Supplementary Materials

Environment generation

All the parameters used for generating environments are described in Table 1. The environment-generating algorithm has three steps:

1. **Maze generation**: A maze is generated with a given number n_s of states and branching rate, where both n_s and the branching rate are free parameters (Table 1). The branching rate determines the number of path intersections in the environment. The algorithm for generating the maze is defined in Algorithm 1.

2. **Room integration**: A specific number n_{rooms} of states in the maze are randomly sampled to transform into rooms with a pre-specified size. Both n_{rooms} and the room size are free parameters (Table 1). A room is a set of states organized in a square grid. Each state within a room has four actions to navigate up, down, left, or right whenever these actions are available (when the state is not at the room border). The neighbors of the state that is transformed into a room are connected to the middle of the room borders (a maximum of four neighbors, one for each side of the square room).

3. **Room-property assignment**: Each room is randomly assigned to be either sink, source, stochastic, or neutral, where the frequency of each room property is specified by free parameters (Table 1). For each sink room, we iteratively sample a state u in the room and a state v outside the room, uniformly at random, and connect v to u. We repeat this procedure until the desired number of edges (a free parameter; Table 1) has been added. For each source room, we do the same process but with an inverted direction of connections – with the desired number of edges being a free parameter (Table 1). For each stochastic room, we change the transition dynamics as follows: When an agent selects an action a from a state s within the room, there is a fixed probability (free parameter; Table 1) that the action will result in the agent moving to a random neighbor of s instead of the intended destination of a. Neutral rooms do not receive any modification.

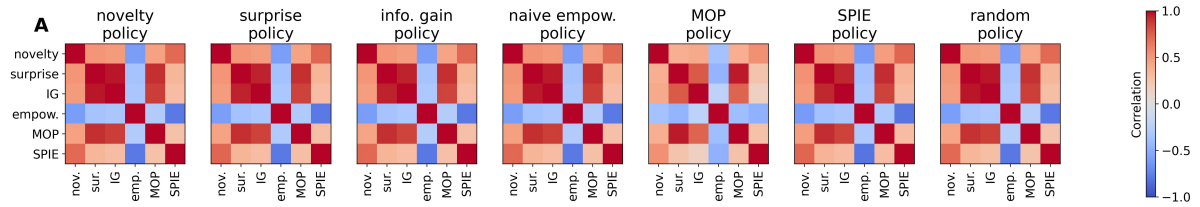
Algorithm 1 Algorithm to generate the initial maze

Require: $n > 0$, branch_rate $\in [0, 1]$
  $Q \leftarrow$ empty queue
  ENQUEUE$(Q, 1)$ ▷ Append 1 to the queue
  next_state $\leftarrow 2$
  while next_state $\leq n$ **do**
    cur_state $\leftarrow$ DEQUEUE(Q) ▷ Pop the first element of the queue
    CONNECT(cur_state, next_state)
    CONNECT(next_state, cur_state)
    rand $\in [0, 1]$ uniformly at random
    if rand $<$ branch_rate **and** $n_{\text{neighbors}}$(cur_state) < 4 **then**
      ENQUEUE$(Q$, cur_state) ▷ The current state is put back in the queue if it does not already have 4 neighbors
    end if
    ENQUEUE$(Q$, next_state) ▷ Append next state to the queue
    next_state += 1
  end while

Parameter	Range	Short description	Environment classes					
			Neutral	Sink	Source	Stochastic	Mixed	Trap (Fig. 4)
n_s	$[1, \infty)$	Number of states in the initial maze.	40	40	40	40	40	97
branch rate	[0,1]	Probability of creating a new intersection when adding a state.	0.2	0.2	0.2	0.2	0.2	$0 \to 1$
n_{rooms}	$[0, n_s]$	Number of rooms.	4	4	4	4	4	1
room size	$[1, \infty)$	Size of the side of rooms.	4	4	4	4	4	2
p_{sink}	[0,1]	Fraction of sink rooms.	0	0.25	0	0	0.25	1
p_{source}	$[0, 1 - p_{\text{sink}}]$	Fraction of source rooms.	0	0	0.25	0	0.25	0
$p_{\text{stochastic}}$	$[0, 1 - p_{\text{sink}} - p_{\text{source}}]$	Fraction of stochastic rooms.	0	0	0	0.25	0.25	0
$n_{\text{edges per sink}}$	$[0, \infty)$	Number of additional connection leading to each sink room.	0	50	0	0	50	$0 \to 200$
$n_{\text{edges per source}}$	$[0, \infty)$	Number of additional connection originating from each source room.	0	0	50	0	50	0
uncontrollability	[0,1]	Probability for an action taken in a stochastic room to lead to a random neighbor instead of the expected destination.	0	0	0	1	1	0

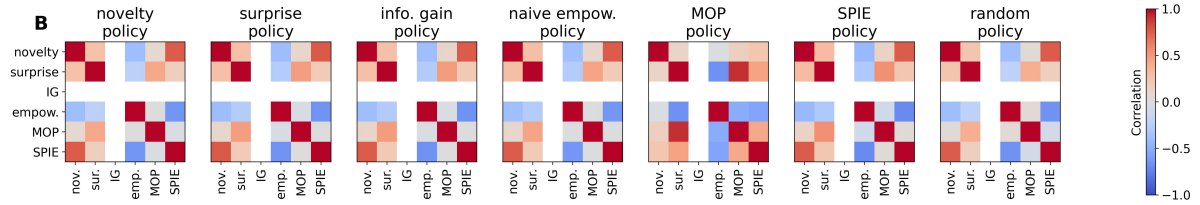
Table 1: Summary of all environment parameters used in the generation process. The right side shows the environment classes considered with the corresponding parameter values.

Robustness of results



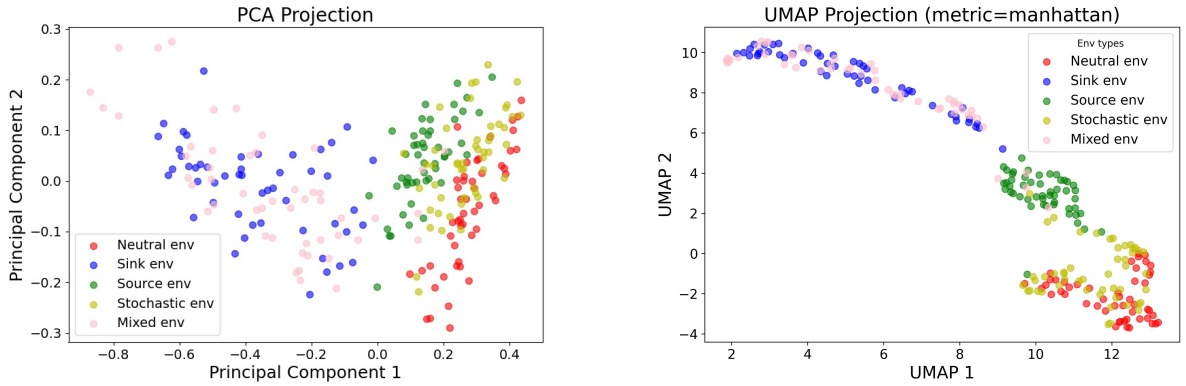
(a) Correlation between intrinsic rewards **during learning**

.



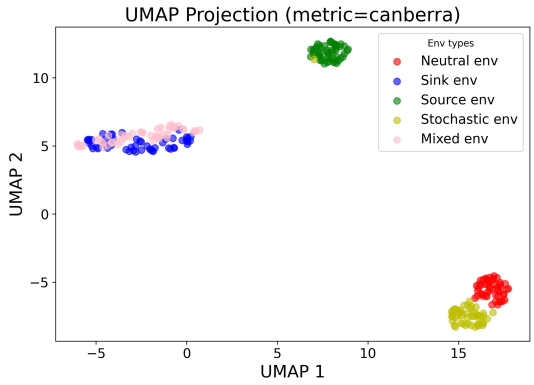
(b) Correlation between intrinsic rewards **after ideal learning**.

Figure S1: **Correlations between intrinsic rewards, under all policies.** Agents were run under the same conditions as in Fig. 7, i.e., for 10,000 steps in 50 instances of Mixed environments with four rooms (see Environment generation). Each room contained 16 states, while the corridor had 36 states. To have the same level of randomness in decision-making, the softmax inverse temperature β was fixed at $\frac{1}{\text{std}(r)}$ where $\text{std}(r)$ is the standard deviation of the intrinsic reward r computed over 10'000 steps under a random policy. **A.** The agent does not have prior knowledge of transition probabilities. **B.** The agent has full knowledge of the environment. In both cases, at each step, all intrinsic rewards are computed, but only one is used to drive the policy. To ensure an unbiased estimation of correlations, we repeat the experiment multiple times, each time using a different intrinsic reward as the policy. For each trajectory type, we compute a separate correlation matrix between all intrinsic rewards. The seven panels in the figure summarize these results.

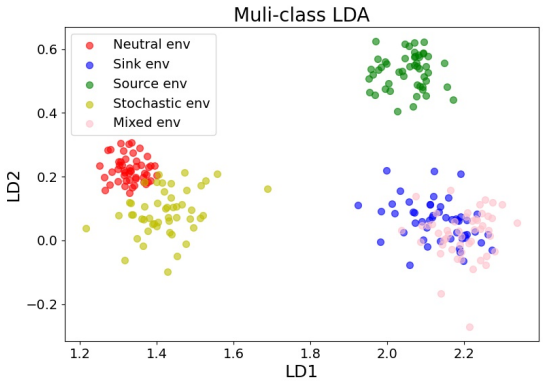


(a) PCA

(b) UMAP with Manhattan distance



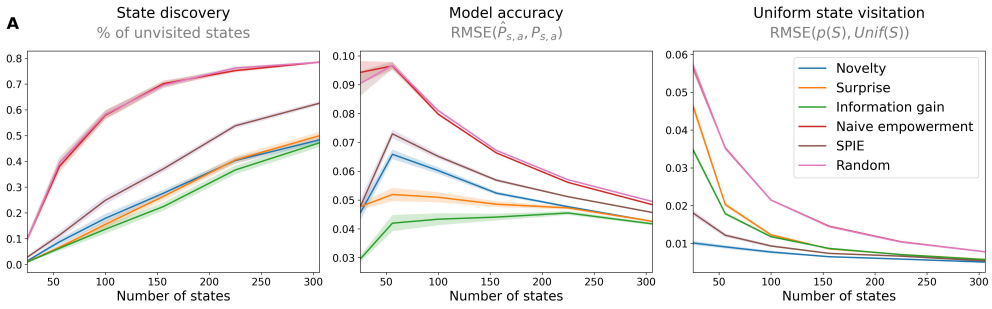
(c) UMAP with Canberra distance



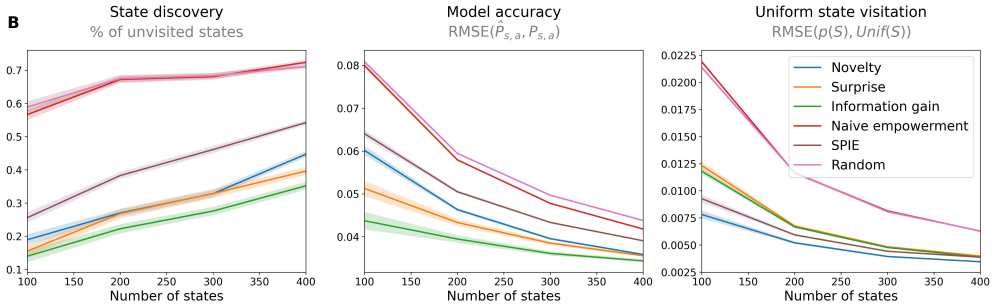
(d) Multi-class Linear Discriminant Analysis (LDA)

Figure S2: **Consistency of performance within each environment class.** The environment classes are described in Environment generation. For different sampled environments from different environment classes, we simulated intrinsically motivated agents with the six different reward signals for 2000 steps. Then, for each pair of intrinsic rewards and sampled environment, we calculated the Area Under the Curve for each of the three performance measures (same curve as in Fig. 3). As a result, for each sampled environment, we obtained a performance vector of size $6 \times 3 = 18$. Different panels show different projections of these performance vectors for different sampled environments – colored by the environment class: **A**. PCA-projection. **B**-**C**. UMAP-projection with Manhattan and Canberra distances, respectively. **D**. Multi-class LDA projection. Each dot corresponds to one sampled environment.

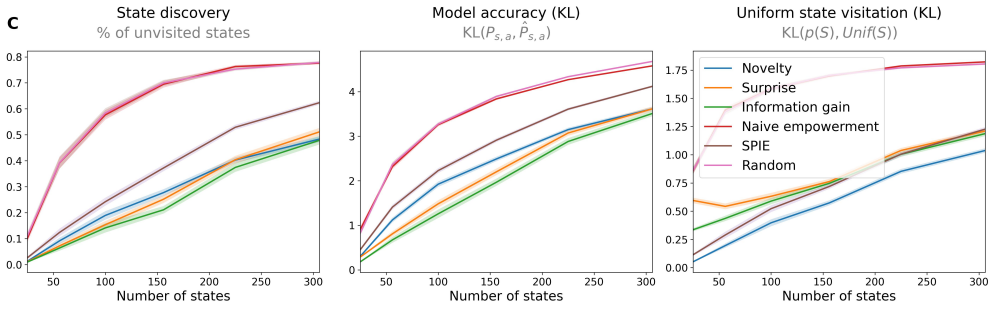
Robustness to the number of states



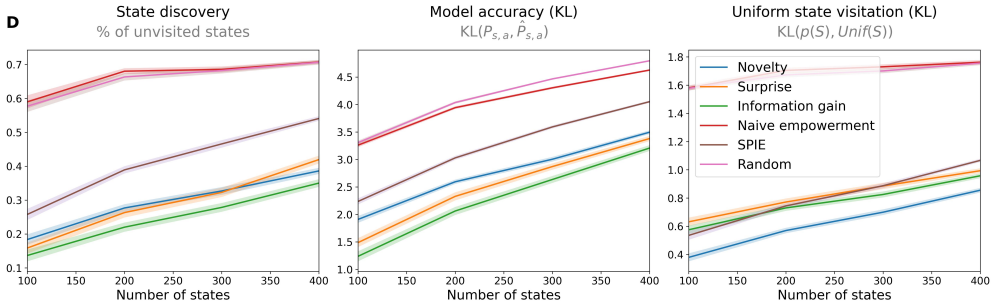
(a) 4 rooms of increasing size, RMSE for measures 2 and 3.



(b) Increasing number of 4x4 rooms, RMSE for measures 2 and 3.



(c) 4 rooms of increasing size, KL divergence for measures 2 and 3.



(d) Increasing number of 4x4 rooms, KL divergence for measures 2 and 3.

Figure S3: Caption on next page.

Figure S3: **The relative performance of intrinsic rewards is stable as the number of states increases.** Agents were tested in a modified Mixed environment class (described in Environment generation) with varying numbers of states. In all scenarios, corridor states made up 36% of the total states. In **A** and **C**, the environments contained 4 rooms, where the room size was varied between 2×2 and 7×7. In **B** and **D**, the room size was kept at 4×4, but the total number of rooms was varied between 4 and 16. We considered a uniform ratio of neutral, sink, source, and stochastic rooms. The number of additional edges per sink and source room increased proportionally with the number of states in the rooms. Each agent was simulated for 6000 steps, and performance was averaged across 50 environment instances. Intuitively, as the number of states increases, agents' performance should decrease. However, in **A** and **B**, the performance seems to improve for Measure 2 and 3. This is because we used RMSE to calculate the difference between two probability distributions with increasing support (the probability distributions are over the number of states). While RMSE is sensible and easy to interpret when the number of states is fixed, it becomes less intuitive when the number of states varies. On the other hand, in **C** and **D**, using KL divergence for measures 2 and 3 yields more intuitive results. MOP agents were excluded from this analysis due to numerical instability in our implementation of the optimization procedure described in Moreno-Bote and Ramirez-Ruiz (2023) for large state spaces, large discount factor γ and highly stochastic transitions; although mixed environments are only slightly stochastic, the agents' initial uniform estimation of transition probabilities across all states introduced significant stochasticity.

Robustness to the number of steps

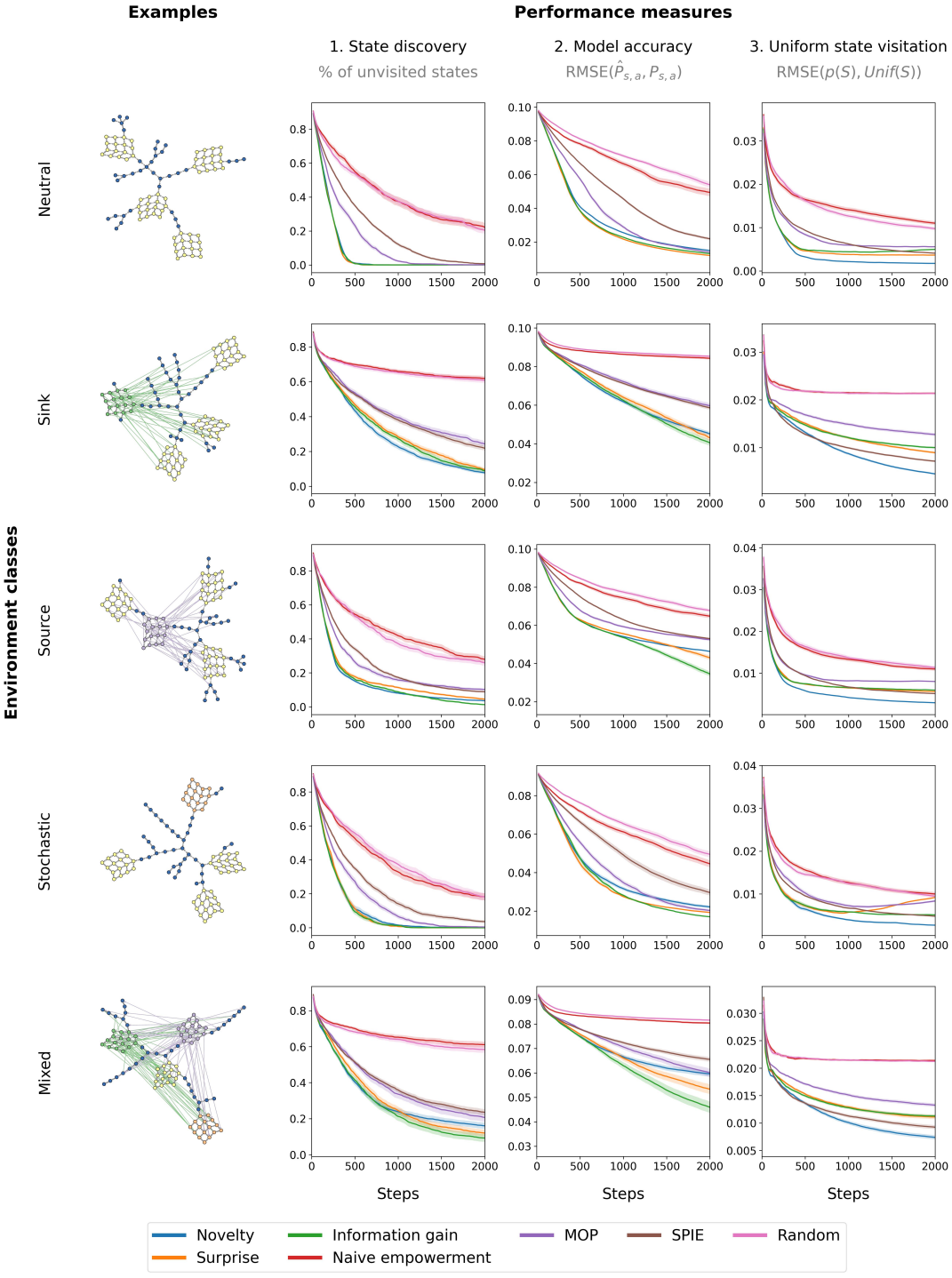


Figure S4: **Reproduction of Fig. 3 with agents running for 4000 steps instead of 2000.** The same qualitative conclusion holds. Some phenomena observed in Fig. 3 are quantitatively more pronounced, such as the advantage of seeking information gain or surprise in building accurate models in environments with sources (column 2, row 3) and the attraction of seeking surprise and MOP to the stochastic room (column 3, row 4).

Robustness to change of metrics

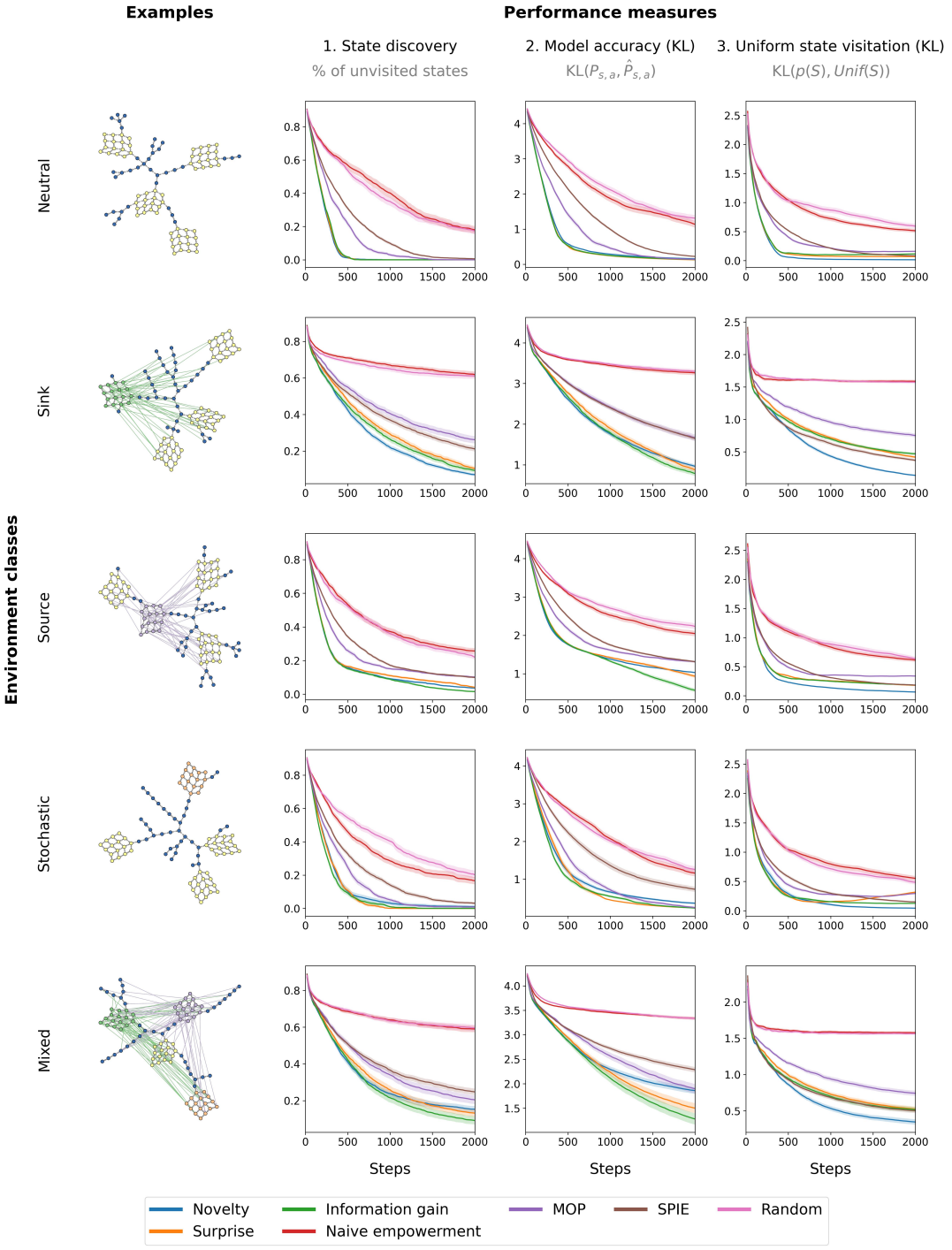


Figure S5: **Reproduction of Fig. 3 using the KL divergence instead of RMSE for Measure 2 and 3.** The agents were optimized for this version of the performance measures, and the simulations of Fig. 3 were repeated. The change of metric does not influence our conclusions.