.25,0.5]\); and so on. As a consequence, with increasingly deeper planning trees, the agent explores more fine-grained intervals. An efficient planner should visit the left interval \([0,0.5]\) extensively and deeply (i.e., approach zero), as it encompasses many maxima.

Figure 6. The function “g” used to test AcT and UCT algorithms in relation to their efficacy in problems with a rough landscape. A) “g” function defined in \([0,1]\) is Lipschitz-continuous for values larger than 0.5, yet it is not otherwise. B) and C) Scatter plots of the values encoded by the leaf states reached respectively by UCT and AcT in 1000 iterations with the histograms \(v(g)\) and \(v(x)\) of their codomain and domain values. D) The same plot for the AcT case with exploration factor \(\epsilon = 10\).
The state-space $S$ can be represented as a binary tree whose depth is constrained by a trade-off set by the condition $b - a < 10^{-3}$. Each state corresponds to one observation, hence resulting in a diagonal matrix $A$ matrix. The a priori distribution $C$ is empirically computed by considering the value of $g(x)$ in the midpoint of the domain interval encoded by the states. The discount factor $\delta$ was set to 0.9.

After running the algorithms for 1000 rounds, we found that UCT fails to explore the whole domain of $g$ thoroughly, but mostly remains stuck in a state corresponding to the maximum obtained for an $x$ value around 0.2; see Figure 6C. Conversely, AcT goes further down the tree search and, subsequently, finds maxima closer to 0.

This optimization problem illustrates the effects of the exploration factor $\epsilon$ on algorithm performance. Figure 6B shows the results of AcT for $\epsilon = 1$, which are similar to those achieved by the FE variant previously used in the deceptive binary tree example. This is because, in this particular problem, the term used to sample the actions (related to the $E$ and controlled by $\epsilon$) is numerically much smaller than the one related to the policy value $G$. Figure 6D shows instead the results of AcT for $\epsilon = 10$. In this case, the $v(g)$ distribution shows that AcT has visited the most of $g$ codomain, choosing to search also unrewarding branches of the state-space-encoding binary tree, even if this entailed a reduction of performance (as seen by noting that the mode of $v(x)$ is far from 0).

**Active Tree Search in large POMDP problems: the case of Rocksample**

*Rocksample($n,k$) [32]* is a well-known benchmark problem used to assess POMDP solvers and their scalability. It simulates a rover whose task is collecting samples of $k$ scientifically valuable rocks, deployed on an $n \times n$ alien soil grid—and then leaving the area. Samples come in two varieties: valuable or invaluable. The rover earns a reward for each valuable sample it collects and a penalty for each invaluable sample. The rover knows the locations of the rocks but can only evaluate whether they are valuable or not via a long-range sensor, whose measures are affected by an error, which increases exponentially with the distance between the rover and the rock examined. We considered two variants of this problem, with $(n,k)$ equal to $(7,8)$ and to $(11,11)$.

Representing this problem in a format suitable for Active Inference is straightforward. In principle, the state space $S$ of the problem can be factorised to reduce the number of states [79], but we decided to retain both variants without factorised representations, to leverage the difficulty of the problem. Therefore, the cardinality of the state space is $|S| = n^2 \times 2^k + 1$ (12,544 in the case (7,8) and 247,808 in (11,11)), necessary to encode every possible combination of locations and the scientific value of the rocks plus an additional “exit” state. This cardinality is needed to define the initial belief state $D$ and the transition state matrix $B$, which is also conditioned on the control state (actions) $a \in U$ that the agent can make.

The set $U$ contains the four actions (north, south, east, west) that the agent uses to move around the square, $k$ actions that the agent uses to evaluate the rocks remotely (one action for each rock), and a sampling action to collect a rock sample. Observations are factorised in three factors, which relate
to the positions on the grid, the scientific qualities of the rocks (“good” or “bad”), and their associated rewards, respectively. Accordingly, the likelihood $A$ is decomposed in three factors, each one encoded as a cubic matrix (generally as a tensor when the state space, in turn, is subdivided into factors), where the first component represents the observations, the second the states, and the last the actions. Introducing a dependency of $A$ on the actions is uncommon in active inference but crucial in many POMDP problems, included Rocksampler($n, k$). This is because the observation one gets by sampling a rock (with an action $k$) is a function of the distance from the rock; encoding this contingency would require a huge number of states if one does not express $A$ as a function of actions. Reward contingencies expressed in $A$ are action-dependent, too, as the agent obtains a “good” observation when it samples a good rock (and a “bad” observation otherwise) — and when it exits the game. Finally, $C$ encodes preferred observations, and comprises three modalities: in the first two, observations are uniformly preferred, while in the last they are drawn from a Bernoulli distribution with a success probability almost equal to 1.

In the experimental setup, parameters were fixed at the same values for both cases. We chose 2000 as the maximum number of time steps; the discount factor $\delta$ was set to 0.9 and the discount horizon $\rho$ equal to 0.01: so that the length of the rolled-out policies $|\pi_D|$ is about 13 steps, a threshold computed by considering that $\delta^{|\pi_D|} < \rho$. Because AcT is a stochastic algorithm, statistical results have to be gathered for each case; here, by executing 50 runs with different seeds from a pseudorandom number generator. For each run, the worst scenario was considered: every rock is “good”, so that the rover needs to sample all the $k$ rocks to accomplish the task.

Figure 7 shows the model performance, here expressed as total undiscounted reward $R_t = \sum_{k=t} \tau_k$ (left) and the total discounted reward $R_t^\delta = \sum_{k=t} \delta^{k-t} \tau_k$ (right). Both values are averaged over the runs and as a function of the simulations cumulated at every time step. AcT obtains total undiscounted reward values of $87.27 \pm 1.49$ out of 90, and of $105.15 \pm 4.68$ out of 120, in Rocksampler(7,8) and Rocksampler(11,11), respectively; see Figure 7, left. AcT obtains total discounted reward values of $13.23 \pm 2.11$ and of $12.44 \pm 2.95$, in Rocksampler(7,8) and Rocksampler(11,11), respectively; see Figure 7, right. These are pleasing results in relation to alternative schemes.

**Rocksampler(7,8)**
Figure 7. Total undiscounted reward and total discounted reward plots as a function of the simulations; all results are averaged over 50 runs. RockSample(7,8) and RockSample(11,11) results are shown respectively at the top e at the bottom of the figure.

It is worth noting that the performance of AcT is markedly better when one considers the average undiscounted reward (about 2% error for (7,8) and 4% for (11,11), see Figure 7, left) rather than average discounted reward (about 16% error for (7,8) and 24%, for (11,11), see Figure 7, right). This is not surprising, since—in contrast to most approaches to solving POMDP problems—AcT does not natively consider any discount factor. In both cases, AcT scales up gracefully with the size of the problem. This is evident if one considers that the agent's performance has the same trend in both smaller (7,8) and larger (11,11) problems. These simulations provide proof of principle that active inference can be scaled to deep planning problems of a real-world nature. In the next section, we consider the real-world implementation of active inference from the perspective of neuronal dynamics.

Simulated neuronal dynamics of Active Tree Search

This section illustrates the usage of Active Tree Search to simulate behavioural and neurophysiological responses during human planning. To exemplify this, we apply Active Tree Search to “Tiger”: a popular POMDP problem introduced in [80] to illustrate the importance of epistemic, information-gathering actions (that aim to acquire information to reduce uncertainty) during planning. In the Tiger problem, an agent stands in front of two doors and has to decide which one to open. The agent knows that one of the two doors hides a treasure, whereas the other conceals a tiger. If the agent opens the door with the treasure it receives a reward, but if it opens the door with the tiger it receives a penalty. The agent does now know where the tiger is but can resolve uncertainty by "listening for animal's noises" (which costs a small penalty).

The domain of this problem is usually represented as a POMDP with 2 states (tiger behind left or right doors), 3 actions (to open the two doors or listen) and 2 observations (reward or penalty). To ensure compatibility with previous Active Inference studies, here we recast the problem as a T-maze with 8 states, 4 actions and 16 observations [8]. The 8 states result from the multiplication of
4 locations times 2 hidden contexts. The 4 locations correspond to the centre (i.e., start location) the left and the right arms (analogous to the two doors, with treasure and tiger, respectively) and the lower arm (analogous to a listening location, where a cue can be found that discriminates the tiger location). The 2 hidden contexts correspond to the 2 possible reward locations (i.e., reward at the left or the right arm, respectively). The 4 actions move the agent deterministically to each of the 4 locations (but cannot change hidden context). Finally, the 16 observations result from the multiplication of 4 positional observations (that correspond 1-to-1 to the 4 locations) by 4 outcomes (i.e., reward, penalty, cue for the tiger at left, and cue for the tiger at right) that are obtained in different states, see below.

The agent's generative model is specified by the matrices $A, B$, and the vectors $C$ and $D$. The (likelihood) matrix $A$ is a probabilistic mapping from states to outcomes. It specifies that the centre location provides an ambiguous cue (i.e., a cue that is identical if the agent is in either of the 2 hidden contexts and hence does not provide any information about the reward location). Furthermore, it specifies that the lower arm provides a disambiguating cue—that discloses which of the 2 hidden contexts the agent is in (and hence the reward location). Finally, the likelihood specifies that if the agent is in the first hidden context ("reward at the right arm"), the right and the left arms provide a reward and a penalty, respectively, with probability $p = 0.90$. On the contrary, if the agent is in the second hidden context ("reward at the left"), the right and the left arms provide a penalty and a reward, respectively, with probability $p = 0.90$.

The $B(u)$ (transition) matrices define 4 action-specific stochastic transitions between states. These move the agent deterministically to each of the 4 locations (but cannot change its hidden context). However, there is a peculiarity: given that the task ends when the agent is in one of the upper arms (i.e., opens one of the two doors), we consider the corresponding hidden states as absorbing states that cannot be left, whatever the action.

The vector $C$ encodes the probability mass over preferred outcomes. It is determined by applying the softmax function over a utility vector having 2 and $-2$ respectively for rewarding and penalty outcomes and zeros otherwise. Finally, the vector $D$ represents the agent's belief about its initial state. The agent knows that it starts from the centre location, but—crucially—it does not know in which of the 2 hidden contexts it is in (i.e., it does not know the reward location, left or right). This is why it is optimal for the agent to go to the lower arm to solicit a cue (i.e., "listen") that disambiguates the hidden context, before deciding whether to visit the left or right arms. As in the previous simulations, we set $\delta = 0.9$.

Figure 8 illustrates the results of Active Tree Search simulations, in which the reward is in the right arm. The upper panel shows the agent's state-belief distribution over time, together with the true states (cyan circles). At the first epoch, the agent knows that it is in the centre location but does not know its current context. This is evident when considering that in the first column of the upper panel, the belief distribution spans states 1 and 2 (i.e., centre location in the first and second hidden contexts). Successively, the agent selects an action to visit the lower arm, to collect a cue; and it discovers that the hidden context is the first (reward at the right arm). This is evident in the second column of the upper panel, where the belief distribution is concentrated in state 7 (i.e., lower arm,
first hidden context). Note that the agent decided to explore the lower arm to secure a cue, instead of guessing which of the two arms is rewarding. Although this entails a cost (a delay in reaching the reward location later), this “epistemic behaviour” ensures the selection of the rewarding arm at the next epoch, see the third column of the upper panel. This epistemic behaviour emerges automatically in Active Inference [7], because policy selection considers the expected reduction of uncertainty along with utility maximization (see Equation (7) of the section “Active Inference”).

Figure 8. Simulated behavioural and neuronal responses of Active Tree Search in the “Tiger” problem, when the tiger is behind the left door. The first (top) panel shows the belief distribution of the eight hidden states (4 locations times 2 contexts) of the problem. The second panel illustrates the search trees generated by AcT in the three epochs of the simulation. The third panel shows simulated neurophysiological responses associated to the planning problem. These are shown as
firing rates of 24 single cells that encode hidden states (There are 8 hidden states for each of the 3 epochs), in a raster plot format. The fourth panel plots the firing rates of two units encoding the right arm (solid line) and left arm (dashed line) on the third epoch. These are the states that will be finally selected (right arm) and unselected (left arm). See the main text for explanation.

The second panel of Figure 8 shows the “search trees” that AcT generates during each epoch. The left picture of the second panel shows the search tree generated during the first epoch, where the thickness of the edges connecting levels reports the probabilities of going to one of the 4 locations. The preferred plan at depth one is to make an “epistemic” move to visit the lower arm. At depth two, the two preferred actions are to visit the lower arm and (to a lesser extent) the centre location. The reason for this is that the tree search has not received yet any observation from the generative process and therefore has no information about the tiger location—and hence it avoids states that include potentially penalties.

The centre picture of the second panel shows the search tree generated during the second epoch, after the agent has visited the lower arm and has observed an informative cue. At this point, the agent constructs a new search tree where the plan to reach the right arm has a high probability. The choice remains the same at the last epoch (see the right picture of the third panel) and the agent collects the rewarding outcome.

The third panel of Figure 8 illustrates simulated neurophysiological responses during the simulation shown in the first two panels. We assume that outcomes are sampled every 250 ms: a timescale compatible to hippocampal theta cycles, where place cells that represent current and prospective locations can be decoded [81–83]. The figure illustrates a raster plot of simulated neuronal activity for units encoding hidden states. The image is organized as a matrix, with 24 rows/neurons (4 locations times 2 contexts times 3 epochs, corresponding to the planning horizon) and a number of columns equal to the number of rollouts—as implemented by AcT during the three decision epochs (16 in this simulation). In other words, the presence of 24 rows/neurons indicates that at each epoch, the agent represents both its current epoch and the other two epochs (e.g., during the first epoch, it represents also the next two epochs; during the second epoch, it represents the previous and the next epochs). Furthermore, there are separate neuronal populations that encode the same hidden states at different epochs (e.g., the first hidden state at the first and second epochs correspond to the 1st and the 9th rows/neurons, respectively). In effect, this endows the agent with a form of working memory that is both predictive and postdictive.

The rows and the columns of the third panel can be grouped to cluster the matrix in $3 \times 3$ blocks of length 8 and 16, respectively. In this format, the elements shown in the main diagonal of the block matrix are beliefs about the present and correspond to the hidden states shown in the first panel. The elements shown in the upper and lower diagonal blocks correspond to (retrospective and predicted) beliefs about the past and the future, respectively. Note that the elements under the main diagonal correspond to the beliefs shown in the search trees of the second panel.

The fourth panel of Figure 8 reports the simulated firing rates of two selected units, which correspond to the states representing the left (dashed line) and the right arm (solid line) during the third epoch. These are the states that will be visited (right arm) and not visited (left arm) during the
third epoch. Initially (first column of the fourth panel) both units have the same firing rates, representing the fact that the agent is uncertain about the state that it will visit next. However, this uncertainty is resolved during the second epoch and confirmed during the third (second and third columns, respectively). It is evident from this panel that expectations about future visitations (corresponding to the firing rates of the two units) diverge during the epochs, following a stepwise evidence accumulation [65].

These simulations exemplify the possibility of establishing a mapping between algorithmic methods of AcT and neuronal processes that are relevant for neuroscience. For example, the neurophysiological responses shown in the last two panels of Figure 8 exemplify prospective (and retrospective) representations of states that have been consistently reported in rodents [84–86], monkeys [87,88] and humans [89,90] engaged in sequential decision or navigation tasks. While computational modelling is widely used in neuroscience, there is still a paucity of methods that can both address large-scale problems that are relevant for AI and generate predictions that are relevant for neuroscience. Some recent studies using powerful deep learning [91–93] and Bayesian methods [94] are already bridging this gap, but they mostly address some specific domains, such as visual perception and motor control. Addressing tasks such as Rocksample or Tiger requires instead designing complete agent architectures, as exemplified by AcT (this paper) and deep reinforcement learning models [95]. AcT and deep reinforcement learning models appeal to different principles (e.g., free energy minimization versus reward maximization; inference versus trial-and-error learning) to design agent architectures that solve complex tasks, hence pointing to different views of neuronal dynamics. Comparing the empirical validity of these assumptions side-by-side is an important objective for future research.

Discussion

Model-based planning is a widely interdisciplinary topic. However, a synthesis of ideas and methods from disciplines as diverse as AI, machine learning, cognitive and computational neuroscience has been challenging to achieve, given their different focus (e.g., scalability and efficiency in AI, biological realism in neuroscience).

Here, we offer a significant step in this direction, by extending a prominent neurobiological theory of model-based control and planning—Active Inference—to scale it up to large-scale POMDP problems. This extension exploits Monte-Carlo planning to elude the extensive evaluation of action policies that is often countenanced in Active Inference. The theoretical synthesis of Active Inference and Monte-Carlo planning—called Active Tree Search—has benefits for both. On the one hand, augmenting Active Inference with Monte-Carlo methods permits realizing a novel and appealing process model for approximate planning, which renders it scalable and potentially useful for explaining bounded forms of cognition and reasoning [56,96–98]. On the other hand, Active Inference provides a theoretically motivated and biologically grounded framework to balance exploration and exploitation, which contextualizes heuristic methods widely adopted in Monte-Carlo planning, and which obtains remarkable results in challenging POMDP problems.

We validated Active Tree Search in three simulative studies. The results of the first study show that AcT addresses successfully deceptive binary trees that challenge most sampling-based planning
methods, as it requires an accurate balance of exploration and exploitation. In AcT, the balance of exploration and exploitation depends implicitly on a single free energy functional that is used for policy evaluation. The results of the second study confirm the adaptivity of the exploration strategy used in AcT. They suggest that AcT can resolve challenging problems whose value functions are not smooth, and which are therefore challenging to explore systematically. The results of the third study show that AcT can successfully address large POMDP problems (here, Rocksmaple) and its performance, as measured by average undiscounted reward, is very high. An objective for future research is to compare AcT with benchmark planning algorithms, such as POMCP, AR-DESPOT (the anytime approximation of R-DESPOT) and SARSOP [30], which is, to date, one of the most effective POMDP offline planning algorithms [99]. However, it is worth noting that while AcT optimizes average undiscounted reward, all the other algorithms are typically evaluated based on average discounted rewards [9], hence making their comparison more challenging.

Finally, we used Active Tree Search to simulate neuronal responses during a representative planning task. This simulation illustrates the ability to map the algorithmic-level planning dynamics of AcT to neuronal-level representations putatively found in the hippocampus (and other areas, such as prefrontal cortex) of animals that solve equivalent tasks [81–83]. Indeed, Active Inference originated in computational and systems neuroscience, with the aim to characterize brain processes from a normative perspective. AcT retains the neurobiological motivation of Active Inference, while expanding it to large-scale problems that could not be confronted by previous implementations. The possibility to use AcT to both address large scale planning problems and to explain neuronal activity can help establish a much-needed bridge between AI and computational neuroscience.

Declaration of conflict-of-interest

The authors have no conflict-of-interest to declare.

Acknowledgements

KJF was funded by a Wellcome Trust Principal Research Fellowship (Ref: 088130/Z/09/Z). GP was funded by the European Union’s Horizon 2020 Framework Programme for Research and Innovation under the Specific Grant Agreement No. 945539 (Human Brain Project SGA3) and the European Research Council under the Grant Agreement No. 820213 (ThinkAhead). The GEFORCE Titan GPU cards used for this research were donated by the NVIDIA Corporation. The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

References

[1] G. Pezzulo, F. Donnarumma, D. Maisto, I. Stoianov, Planning at decision time and in the background during spatial navigation, Curr. Opin. Behav. Sci. 29 (2019) 69–76. https://doi.org/10.1016/j.cobeha.2019.04.009.
[2] H. Geffner, Model-free, Model-based, and General Intelligence, ArXiv Prepr.
ArXiv180602308. (2018).
[3] N.D. Daw, P. Dayan, The algorithmic anatomy of model-based evaluation, Philos. Trans. R. Soc. B Biol. Sci. 369 (2014) 20130478.
[4] S.J. Russell, P. Norvig, Artificial intelligence: a modern approach (International Edition), (2002). http://www.citeulike.org/group/1104/article/115157 (accessed September 13, 2017).
[5] R.S. Sutton, A.G. Barto, Reinforcement Learning: An Introduction, MIT Press, Cambridge MA, 1998.
[6] H. Geffner, Computational models of planning, Wiley Interdiscip. Rev. Cogn. Sci. 4 (2013) 341–356.
[7] K.J. Friston, The free-energy principle: a unified brain theory?, Nat Rev Neurosci. 11 (2010) 127–138. https://doi.org/10.1038/nrn2787.
[8] K. Friston, F. Rigoli, D. Ognibene, C. Mathys, T. Fitzgerald, G. Pezzulo, Active inference and epistemic value, Cogn. Neurosci. 6 (2015) 187–214. https://doi.org/10.1080/17588928.2015.1020053.
[9] K. Friston, T. FitzGerald, F. Rigoli, P. Schwartenbeck, G. Pezzulo, Active Inference: A Process Theory, Neural Comput. 29 (2017) 1–49. https://doi.org/10.1162/NECO_a_00912.
[10] K. Friston, L. Da Costa, D. Hafner, C. Hesp, T. Parr, Sophisticated Inference, ArXiv200604120 Cs Q-Bio. (2020). http://arxiv.org/abs/2006.04120 (accessed January 30, 2021).
[11] E. Todorov, General duality between optimal control and estimation, in: 2008 47th IEEE Conf. Decis. Control, IEEE, 2008: pp. 4286–4292.
[12] S. Levine, Reinforcement Learning and Control as Probabilistic Inference: Tutorial and Review, ArXiv180500909 Cs Stat. (2018). http://arxiv.org/abs/1805.00909 (accessed July 18, 2020).
[13] K. Rawlik, M. Toussaint, S. Vijayakumar, On stochastic optimal control and reinforcement learning by approximate inference, in: Twenty-Third Int. Jt. Conf. Artif. Intell., 2013.
[14] H. Attias, Planning by Probabilistic Inference, in: Proc. Ninth Int. Workshop Artif. Intell. Stat., 2003.
[15] M. Botvinich, M. Toussaint, Planning as inference, Trends Cogn. Sci. 16 (2012) 485–488. https://doi.org/10.1016/j.tics.2012.08.006.
[16] H.J. Kappen, V. Gómez, M. Opper, Optimal control as a graphical model inference problem, Mach. Learn. 87 (2012) 159–182.
[17] G. Pezzulo, F. Rigoli, K. Friston, Hierarchical Active Inference: A Theory of Motivated Control, Trends Cogn. Sci. 22 (2018) 294–306. https://doi.org/10.1016/j.tics.2018.01.009.
[18] K.J. Friston, P. Schwartenbeck, T. FitzGerald, M. Moutoussis, T. Behrens, R.J. Dolan, The anatomy of choice: dopamine and decision-making, Philos. Trans. R. Soc. Lond. B Biol. Sci. 369 (2014) 20130481. https://doi.org/10.1098/rstb.2013.0481.
[19] K.J. Friston, T. Parr, B. de Vries, The graphical brain: Belief propagation and active inference, Netw. Neurosci. 1 (2017) 381–414. https://doi.org/10.1162/NETN_a_00018.
[20] K.J. Friston, R. Rosch, T. Parr, C. Price, H. Bowman, Deep temporal models and active inference, Neurosci. Biobehav. Rev. 77 (2017) 388–402. https://doi.org/10.1016/j.neubiorev.2017.04.009.
[21] K. Ueltzhöffer, Deep Active Inference, ArXiv170902341 Q-Bio. (2017). http://arxiv.org/abs/1709.02341 (accessed September 8, 2017).
[22] B. Millidge, Deep Active Inference as Variational Policy Gradients, ArXiv190703876 Cs. (2019). http://arxiv.org/abs/1907.03876 (accessed February 10, 2020).
[23] A. Clark, Surfing Uncertainty: Prediction, Action, and the Embodied Mind, Oxford University Press, Incorporated, 2015.
[24] S. Gelly, D. Silver, Monte-Carlo tree search and rapid action value estimation in computer Go, Artif. Intell. 175 (2011) 1856–1875. https://doi.org/10.1016/j.artint.2011.03.007.
[25] L. Kocsis, C. Szepesvári, Bandit Based Monte-Carlo Planning, in: J. Fürnkranz, T. Scheffer, M. Spiliopoulou (Eds.), Mach. Learn. ECML 2006, Springer, Berlin, Heidelberg, 2006: pp. 282–
Observability.pdf.

[46] E. Galceran, A.G. Cunningham, R.M. Eustice, E. Olson, Multipolicy decision-making for autonomous driving via changepoint-based behavior prediction: Theory and experiment, Auton. Robots. 41 (2017) 1367–1382. https://doi.org/10.1007/s10514-017-9619-z.

[47] W. Schwarting, J. Alonso-Mora, D. Rus, Planning and Decision-Making for Autonomous Vehicles, Annu. Rev. Control Robot. Auton. Syst. 1 (2018) 187–210. https://doi.org/10.1146/annurev-control-060117-105157.

[48] C. Badue, R. Guidolini, R.V. Carneiro, P. Azevedo, V.B. Cardoso, A. Forechi, L. Jesus, R. Berriel, T. Paixão, F. Mutz, L. Veronese, T. Oliveira-Santos, A.F. De Souza, Self-Driving Cars: A Survey, ArXiv190104407 Cs. (2019). http://arxiv.org/abs/1901.04407 (accessed July 25, 2020).

[49] W.R. Ashby, Design for a brain, Wiley, Oxford, England, 1952.

[50] W.T. Powers, Behavior: The Control of Perception, Aldine, Hawthorne, NY, 1973.

[51] N. Wiener, Cybernetics: or Control and Communication in the Animal and the Machine, The MIT Press, Cambridge, MA, 1948.

[52] G. Pezzulo, F. Rigoli, K.J. Friston, Active Inference, homeostatic regulation and adaptive behavioural control, Prog. Neurobiol. 136 (2015) 17–35.

[53] A.K. Seth, The cybernetic Bayesian brain: from interoceptive inference to sensorimotor contingencies, MIND Proj. Eds T Metzinger J Windt. (2014).

[54] F. Donnarumma, D. Maisto, G. Pezzulo, Problem Solving as Probabilistic Inference with Subgoaling: Explaining Human Successes and Pitfalls in the Tower of Hanoi, PLOS Comput. Biol. 12 (2016) e1004864. https://doi.org/10.1371/journal.pcbi.1004864.

[55] G. Pezzulo, F. Donnarumma, P. Iodice, D. Maisto, I. Stoianov, Model-Based Approaches to Active Perception and Control, Entropy. 19 (2017) 266. https://doi.org/10.3390/e19060266.

[56] G. Pezzulo, F. Rigoli, F. Chersi, The Mixed Instrumental Controller: using Value of Information to combine habitual choice and mental simulation, Front. Cogn. 4 (2013) 92. https://doi.org/10.3389/fpsyg.2013.00092.

[57] K. Friston, S. Samothrakis, R. Montague, Active inference and agency: optimal control without cost functions, Biol. Cybern. 106 (2012) 523–541.

[58] K.J. Friston, T. FitzGerald, F. Rigoli, P. Schwartenbeck, J. O’Doherty, G. Pezzulo, Active inference and learning, Neurosci. Biobehav. Rev. 68 (2016) 862–879. https://doi.org/10.1016/j.neubiorev.2016.06.022.

[59] H. Attias, Planning as Probabilistic Inference, in: Proc. Ninth Int. Workshop Artif. Intell. Stat. AISTATS 2003, Society for Artificial Intelligence and Statistics, Key West, Florida, USA, 2003.

[60] M.J. Beal, Variational algorithms for approximate Bayesian inference, University of London, 2003. http://www.cse.buffalo.edu/faculty/mbeal/papers/beal03.pdf (accessed August 5, 2015).

[61] K. Friston, P. Schwartenbeck, T. FitzGerald, M. Moutoussis, T. Behrens, R.J. Dolan, The anatomy of choice: active inference and agency, Front. Hum. Neurosci. 7 (2013) 598. https://doi.org/10.3389/fnhum.2013.00598.

[62] D. Maisto, K. Friston, G. Pezzulo, Caching mechanisms for habit formation in Active Inference, Neurocomputing. 359 (2019) 298–314. https://doi.org/10.1016/j.neucom.2019.05.083.

[63] T. Parr, K.J. Friston, Uncertainty, epistemics and active inference, J. R. Soc. Interface. 14 (2017) 20170376. https://doi.org/10.1098/rsif.2017.0376.

[64] P. Schwartenbeck, J. Passecker, T.U. Hauser, T.H. FitzGerald, M. Kronbichler, K.J. Friston, Computational mechanisms of curiosity and goal-directed exploration, ELife. 8 (2019) e41703. https://doi.org/10.7554/eLife.41703.

[65] K.J. Friston, T. FitzGerald, F. Rigoli, P. Schwartenbeck, G. Pezzulo, Active Inference: A Process Theory, Neural Comput. (2016) 1–49. https://doi.org/10.1162/NECO_a_00912.

[66] P. Schwartenbeck, T.H. FitzGerald, C. Mathys, R. Dolan, K. Friston, The dopaminergic midbrain encodes the expected certainty about desired outcomes, Cereb. Cortex. (2014) bhu159.
[67] A. Tschantz, M. Baltieri, A.K. Seth, C.L. Buckley, Scaling active inference, ArXiv191110601 Cs Eess Math Stat. (2019). http://arxiv.org/abs/1911.10601 (accessed December 4, 2019).

[68] B. Abramson, Expected-outcome: a general model of static evaluation, IEEE Trans. Pattern Anal. Mach. Intell. 12 (1990) 182–193. https://doi.org/10.1109/34.44404.

[69] L. Kocsis, C. Szepesvari, J. Willemsen, Improved Monte-Carlo Search, University of Tartu, Tartu, Estonia, 2006.

[70] T. Parr, K.J. Friston, The Computational Anatomy of Visual Neglect, Cereb. Cortex. 28 (2018) 777–790. https://doi.org/10.1093/cercor/bhx316.

[71] V. Kuleshov, D. Precup, Algorithms for multi-armed bandit problems, ArXiv14026028 Cs. (2014). http://arxiv.org/abs/1402.6028 (accessed February 11, 2020).

[72] Y.W. Teh, M.I. Jordan, M.J. Beal, D.M. Blei, Hierarchical dirichlet processes, J. Am. Stat. Assoc. 101 (2006) 1566–1581.

[73] A. Somani, N. Ye, D. Hsu, W.S. Lee, DESPOT: Online POMDP Planning with Regularization, in: C.J.C. Burges, L. Bottou, M. Welling, Z. Ghahramani, K.Q. Weinberger (Eds.), Adv. Neural Inf. Process. Syst. 26, Curran Associates, Inc., 2013: pp. 1772–1780.

[74] Z. Sunberg, M. Kochenderfer, Online algorithms for POMDPs with continuous state, action, and observation spaces, in: Twenty-Eighth Int. Conf. Autom. Plan. Sched., 2018.

[75] D.S. Nau, An investigation of the causes of pathology in games, Artif. Intell. 19 (1982) 257–278. https://doi.org/10.1016/0004-3702(82)90002-9.

[76] R. Ramanujan, A. Sabharwal, B. Selman, On adversarial search spaces and sampling-based planning, in: Proc 20th Int Conf Autom Plan Sched, Toronto, ON, Canada, 2010: pp. 242–245.

[77] J. Steven, G. Konidaris, B. Rosman, An analysis of monte carlo tree search, in: Thirty-First AAAI Conf. Artif. Intell., 2017.

[78] J. Pazis, R. Parr, Non-parametric approximate linear programming for MDPs, in: Proc. Twenty-Fifth AAAI Conf. Artif. Intell., AAAI Press, San Francisco, California, 2011: pp. 459–464.

[79] S.C.W. Ong, S.W. Png, D. Hsu, W.S. Lee, POMDPs for Robotic Tasks with Mixed Observability, in: Robot. Sci. Syst., 2009.

[80] A.R. Cassandra, L.P. Kaelbling, M.L. Littman, Acting optimally in partially observable stochastic domains, in: Aaaai, 1994: pp. 1023–1028.

[81] D.J. Foster, M.A. Wilson, Hippocampal theta sequences, Hippcampus. 17 (2007) 1093–1099. https://doi.org/10.1002/hipo.20345.

[82] G. Pezzulo, M.A.A. van der Meer, C.S. Lansink, C.M.A. Pennartz, Internally generated sequences in learning and executing goal-directed behavior, Trends Cogn. Sci. 18 (2014) 647–657. https://doi.org/10.1016/j.tics.2014.06.011.

[83] G. Pezzulo, C. Kemere, M. van der Meer, Internally generated hippocampal sequences as a vantage point to probe future-oriented cognition, Ann. N. Y. Acad. Sci. 1396 (2017) 144–165.

[84] B.E. Pfeiffer, D.J. Foster, Hippocampal place-cell sequences depict future paths to remembered goals, Nature. 497 (2013) 74–79. https://doi.org/10.1038/nature12112.

[85] A.D. Redish, Vicarious trial and error, Nat. Rev. Neurosci. 17 (2016) 147–159. https://doi.org/10.1038/nrn.2015.30.

[86] K.J. Miller, M.M. Botvinick, C.D. Brody, Dorsal hippocampus contributes to model-based planning, Nat. Neurosci. 20 (2017) 1269–1276. https://doi.org/10.1038/nn.4613.

[87] H. Mushiake, N. Saito, K. Sakamoto, Y. Itoyama, J. Tanji, Activity in the lateral prefrontal cortex reflects multiple steps of future events in action plans., Neuron. 50 (2006) 631–641. https://doi.org/10.1016/j.neuron.2006.03.045.

[88] N. Saito, H. Mushiake, K. Sakamoto, Y. Itoyama, J. Tanji, Representation of immediate and final behavioral goals in the monkey prefrontal cortex during an instructed delay period., Cereb Cortex. 15 (2005) 1535–1546. https://doi.org/10.1093/cercor/bhi032.
[89] Z. Kurth-Nelson, M. Economides, R.J. Dolan, P. Dayan, Fast Sequences of Non-spatial State Representations in Humans, Neuron. 91 (2016) 194–204. https://doi.org/10.1016/j.neuron.2016.05.028.

[90] N.W. Schuck, Y. Niv, Sequential replay of nonspatial task states in the human hippocampus, Science. 364 (2019). https://doi.org/10.1126/science.aaw5181.

[91] D.L.K. Yamins, J.J. DiCarlo, Using goal-driven deep learning models to understand sensory cortex, Nat. Neurosci. 19 (2016) 356–365. https://doi.org/10.1038/nn.4244.

[92] D. Sussillo, M.M. Churchland, M.T. Kaufman, K.V. Shenoy, A neural network that finds a naturalistic solution for the production of muscle activity, Nat. Neurosci. 18 (2015) 1025–1033. https://doi.org/10.1038/nn.4042.

[93] K.R. Storrs, N. Kriegeskorte, Deep learning for cognitive neuroscience, ArXiv Prepr. ArXiv190301458. (2019).

[94] D. George, W. Lehrach, K. Kansky, M. Lázaro-Gredilla, C. Laan, B. Marthi, X. Lou, Z. Meng, Y. Liu, H. Wang, A. Lavin, D.S. Phoenix, A generative vision model that trains with high data efficiency and breaks text-based CAPTCHAs, Science. 358 (2017). https://doi.org/10.1126/science.aag2612.

[95] M. Botvinick, J.X. Wang, W. Dabney, K.J. Miller, Z. Kurth-Nelson, Deep Reinforcement Learning and Its Neuroscientific Implications, Neuron. 107 (2020) 603–616. https://doi.org/10.1016/j.neuron.2020.06.014.

[96] Q.J.M. Huys, N. Lally, P. Faulkner, N. Eshel, E. Seifritz, S.J. Gershman, P. Dayan, J.P. Roiser, Interplay of approximate planning strategies, Proc. Natl. Acad. Sci. U. S. A. 112 (2015) 3098–3103. https://doi.org/10.1073/pnas.1412431112.

[97] M. Keramati, P. Smittenaar, R.J. Dolan, P. Dayan, Adaptive integration of habits into depth-limited planning defines a habitual-goal–directed spectrum, Proc. Natl. Acad. Sci. U. S. A. 113 (2016) 12868–12873. https://doi.org/10.1073/pnas.1609094113.

[98] P.A. Ortega, D.A. Braun, Thermodynamics as a theory of decision-making with information-processing costs, Proc. R. Soc. Math. Phys. Eng. Sci. 469 (2013). https://doi.org/10.1098/rspa.2012.0683.

[99] S.C.W. Ong, Shao Wei Png, D. Hsu, Wee Sun Lee, Planning under Uncertainty for Robotic Tasks with Mixed Observability, Int. J. Robot. Res. 29 (2010) 1053–1068. https://doi.org/10.1177/0278364910369861.