LOOK BEFORE YOU STOP: HOW LANGUAGE MODELS RESPOND TO CHANGES IN SEARCH COSTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Recent developments in the capabilities of large language models (LLMs) have allowed for their increasingly effective deployment in situations where they are delegated to make decisions. In many of these economically relevant settings, LLMs must make dynamic choices where information acquisition is costly, which requires them to understand the value of their own future decision-making behavior. We evaluate how LLMs trade off the value of continued search against its cost across various dynamic economic contexts and cost regimes. In a sequential search game where exploration is costly, we find that meaningful sensitivity of exploration to changes in search costs, an essential feature of payoff-optimal Bayesian decision-makers, is a capability exhibited by only the most recent reasoning LLMs. We additionally allow this exploration cost to be charged in terms of token consumption rather than reward observations to more closely reflect the costs of deploying LLMs in real-world settings and measure LLMs' ability to internalize search costs more directly. We find that LLMs use this cost regime to adapt their thinking in order to observe more states, but that the extent to which they do so varies across models. We extend our game to a real-world shopping environment and find that reasoning LLMs' cost sensitivity generalizes to this setting but with less consistent results, indicating that this reasoning is key to dynamic decision-making in deployment but that models must possess broader capabilities in order to fully succeed in these settings. [1]

1 INTRODUCTION

As large language models (LLMs) have improved across a broad range of capabilities, they have increasingly been deployed in agentic settings where they are delegated to make meaningful decisions on behalf of users (Wang et al., 2024; Nguyen et al., 2025; Zhang et al., 2025). Many economically relevant tasks, such as hiring, shopping, and investing, require an agent to make repeated choices over time under incomplete information, forecasting the value of its own future behavior while updating its beliefs about the world through costly information acquisition. We refer to the ability to make value-maximizing decisions in the face of resource costs as *constrained recursive reasoning*, where the cost of search takes the form of compute, time spent, or user attention. Agents that lack this capability cannot accurately price the opportunity cost of continued search, so delegating to them requires greater human oversight even when their one-off decisions appear reasonable.

These costs come in two forms that place different demands on an agent. *Explicit costs* are features of the environment and known before the agent acts: the fee for pulling another product listing, the salary paid to keep a role open for another round of interviews, or the return a user gives up when their capital remains uninvested for an additional period. *Implicit costs* instead arise from the agent's own operation, for example, the computational cost of running inference measured in tokens and users' waiting time while the agent deliberates. Their distinction matters because an agent's actions commit one budget while its deliberation consumes another (Lin et al., 2026), and internalizing the two requires different things. Responding to an explicit cost requires reasoning about the environment, while responding to an implicit cost requires the agent to reason about itself, and to recognize that deliberating over whether to keep searching is itself a form of search that must

[1] We make our code available at https://anonymous.4open.science/r/llm-agent-search-costs-530F/

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

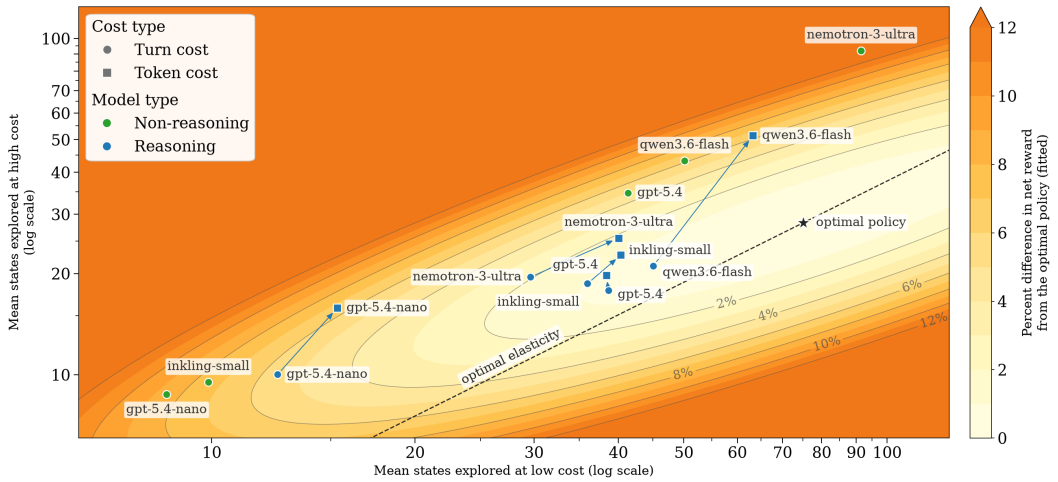


Figure 1: Mean states explored at low cost ($c = 0.15\%$, horizontal axis) and high cost ($c = 0.5\%$, vertical axis) on log scales under explicit (turn, circles) and implicit (token, squares) pricing, with the Bayes-optimal policy marked by the star. Green denotes non-reasoning models under turn cost; blue denotes reasoning models. Arrows connect each reasoning model's turn- and token-cost point. Points above the dashed line do not optimally reduce exploration as cost rises. Shading and contours show differences in net reward relative to the optimal policy as a percentage of the optimal policy's net reward, averaged across costs and fit as a quadratic function of log states explored at each cost across all runs; lighter regions indicate smaller losses.

be paid for. In particular, there is no guarantee that both costs can be learned in the same way during training.

This paper evaluates constrained recursive reasoning in LLMs by developing a benchmark based on finite-horizon sequential search games with costly exploration. In such settings, dynamically optimal behavior features a decline in both the expected payoff and the number of exploration steps as sampling becomes more expensive. We measure the sensitivity with which an agent reduces its search as these costs rise, and construct this measure as the cost elasticity of exploration (which we use as a parsimonious description of the adaptability of different LLMs to these game environments). We estimate this elasticity across a range of popular LLMs and in contexts in which you have costs that are solely based off the turns of decisions and those that reflect the actual tokens consumed in the decision-making process. We organize the main contributions of our work as follows:

- **A controlled environment with an exactly solvable optimal policy.** We design a finite-horizon sequential search game in which the agent is told the prior over the reward distribution, so the Bayes-optimal policy is computable exactly and behavior can be scored against it. We summarize each agent by a single number, cost elasticity: how much it cuts back its search when the cost of one more sample goes up so that a value of 1 means a 10% cost increase reduces exploration by 10%. Exploration is then priced two ways: explicitly, charged for every additional sample drawn, and implicitly, charged for every additional token the model generates, with the per-token price calibrated so that an agent with the same amount of reasoning as it did under explicit pricing faces exactly the same cost per sample.
- **Under explicit costs, cost sensitivity is a reasoning capability, and even the best models fall short of optimal.** With reasoning disabled, every model we test is close to cost-blind (elasticities between -0.01 and 0.30), and three show no sensitivity at all. Enabling reasoning raises elasticity for every model family, and for GPT-5.4 Nano it climbs monotonically with reasoning effort. Qwen3.6 Flash and GPT-5.4 reach elasticities of only 0.65 and 0.64 against 0.89 for the Bayes-optimal policy, even though GPT-5.4 earns net rewards within 2% of optimal at every cost we test. Near-optimal payoffs at a given cost therefore do not imply that an agent re-optimizes correctly when the cost changes.

- **Under implicit costs, reasoning models adapt their thinking in order to explore further.** We extend our game to a cost regime where agents must consider the cost of each marginal token rather than new states. When given the ability to adapt their reasoning to the importance and difficulty of the decision at hand, agents tend to think less per decision and often use the saved tokens to explore more states. The extent to which models make these adjustments varies across models, but reduced thinking and heightened exploration generally counteract one another which leaves total costs similar for the model. We find that only Nemotron 3 Ultra both increases its thinking and its exploration.

Figure 1 summarizes the first two findings. Non-reasoning models sit near the 45-degree line, barely adjusting their exploration as costs rise. Reasoning models under per-turn pricing lie much closer to the optimal policy, and the arrows show what happens when the same cost is charged per token. GPT-5.4, Inkling Small, and Nemotron 3 Ultra stay near their per-turn positions, while GPT-5.4 Nano and Qwen3.6 Flash diverge from their original behavior. Together, these results suggest that recent reasoning models can internalize explicit search costs better than non-reasoning models, but that deploying them where their own inference is the cost has more indeterminate behavior.

2 Related Work

Classical search theory. The setting in which we evaluate the cost sensitivity of exploration is a candidate-selection optimal stopping problem under incomplete information on the data-generating process (Rothschild, 1974; Weitzman, 1979; Bikhchandani & Sharma, 1996). Variants of this setting are the backbone for many seminal models of behavior in economic theory, including job-finding (McCall, 1970; Rothschild, 1974), consumer shopping (Stigler, 1961), and investment (Dixit & Pindyck, 1994). Our experiments capture the core economic tradeoffs these models describe: decision-makers must trade off the value of new information against the costs of search (or waiting, in the case of investment) based on their beliefs about the distribution of potential payoffs.

Bandits and Bayesian learning. The multi-armed bandit literature has developed several techniques for understanding the exploration-exploitation tradeoff. One example is the frequentist UCB algorithm, which attains logarithmic regret by constructing an optimistic upper bound on the empirical mean of each arm, so that under-sampled arms are scored more highly and thereby explored (Auer et al., 2002). The complementary Bayesian approaches treat the action-selection process as a function of posterior uncertainty. Such algorithms frame sequential actions as balancing exploitation against investment in information that improves later decisions, i.e., the actions are chosen not only for their payoff but also for the information that can improve future decisions (Russo et al., 2018; Ghavamzadeh et al., 2015). Our sequential search game has this structure, i.e., it is a finite-horizon sequential decision problem in which costly information acquisition improves later choices.

Sequential decision-making with LLM agents. There is a growing literature evaluating the economic rationality of LLMs. Chen et al. (2023) find that GPT's budget choices are largely consistent with utility maximization, and often more so than humans. Filippas et al. (2024) use LLMs as simulated economic agents, which they call "homo silicus", to replicate classic behavioral experiments. Extending evaluation to multi-turn interaction, Fish et al. (2025) build several benchmarks in which agents must learn an unknown environment as well as litmus tests to evaluate how agents navigate open-ended tradeoffs. Closer to our setting, Cherep et al. (2026) study sequential information acquisition with costly observations and find that LLM agents are substantially more responsive to nudges than humans. Ding et al. (2026) explicitly study sequential decision-making under uncertainty, showing that supplying inferred priors improves agents' decisions. Our setting departs from prior works in two ways: we provide the data-generating process up front, isolating in-game behavior from the model's prior, and examine agents' search behavior under an implicit per-token cost in addition to their behavior under an explicit per-sample exploration cost, allowing us to measure how agents price continued search in a state space too large to be planned over in advance.

Explicit token cost penalization during training. A related literature studies how incorporating response length into training rewards can control the computational cost of reasoning. Arora & Zanette (2025) and Luo et al. (2025) incorporate explicit incentives for shorter responses into RL-style rewards, reporting reduced token usage or inference overhead while largely preserving or improving accuracy. Aggarwal & Welleck (2025) train L1 with rewards that instead incentivize adherence to user-specified length constraints. More recent model releases also incorporate such

training incentives during reinforcement learning: Kimi k1.5 (Kimi Team, 2025) and Nemotron 3 Ultra's medium-effort reasoning mode (NVIDIA, 2026) both added length-related rewards; Thinking Machines Lab (2026) reports training Inkling to control thinking effort by jointly varying the requested effort in the system message and adjusting the per-token cost. Our study extends this literature beyond token-accuracy tradeoffs to economic decision-making, showing that token-cost training incentives are related to agents' capability to internalize the computational costs of information acquisition.

3 Task Environment

We design a finite-horizon sequential search game to test agents' ability to form beliefs and respond to exploration-exploitation tradeoffs at varying exploration costs. Agents at each stage of the game observe a reward as a real number drawn from a distribution that is unobserved but learnable over time. Following the sample draw, the agent can choose to either end the game and receive the highest reward seen or pay a cost to resample. Across different experimental configurations, we vary both the size of the resampling cost and whether it is based on the turns used or the payoff-equivalent amount of tokens consumed by the LLM. Despite the simple setting described below in Section 3.1, successful performance requires agents to forecast the value of continued search, recursively update their beliefs about the data generating process, and realize when sampling on the margin is no longer worth the cost of doing so. At the same time, the setting we implement remains tractable to calculate the optimal decision rule (Section 3.2) against which we can assess the performance of LLMs.

3.1 Sequential Search Game

Consider the following setting. Let $t = 0, \ldots, T-1$ denote the discrete game turns. At the start of each game, a latent environment parameter Θ is drawn from some distribution G_Φ. Then, candidate rewards are generated conditionally independently according to a distribution F_Θ,

$$X_t \mid \Theta \overset{i.i.d.}{\sim} F_\Theta.$$

The agent only observes X_t upon reaching turn t of the game, and knows the prior G_Φ, but not the specific value of Θ drawn.

At turn t, after observing the history $\mathcal{X}_t \equiv \{X_0, \ldots, X_t\}$, the agent chooses an action $a_t \in \{\text{stop}, \text{explore}\}$. Let

$$m_t = \max \mathcal{X}_t$$

denote the best reward observed so far. If the agent chooses $a_t = \text{stop}$, the game ends and the agent receives m_t, net of any costs already incurred. If the agent chooses $a_t = \text{explore}$, it pays cost $c > 0$ to proceed to the next period and observe X_{t+1}. At $t = T$, the game ends and the agent receives m_T. After τ periods of exploration, the agent's total payoff is

$$U_\tau(\mathcal{X}_\tau) = m_\tau - c\tau. \tag{1}$$

3.2 Bayes-optimal Algorithmic Benchmark

For a Bayesian agent, the complete game state at time t is given by the posterior belief

$$\pi_t = P(\Theta \mid \mathcal{X}_t),$$

the current best reward m_t, and the remaining number of turns. Define the value function

$$V_t(\pi_t, m_t) = \max_{a \in \{\text{stop},\text{explore}\}} Q_t^a(\pi_t, m_t), \tag{2}$$

with terminal condition $V_T(\pi_T, m_T) = m_T$ and Q_t^a the state-action values, defined as

$$Q_t^{\text{stop}}(\pi_t, m_t) = m_t$$

and

$$Q_t^{\text{explore}}(\pi_t, m_t) = -c + \mathbb{E}_{X_{t+1} \sim p(\cdot \mid \pi_t)} \left[V_{t+1} \left(\pi_{t+1}, \max\{m_t, X_{t+1}\} \right) \right],$$

where $p(\cdot \mid \pi_t)$ is the posterior predictive distribution and π_{t+1} is the posterior obtained after observing X_{t+1} and updating beliefs.

The Bayes-optimal action $a_t^*(\pi_t, m_t)$ is the action that maximizes the right-hand side of Equation 2. It can be characterized by a simple threshold in which the decision-maker stops when the exploration cost exceeds the difference between the expected continuation value and the current maximum reward. Algorithmic details of belief updating and solving the value function are in Appendix A.2 and Appendix A.3.

4 EXPERIMENTS

In this section, we describe the results from a series of sequential search experiments that vary across LLMs and the experimental configuration. Section 4.1 describes the implementation of the sequential game described in Section 3. We present the results in the case where LLMs are provided the explicit turn-based costs (Section 4.2) as well as in the case where they are told that they are penalized for token usage priced per token at the turn-based exploration cost divided by the average number of tokens the model uses per turn (Section 4.3). Section 4.4 then provides a more realistic sequential search game by adapting the WebShop environment of Yao et al. (2022).

We compare the algorithmic policy from Section 3.2 to policies formed by LLMs. The full experimental configuration is available in Appendix A.1.

4.1 LLM POLICY IMPLEMENTATION

We parameterize the setting by having rewards drawn from a normal distribution whose mean and variance $\Theta = (\mu, \sigma^2)$ are unknown to the agent. The parameter Θ is drawn from the normal-inverse-gamma conjugate prior G_Φ. In the system prompt shown to the agent at the start of the game, a new LLM instance is told that it will observe i.i.d. random variables from a normal distribution whose parameters are derived from a normal-inverse-gamma distribution from which the true parameters are given.[2] It is specifically advised that it should update its beliefs about the distribution based on the new observations. In the Appendix C.1, we also show that the main results are not primarily driven by this explicit update instruction. The instructions also clarify that it should value its actions linearly in the reward net of exploration costs.

In the user prompt shown to the agent at each successive turn, it is told the realized value for the newly observed random variable, the current turn, the number of turns remaining, and the total cost so far. It is instructed to respond only with a single word, STOP or EXPLORE, the latter of which leads to incurring the associated exploration cost. The conversation is continuous across turns so every prompt includes the full decision and observation history. Under this experimental configuration, the conversation has a context length of approximately 700 tokens on initial turns and approximately 15,000 tokens on final turns approaching T.

Our baseline experimental configuration frames the game as a theoretical decision-making problem where agents can incur a cost to sample i.i.d. random variables from an unknown distribution and must commit to one realized value. This abstract description is done to reduce misunderstandings related to the rules and allow the results to encompass a variety of economic decisions. In Appendix D.2, we use different prompt scenarios that describe economic decisions in investment, shopping, hiring, and abstract settings. All prompts are described in Appendix E.1.

4.2 SEARCH GAME RESULTS

We test our experimental configuration across a variety of models and against a turn-based cost $c = 0.05\%, 0.15\%, 0.5\%$ of the expected prior mean of the period reward, X_t. We report the mean net reward (U_τ, inclusive of exploration costs) normalized by the prior mean of 20 and average number of states explored during search for the models GPT-5.4, GPT-5.4 Nano, and Inkling Small in Figure 2. Panels (a) and (b) show the results from when reasoning is enabled for all models. The results show substantial heterogeneity across agents in both performance and behavior. Amongst the three models, GPT-5.4 (low reasoning) appears to perform the closest to the Bayes-optimal policy (Panel (a)), which is driven by the model's reduction in exploration when the sampling cost increases (Panel (b)). Inkling Small with reasoning also performs relatively close to the optimal policy and

[2]The prompt also provides the quartiles of the variance generated from the inverse-gamma distribution along with 1st- and 99th-percentile values of the initial draw to aid in its understanding of the distribution.

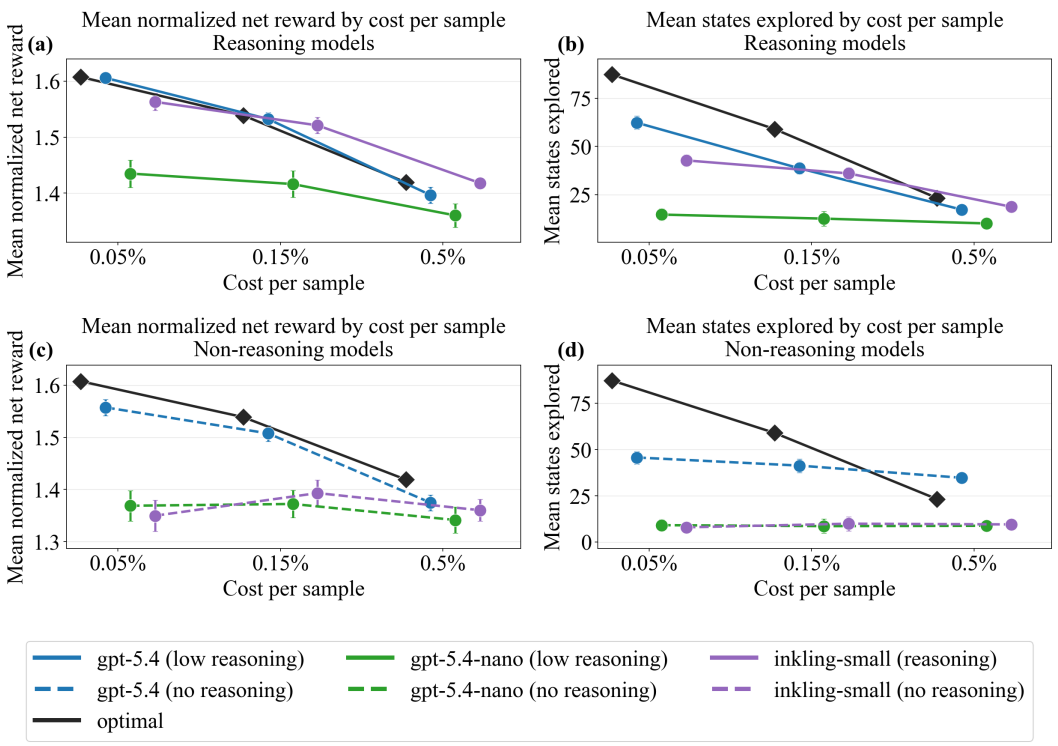


Figure 2: Mean net reward normalized by the prior mean of 20 (left column) and mean number of states explored (right column) across 200 games for up to $T = 100$ turns, shown across exploration cost $c \in \{0.05\%, 0.15\%, 0.5\%\}$. Panels (a) and (b) show the reasoning variants of GPT-5.4 and GPT-5.4 Nano (low reasoning effort) and Inkling Small; panels (c) and (d) show the same quantities for their non-reasoning variants. Solid lines indicate reasoning models; dashed lines indicate non-reasoning models. The optimal benchmark shows the performance for the Bayesian optimal policy (Section 3.2) simulated for 100,000 games. Error bars for LLM agents show 95% confidence intervals computed from the differences between the model and the optimal policy at each random seed.

exhibits a similar degree of cost sensitivity. On the other hand, GPT-5.4 Nano underperforms, with its mean state exploration virtually invariant to the costs in Panel (b). By comparison, the Bayes-optimal policy (black) responds strongly to search costs. As c rises from 0.05% to 0.5%, its mean states explored falls from 87.38 to 23.21.

In Panel (c) we run the same set of configurations for these models except we turn off reasoning. We find that prohibiting reasoning leads to clear degradation of performance of models, but to varying degrees. Both GPT-5.4 models obtain lower rewards that are around 2-3% lower than their reasoning values. Inkling Small experiences the largest decline in performance among the three model families, with mean net reward falling by approximately 8.8% when reasoning is disabled. Panel (d) shows clear evidence that the absence of reasoning makes models unresponsive to the costs of exploration. A lack of reasoning reducing adaptive behavior is unsurprising given the decision environment, which requires agents to "think ahead" in computing the future value of exploration given imperfect information on the underlying data generating process. In Appendix A.9, we further decompose changes in net reward and show that, because reasoning models explore more as the sample cost falls, their behavior across costs shifts more similarly to the optimal policy than that of their non-reasoning counterparts. Net reward and states explored normalized to the optimal policy are presented in Appendix Figure 6 and Figure 7. Full results for all models evaluated are reported in Appendix Table 2.

The differences in models' adaptability to exploration costs can be clearly captured by estimating the *cost elasticity* for each model a in experimental arm e, where an arm specifies the model's reasoning

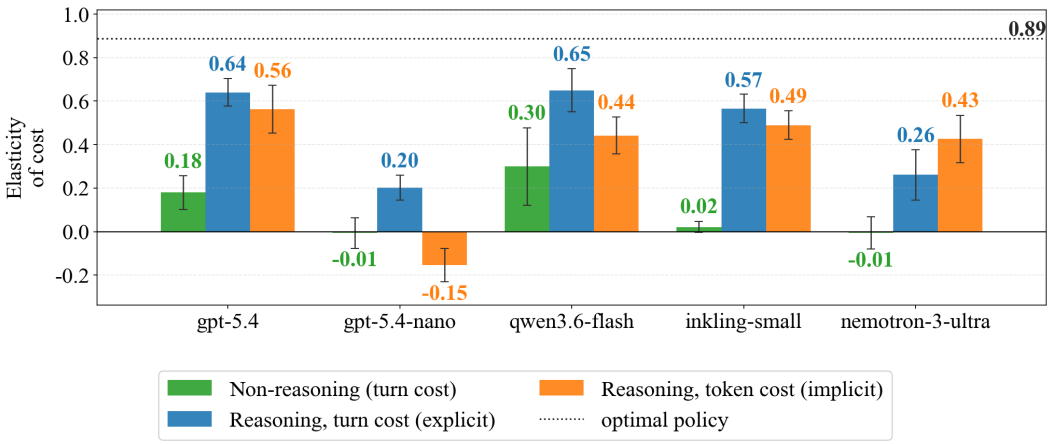


Figure 3: Sample cost elasticity of exploration for each model's non-reasoning variant under explicit (turn) pricing (green) and its reasoning variant under explicit (turn, blue) and implicit (token, orange) pricing. The elasticity is the negative slope of the log-log relationship between the number of states explored and the cost per sample, estimated from 200 runs at each of two costs, $c \in \{0.15\%, 0.5\%\}$; token-cost elasticities are taken with respect to c. All fifteen bars are estimated in one regression with agent and seed fixed effects, and error bars are 95% confidence intervals from standard errors clustered by seed, as defined in Appendix F. The dotted line shows the elasticity of the Bayesian optimal policy (Section 3.2).

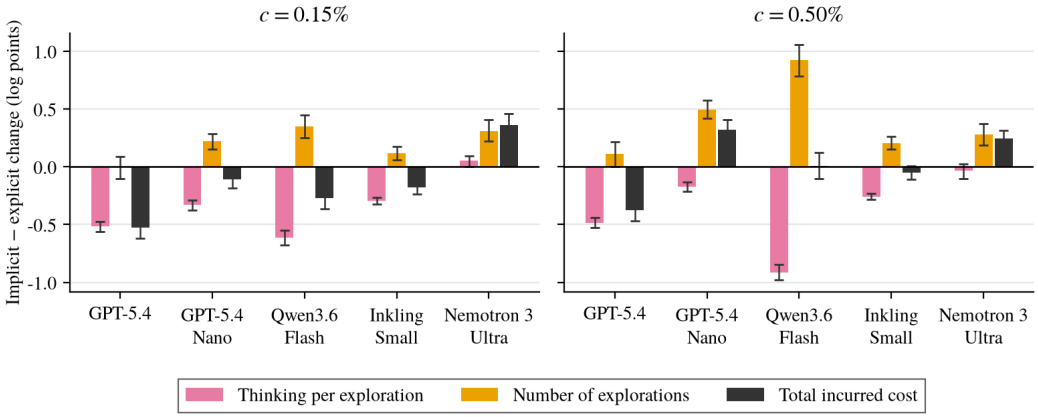


Figure 4: Decomposition of the change in total cost incurred under implicit token pricing relative to explicit turn pricing. The thinking component is the change in token cost per paid exploration, the exploration component is the change in the mean number of paid explorations, and the total is their sum, with all changes expressed in log points. A negative total cost means that the when agent is charged for per-token cost it incurs less cost than it does with per-turn cost. Error bars are 95% intervals from a paired bootstrap over the 200 matched seeds.

setting and whether exploration is priced per turn or per token. We define it as

$$\varepsilon_{a,e} = 1 - \frac{\partial\, \mathbb{E}\left[\log\left(c\, \tau_{a,e}(c)\right)\right]}{\partial \log c}, \tag{3}$$

where $\tau_{a,e}(c)$ is the number of periods of exploration τ in Equation 1 in a game under arm e at per-sample cost c, and the expectation is taken over games. When the per-turn cost is constant, we can interpret an elasticity of ε as meaning that a 1% increase in the per-sample cost reduces the number of states explored by approximately ε%. A cost-elasticity of 1 means that the agent adjusts its behavior such that it offsets changes in the direct cost of exploration.

Figure 3 presents the average elasticity within each pair of agent and experimental configuration, estimated as coefficients on linear regression with standard errors clustered by the simulation seed (Appendix F). The Bayes-optimal policy exhibits an elasticity of 0.89, which is a cost sensitivity higher than any of the LLMs tested.[3] Consistent with Figure 2, all models are less sensitive to search cost changes when reasoning is turned off (green). The nonreasoning variants of GPT-5.4 Nano, Inkling Small, and Nemotron 3 Ultra exhibit complete cost insensitivity ($\varepsilon \approx 0$). By comparison, when reasoning is turned on (blue), all models exhibit some sensitivity to turn-based exploration costs. Among the primary models we test, Qwen3.6 Flash, GPT-5.4, and Inkling Small are significantly more sensitive than the Nemotron 3 Ultra and GPT-5.4 Nano. Appendix Figure 8 further highlights the importance of reasoning in this setting by showing that, for even small LLMs, greater cost sensitivity monotonically scales with reasoning effort. Moreover, the advantage of native reasoning models is not fully explained by capabilities acquired during training independently of reasoning itself. In Appendix D.1, we show that prompting non-reasoning models to produce a freeform scratchpad also increases their performance and cost elasticity of exploration.

Taking Figure 2 and Figure 3 together, although reasoning models like GPT-5.4 may achieve net rewards close to the Bayes-optimal policy at the tested cost levels, their exploration elasticities remain lower. Thus, near-optimal realized payoffs alone do not necessarily imply that agents adjust their search behavior optimally when exploration costs change. Appendix C provides a set of additional tests of agents' decision-making behavior, finding that reasoning models make decisions more consistent with the Bayes-optimal policy.

4.3 Behavior under Token Cost Pricing

We extend the previous experiments to examine how agent internalizes their own token usage when considering exploration. To do this, we modify our experiments by setting the cost of exploration in terms of tokens rather than reward observations. To allow for comparability with LLM behavior under explicit turn-based costs, we calibrate the cost per token reported to the agent by dividing the original cost of exploration by the mean number of tokens spent per decision by the agent at that cost in the reasoning configuration of the experiments in Section 4.2. Appendix A.5 details more precisely how we calibrate the per-token costs. For agent a facing per-turn cost c, the price per token is $p(c, r^E_{a,c}) = c/r^E_{a,c}$, where $r^E_{a,c}$ is the mean number of tokens agent a generates per decision at cost c under explicit, turn-based costs.

Relative to the previous set of experiments, agents can reduce the amount of reasoning that they perform per turn to incur lower costs, in addition to reducing the number of turns. As a result, we can decompose the total cost term in Equation 1 into its explicit and implicit forms, $C^E = c\,\tau^E$ and $C^I = p\,r^I\,\tau^I$, where τ^E and τ^I are the number of states explored and r^E and r^I the average numbers of tokens per turn under explicit and implicit costs. Writing each elasticity as in Equation 3, $\varepsilon^k_x = -d\log x^k / d\log c$ for $x \in \{\tau, r\}$ and $k \in \{E, I\}$, the cost elasticity of the implicit cost, net of the one-for-one effect of the price, $\varepsilon^I_C = 1 - d\log C^I/d\log c$, decomposes as

$$\varepsilon^I_C = \left(\varepsilon^I_r - \varepsilon^E_r\right) + \varepsilon^I_\tau. \tag{4}$$

The estimates for the token-reported cost elasticity of exploration for models with reasoning are given in the orange bars in Figure 3. Among the primary models we tested, GPT-5.4 Nano and Qwen3.6 Flash exhibit the largest decline in sensitivity to costs, with the former flipping the sign in terms of how it responds to change in token pricing. On the other hand, GPT-5.4 and Inkling-Small experience relatively muted reductions in their cost elasticity, while Nemotron-3-Ultra appears to be more sensitive under the token cost formulation.

While the preceding results describe how behavior and payoffs respond as the sample cost changes, Figure 4 compares the level of total cost incurred under the two pricing regimes at each sample cost. We decompose this difference into the change in the token charge per exploration and the change in the number of explorations. Token pricing reduces the cost of exploration and increases exploration for most models; this causes the two cost effects to generally offset. The per-decision cost reduction is particularly important for GPT-5.4, but the effects are more balanced for GPT-5.4 Nano, Qwen3.6

[3] It is possible for decision-makers to exhibit a sensitivity that exceeds that of the optimal policy, as they may overcorrect their behavior from changes in the costs of exploration. However, we do not observe this behavior in any of the experiments we performed.

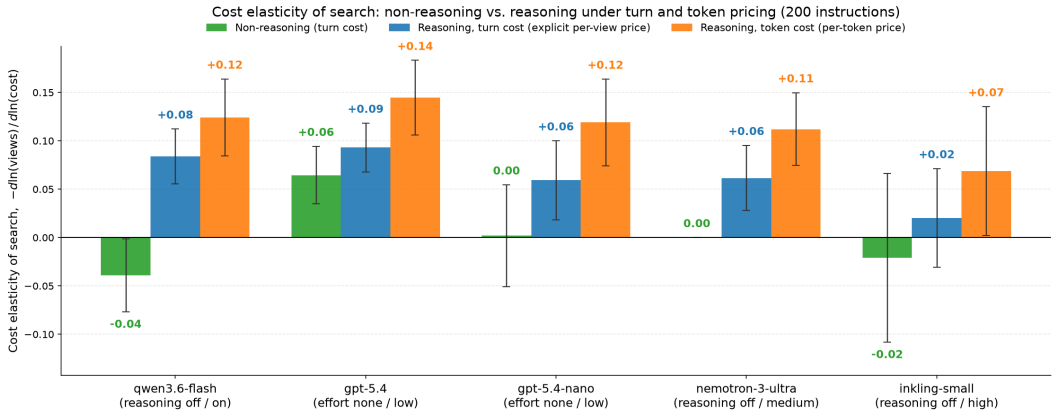


Figure 5: Sample cost elasticity of exploration in the WebShop environment for each model's non-reasoning variant under explicit (turn) pricing (green) and its reasoning variant under explicit (turn, blue) and implicit (token, orange) pricing. Elasticities are calculated in the same manner as Figure 3.

Flash, and Inkling Small. Nemotron 3 Ultra is an exception in this case: it explores more and its cost of exploration does not decrease, leading its total cost to rise with token pricing in place.

4.4 Extrapolation to Real-World Environments

To evaluate the exploration behavior of LLM agents in an economically realistic context, we adapt our primary experiments to WebShop, which is a popular environment for training agents to shop based upon product and price constraints from real user instructions (Yao et al., 2022). Agents successfully trained in WebShop also perform competently when deployed on platforms such as Amazon and eBay.

To test the cost sensitivity behavior from our environment, we modify the WebShop environment by adding a search cost for observing a new product from a list of search results and allow LLMs to decide how many products to search through before selecting one to buy. Agents in the WebShop environment are not instructed purely to minimize costs, so we expect that there will be some behavioral differences from agents in our environment. Complete implementation details and results for the WebShop environment can be found in Appendix B. Prompts are detailed in Appendix E.5.

Figure 5 shows the main results and mirrors Figure 3, which shows the cost sensitivity across the three experimental configurations. With the exception of GPT-5.4, we find high cost-insensitivity for models when reasoning is turned off. Similarly, we also find that the cost elasticities under reasoning between turn-based and calibrated token-based runs are comparable, which is consistent with the results of the experiments from Section 4.3. The relatively lower elasticities for the WebShop experiments reflect the fact that, consistent with the realism of the scenario, the agent is forming its decisions based on a larger set of considerations to optimize for than in the stylized search game.

5 Conclusion

We study how LLM agents respond to changes in cost in a variety of sequential decision-making settings under incomplete information, using a finite-horizon search game for which the Bayes-optimal policy can be computed exactly and serves as the standard of comparison. We find that the ability to achieve strong performance and adjust exploration to changing search costs emerges almost exclusively in larger and more recent reasoning models. We interpret strong performance on this task as demonstrating the model's capability to adapt to changing economic constraints and effectively manage exploration-exploitation tradeoffs. These capabilities are key to deploying LLMs as independent economic agents that can internalize user preferences and make decisions effectively. As LLMs continue to improve in this role, our evaluations can also serve as a basis for developing more sophisticated tests of constrained recursive reasoning.

AI USE STATEMENT

All research ideas presented in this work are original and human-made. LLMs were used for writing code for experiments and plot generation. LLMs were also used to generate prompts for our experiments, which were reviewed and modified in certain parts by humans. They are shown in Appendix E.

REPRODUCIBILITY STATEMENT

Our experiments access language models through inference APIs, including OpenRouter.

REFERENCES

Pranjal Aggarwal and Sean Welleck. L1: Controlling how long a reasoning model thinks with reinforcement learning. *arXiv preprint arXiv:2503.04697*, 2025.

Daman Arora and Andrea Zanette. Training language models to reason efficiently. *arXiv preprint arXiv:2502.04463*, 2025.

Peter Auer, Nicolo Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine learning*, 47(2):235–256, 2002.

Sushil Bikhchandani and Sunil Sharma. Optimal search with learning. *Journal of Economic Dynamics and Control*, 20(1-3):333–359, 1996.

Yiting Chen, Tracy Xiao Liu, You Shan, and Songfa Zhong. The emergence of economic rationality of gpt. *Proceedings of the National Academy of Sciences*, 120(51):e2316205120, 2023.

Manuel Cherep, Pattie Maes, and Nikhil Singh. Ai agents are sensitive to nudges. *Proceedings of the National Academy of Sciences*, 123(25):e2537030123, 2026.

Wenxuan Ding, Nicholas Tomlin, and Greg Durrett. Calibrate-Then-Act: Cost-aware exploration in LLM agents. *arXiv preprint arXiv:2602.16699*, 2026.

Avinash K. Dixit and Robert S. Pindyck. *Investment under Uncertainty*. Princeton University Press, Princeton, NJ, 1994.

Apostolos Filippas, John J Horton, and Benjamin S Manning. Large language models as simulated economic agents: What can we learn from homo silicus? In *Proceedings of the 25th ACM Conference on Economics and Computation*, pp. 614–615, 2024.

Sara Fish, Julia Shephard, Minkai Li, Ran I Shorrer, and Yannai A Gonczarowski. Econevals: Benchmarks and litmus tests for economic decision-making by llm agents. *arXiv preprint arXiv:2503.18825*, 2025.

Mohammad Ghavamzadeh, Shie Mannor, Joelle Pineau, and Aviv Tamar. Bayesian reinforcement learning: A survey. *Foundations and Trends® in Machine Learning*, 8(5-6):359–483, 2015.

Kimi Team. Kimi k1.5: Scaling reinforcement learning with LLMs. *arXiv preprint arXiv:2501.12599*, 2025.

Yuxiang Lin, Zihan Wang, Mengyang Liu, Yuxuan Shan, Longju Bai, Junyao Zhang, Xing Jin, Boshan Chen, Jinyan Su, Xingyao Wang, Jiaxin Pei, and Manling Li. Bagen: Are llm agents budget-aware? *arXiv preprint arXiv:2606.00198*, 2026.

Haotian Luo, Li Shen, Haiying He, Yibo Wang, Shiwei Liu, Wei Li, Naiqiang Tan, Xiaochun Cao, and Dacheng Tao. O1-Pruner: Length-harmonizing fine-tuning for O1-like reasoning pruning. *arXiv preprint arXiv:2501.12570*, 2025.

John Joseph McCall. Economics of information and job search. *The Quarterly Journal of Economics*, 84(1):113–126, 1970.

Dang Nguyen, Jian Chen, Yu Wang, Gang Wu, Namyong Park, Zhengmian Hu, Hanjia Lyu, Junda Wu, Ryan Aponte, Yu Xia, et al. Gui agents: A survey. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 22522–22538, 2025.

NVIDIA. Nemotron 3 Ultra: Open, efficient mixture-of-experts hybrid Mamba-Transformer model for agentic reasoning, 2026.

Michael Rothschild. Searching for the lowest price when the distribution of prices is unknown. *Journal of Political Economy*, 82(4):689–711, 1974.

Daniel J. Russo, Benjamin Van Roy, Abbas Kazerouni, Ian Osband, and Zheng Wen. A tutorial on thompson sampling. *Foundations and Trends in Machine Learning*, 11(1):1–99, 2018.

George J Stigler. The economics of information. *Journal of Political Economy*, 69(3):213–225, 1961.

Thinking Machines Lab. Inkling: Our open-weights model. Thinking Machines Lab blog, July 2026. Published July 15, 2026.

Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, et al. A survey on large language model based autonomous agents. *Frontiers of Computer Science*, 18(6):186345, 2024.

Martin L Weitzman. Optimal search for the best alternative. *Econometrica: Journal of the Econometric Society*, pp. 641–654, 1979.

Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. Webshop: Towards scalable real-world web interaction with grounded language agents. In *Advances in Neural Information Processing Systems*, volume 35, pp. 20744–20757, 2022.

Chaoyun Zhang, Shilin He, Jiaxu Qian, Bowen Li, Liqun Li, Si Qin, Yu Kang, Minghua Ma, Guyue Liu, Qingwei Lin, et al. Large language model-brained gui agents: A survey. *Transactions on Machine Learning Research*, 2025.

APPENDIX ROADMAP

We organize the appendix as follows:

- **A Abstract Sequential Search Game.** This section shows the experimental configurations, the computation of the Bayes-optimal benchmark, and some supplementary results for the abstract sequential search game.
  - **A.1 Experimental Configuration.** The complete set of parameters for the main experiment, including the distribution parameters, game settings, language models evaluated, and the optimal policy parameters.
  - **A.2 Belief Updating.** The normal-inverse-gamma prior initialization and the posterior belief parameter update rules.
  - **A.3 Dynamic Programming for Value Function Calculation.** The dynamic programming solution for the value function in the optimal policy, including the construction of a grid of game states which we approximate the value function over and the use of this grid at decision time for the optimal policy.
  - **A.4 Proof of Value Function Translation Equivariance.** A proof that the value function is translation equivariant, a fact which is used in the dynamic programming solution.
  - **A.5 Token Cost Calibration.** Details of the calibration of the cost per token.
  - **A.6 Plots Normalized to the Optimal Policy.** Plots from the primary experiment where results are shown as a fraction of the Bayes-optimal benchmark tested over the same seeds.
  - **A.7 Mean Reward and States Explored by Model.** A complete table of the mean normalized net reward and mean states explored across costs for every model tested.
  - **A.8 Reasoning Effort Levels.** The cost elasticity of exploration and net reward of GPT-5.4 Nano across reasoning effort levels.
  - **A.9 Net Reward Decomposition.** A decomposition of agents' change in net reward when sample cost changes into the change in the best reward observed and the net cost effect.
  - **A.10 Cost Elasticity Decomposition under Token Pricing.** A decomposition of differences between turn- and token-priced cost elasticities into changes in exploration behavior and tokens used per turn.
  - **A.11 Gross Reward Elasticity.** Gross reward elasticities with respect to search costs and a decomposition of differences between pricing regimes into an exploration channel and a residual reflecting stopping choices.
  - **A.12 Semi-Elasticities of Gross Reward, Search Cost and Net Reward.** Semi-elasticities showing how changes in gross reward and search expenditure combine to determine the response of net reward under turn and token pricing.
  - **A.13 Dollar Cost of Experiments across Models.** The average output tokens and reasoning cost per simulation for each reasoning model under explicit turn cost and implicit token cost.
- **B WebShop Experiments.** The experiment configuration and full results for the WebShop experiment.
  - **B.1 Experimental Configuration.** The configuration and setup of the WebShop environment.
  - **B.2 Full Results.** The mean number of products viewed for every model and pricing arm across search-cost levels in the WebShop experiment.
  - **B.3 Cost Elasticity Decomposition under Token Pricing.** Applying the token-pricing decomposition to WebShop, distinguishing how the number of products reviewed and tokens used per product drive total search-cost elasticity.
- **C Testing for Understanding.** Additional experiments that probe how agents choose whether to explore or stop and evaluate these decisions against the Bayes-optimal policy.
  - **C.1 Update Priming Prompt Ablation.** An ablation experiment removing the system prompt's direct instruction to update beliefs, comparing exploration, net reward, and tokens per response with and without it.

  - **C.2 Tests for Environmental Understanding.** A test of agents' understanding of the task environment by asking them to perform normal-inverse-gamma posterior belief updates after observations of varying likelihood.
  - **C.3 Continuation Value Investigation.** An experiment testing whether agents value multi-step continuation or just immediate rewards by comparing their behavior against a myopic policy and the true optimal policy.
  - **C.4 Decision-Level Value Analysis.** A decision-level evaluation of agents' explore-or-stop choices against the Bayes-optimal policy by quantifying value loss from suboptimal actions.
  - **C.5 Joint Test of Constrained Recursive Reasoning.** A joint test of whether agents choose the Bayes-optimal action when correct incorporation of posterior beliefs, sample cost, and continuation value are simultaneously required at the same decision.
  - **C.6 Equivalent Parameter Sensitivities to Cost.** An experiment testing whether agents respond identically to equivalent transformations of sample cost and prior variance.
- **D Extensions.** Experiments on how scratchpad reasoning and different prompt scenarios affect decision-making and sensitivity to search costs.
  - **D.1 Scratchpad Reasoning Experiments.** Additional experiments where we provide non-reasoning models the ability to reason and compare to fully reasoning and non-reasoning models.
  - **D.2 Alternative Prompt Scenarios.** An experiment rerunning the main configuration with Qwen3.6 Flash on three alternative prompt scenarios: shopping, investment, and hiring prompt scenarios.
- **E Prompts.** The prompts used to present the tasks and run the experiments.
  - **E.1 Prompt Scenarios.** Prompts for the abstract sequential search game and the three alternative scenarios.
  - **E.2 Scratchpad Prompts.** Prompts for the scratchpad reasoning experiments.
  - **E.3 Continuation Value Prompts.** Declining-cost and constant-cost prompts for the continuation value test.
  - **E.4 Token Cost Prompts.** Prompts for the sequential search game under implicit token cost.
  - **E.5 WebShop Prompts.** Prompts for the WebShop experiment.
  - **E.6 Environmental Understanding Prompts.** Prompts used to test agents' ability to understand the task environment.
- **F Estimation of Elasticities and Confidence Intervals.** The regression and clustered standard errors behind the cost elasticity of exploration and its confidence intervals, along with robustness check.

A Abstract Sequential Search Game

A.1 Experimental Configuration

Table 1 summarizes the configuration used in our experiments, including the prior parameters, game settings, language models evaluated, and the parameters used to compute the optimal policy.

A.2 Belief Updating

As the game progresses, we update the beliefs π about the underlying distribution of X_t using a normal-inverse-gamma prior which is initialized as

$$\pi_0 = (\mu_0, \kappa_0, \alpha_0, \beta_0) = (\mu^*, \kappa^*, \alpha^*, \beta^*) = \Phi,$$

the same parameters from which the true $\Theta = (\mu, \sigma^2)$ is drawn.

At each turn, X_{t+1} is observed and the beliefs π_t are updated to π_{t+1} as

$$\kappa_{t+1} = \kappa_t + 1, \quad \mu_{t+1} = \frac{\kappa_t \mu_t + X_{t+1}}{\kappa_{t+1}}, \quad \alpha_{t+1} = \alpha_t + \frac{1}{2}, \quad \beta_{t+1} = \beta_t + \frac{\kappa_t}{2\kappa_{t+1}}(X_{t+1} - \mu_t)^2$$

Table 1: Experiment configuration. Reasoning settings are given in parentheses. High effort for Inkling Small corresponds to a continuous effort level of 0.9 on its $[0, 0.99]$ scale.

Parameter	**Value**
Normal-inverse-gamma parameters	
μ^*	20
κ^*	1
α^*	10
β^*	250
LLM game parameters	
Number of turns T	100
Number of games	200
Scenario	Abstract
Sample costs c	{0.05%, 0.15%, 0.5%}
Large language models	
	GPT-5.4 (low)
	GPT-5.4 (no reasoning)
	GPT-5.4 mini (no reasoning)
	GPT-5.4 nano (low)
	GPT-5.4 nano (no reasoning)
	o3-mini (low)
	Qwen3.6 Flash (reasoning)
	Qwen3.6 Flash (no reasoning)
	Mistral Medium 3.1
	Inkling Small (high)
	Inkling Small (no reasoning)
	Nemotron 3 Ultra (medium)
	Nemotron 3 Ultra (no reasoning)
	DeepSeek-V4-Flash (think high)
	DeepSeek-V4-Flash (no reasoning)
	Llama-3.1-Nemotron-Ultra-253B (reasoning)
	Llama-3.1-Nemotron-Ultra-253B (no reasoning)
	Llama-3.3-Nemotron-Super-49B (no reasoning)
Optimal policy parameters	
Number of games	100,000
Grid size ($N_z \times N_\ell$)	603×243
Bound samples (N_b)	200,000
Tail probability (p_{tail})	0.001

A.3 Dynamic Programming for Value Function Calculation

We solve for the value function $V_t(\pi_t, m_t)$ with posterior belief π_t defined in Appendix A.2. We first construct a grid of possible states for each turn $t = 0, 1, \ldots, T$ and approximate the value function at each state.

Notice that κ_t and α_t are determined entirely by t and their respective initial values, κ_0 and α_0, so the value for a state (π_t, m_t) at turn t can be defined fully in terms of μ_t, β_t, m_t. Additionally, the value function is translation equivariant, proven in Appendix A.4, which allows us to reduce the dimensionality of the state space for our grid further by working with the difference $d_t = m_t - \mu_t$ rather than both values separately: to solve for $V_t(\mu_t, \beta_t, m_t)$, we instead solve for $V'(d_t, \beta_t) = V_t(0, \beta_t, d_t) = V_t(\mu_t, \beta_t, \mu_t + d_t) - \mu_t$ and add μ_t to our solution in order to obtain V_t.

A.3.1 GRID CONSTRUCTION

To construct the grid over stable coordinates, we transform d and β to

$$z = \frac{d}{\tau_t(\beta)}, \quad \ell = \log \beta, \quad \text{where} \quad \tau_t(\beta) = \sqrt{\frac{\beta(\kappa_t + 1)}{\alpha_t \kappa_t}}$$

to form the grid $G_t(z_{t,i}, \ell_{t,j}) = V'(d_t, \beta_t)$ for $t = 0, 1, \ldots, T$, $i = 0, 1, \ldots, N_z$, $j = 0, 1, \ldots, N_\ell$. Note that for observations from $Y \sim t_{2\alpha_t}$ (standard Student-t), we have $X_{t+1} = \mu_t + \tau_t(\beta) Y_{t+1}$. Using the update rules

$$\begin{aligned}
\beta_{t+1} &= \beta_t + \tfrac{\kappa_t}{2(\kappa_t+1)}(X_{t+1} - \mu_t)^2 = \beta_t + \tfrac{\kappa_t}{2(\kappa_t+1)} \cdot \tfrac{\beta_t(\kappa_t+1)}{\alpha_t \kappa_t} Y_{t+1}^2 = \beta_t(1 + \tfrac{Y_{t+1}^2}{2\alpha_t}), \\
\mu_{t+1} &= \tfrac{\kappa_t \mu_t + X_{t+1}}{\kappa_t + 1} = \mu_t + \tfrac{X_{t+1} - \mu_t}{\kappa_t + 1} = \mu_t + \tfrac{\tau_t(\beta) Y_{t+1}}{\kappa_t + 1}, \\
d_{t+1} &= m_{t+1} - \mu_{t+1} = (m_{t+1} - \mu_t) - (\mu_{t+1} - \mu_t) = \max(d_t, \tau_t Y_{t+1}) - \tfrac{\tau_t Y_{t+1}}{\kappa_t + 1} \\
&= \tau_t[\max(z_t, Y_{t+1}) - \tfrac{Y_{t+1}}{\kappa_t + 1}],
\end{aligned}$$

z and ℓ update as $\ell_{t+1} = \ell_t + \log(1 + \frac{Y_{t+1}^2}{2\alpha_t})$ and $z_{t+1} = \frac{d_{t+1}}{\tau_{t+1}} = \frac{\tau_t}{\tau_{t+1}}[\max(z_t, Y_{t+1}) - \frac{Y_{t+1}}{\kappa_t + 1}]$ where $\frac{\tau_t}{\tau_{t+1}} = \sqrt{\frac{(\kappa_t+1)/(\alpha_t \kappa_t)}{(1 + \frac{Y_{t+1}^2}{2\alpha_t})(\kappa_t+2)/((\alpha_t + \frac{1}{2})(\kappa_t+1))}}$ which is independent of β so that each update only depends on the turn and new observation. β updates multiplicatively, so transforming its grid coordinate into ℓ makes the update additive, allowing a linearly spaced grid to have more uniform density across the axis. Transforming d into z normalizes the units of the opposing axis in terms of the current posterior predictive scale and keeps it bounded by not depending on the monotonically increasing β_t.

To choose the grid points, the bounds for the axes are first selected. N_b samples of X_0 are taken from the initialized normal-inverse-gamma distribution and updates for z_1 and ℓ_1 are applied using each sample. The p_{tail} and $1 - p_{\text{tail}}$ quantiles of the resulting z_1 and ℓ_1 parameters are taken as bounds $(z_1^{\min}, z_1^{\max})$ and $(\ell_1^{\min}, \ell_1^{\max})$. For $t = 1, \ldots, T-1$, $[p_{\text{tail}}, 1 - p_{\text{tail}}]$-bounded Gauss-Legendre quadrature nodes $u_1, u_2, \ldots, u_Q$ are taken and $y_q = F_Y^{-1}(u_q)$ for $q = 1, \ldots, Q$ (with F_Y the CDF of $Y \sim t_{2\alpha_t}$) are used as samples to update the already computed $(z_t^{\min}, z_t^{\max})$, $(\ell_t^{\min}, \ell_t^{\max})$ from which the new minimum and maximum values are taken to form $(z_{t+1}^{\min}, z_{t+1}^{\max})$, $(\ell_{t+1}^{\min}, \ell_{t+1}^{\max})$. For the interior grid points, a linearly spaced axis is used for both ℓ and z.

To solve the grid, we proceed backwards from T by calculating

$$G_t(z_{t,i}, \ell_{t,j}) = \max\{d_t, -c + C_t(z_{t,i}, \ell_{t,j})\}$$

with boundary condition $G_T(z_{T,i}, \ell_{T,j}) = \tau_T z_T = d_T$ and continuation value C_t for the expected continuation value. This is computed using $[0, 1]$-bounded Gauss-Legendre quadrature weights $\omega_1, \ldots, \omega_Q$ and nodes $u_1, u_2, \ldots, u_Q$ transformed into the probability space as $y_q = F_Y^{-1}(u_q)$ for $q = 1, \ldots, Q$:

$$C_t(z_{t,i}, \ell_{t,j}) = \sum_{q=1}^{Q} \omega_q[\mu_{t+1} - \mu_t + \widehat{G}_{t+1}(z_{t,i}, \ell_{t,j}, y_q)] = \sum_{q=1}^{Q} \omega_q[\frac{\tau_t y_q}{\kappa_t + 1} + \widehat{G}_{t+1}(z_{t,i}, \ell_{t,j}, y_q)].$$

Here, $\widehat{G}_{t+1}(z_{t,i}, \ell_{t,j}, y_q)$ is an approximation of $G_{t+1}(z', \ell')$, the next-period value at the exact coordinates (z', ℓ') reached by updating $(z_{t,i}, \ell_{t,j})$ from the y_q observation, calculated by bilinear interpolation over the next-period grid. To perform the interpolation, choose a, b such that $z_{t+1,a} \le z' \le z_{t+1,a+1}$ and $\ell_{t+1,b} \le \ell' \le \ell_{t+1,b+1}$. Define the local interpolation weights

$$\rho_z = \frac{z' - z_{t+1,a}}{z_{t+1,a+1} - z_{t+1,a}}, \qquad \rho_\ell = \frac{\ell' - \ell_{t+1,b}}{\ell_{t+1,b+1} - \ell_{t+1,b}}.$$

Then

$$\begin{aligned}
\widehat{G}_{t+1}(z_{t,i}, \ell_{t,j}, y_q) = (1 - \rho_z)(1 - \rho_\ell) G_{t+1}(z_{t+1,a}, \ell_{t+1,b}) + \rho_z(1 - \rho_\ell) G_{t+1}(z_{t+1,a+1}, \ell_{t+1,b}) \\
+ (1 - \rho_z)\rho_\ell G_{t+1}(z_{t+1,a}, \ell_{t+1,b+1}) + \rho_z \rho_\ell G_{t+1}(z_{t+1,a+1}, \ell_{t+1,b+1}).
\end{aligned}$$

A.3.2 Calculating Value Function Using Grid

At each decision time t during the game, the agent can use its sufficient state $(\pi_t, m_t) = (\mu_t, \kappa_t, \alpha_t, \beta_t, m_t)$ to construct a point in the grid space $(z_t, \ell_t) = (\frac{m_t - \mu_t}{\tau_t(\beta_t)}, \log \beta_t)$. Substituting (z_t, ℓ_t) into the same bilinear interpolation procedure used for (z', ℓ') above, now applied to the continuation values over the turn-t axes $(z_{t,a}, \ell_{t,b})$, $C_t(z_t, \ell_t)$ can be approximated as $\widehat{C}_t(z_t, \ell_t)$. Then, the estimated raw stop and explore values $\widehat{Q}$ are

$$\widehat{Q}_t^{\text{stop}}(\pi_t, m_t) = m_t, \qquad \widehat{Q}_t^{\text{explore}}(\pi_t, m_t) = \mu_t - c + \widehat{C}_t(z_t, \ell_t)$$

and the final value function is as originally defined:

$$\widehat{V}_t(\pi_t, m_t) = \max_{a \in \{\text{stop,explore}\}} \widehat{Q}_t^a(\pi_t, m_t).$$

A.4 Proof of Value Function Translation Equivariance

We show that the reduced value function defined in Appendix A.3 satisfies

$$V_t(\mu + \Delta, \beta, m + \Delta) = V_t(\mu, \beta, m) + \Delta$$

for every constant $\Delta \in \mathbb{R}$. Since κ and α are deterministic functions of t, they are identical for the shifted and unshifted states and we suppress them throughout.

We proceed by induction.

Base case. At $t = T$ the game stops, so $V_T(\mu + \Delta, \beta, m + \Delta) = m + \Delta = V_T(\mu, \beta, m) + \Delta$.

Inductive step. Assume $V_{t+1}(\mu + \Delta, \beta, m + \Delta) = V_{t+1}(\mu, \beta, m) + \Delta$ for all μ, β, m. By definition,

$$V_t(\mu, \beta, m) = \max\Big(m, \;\; -c + \mathbb{E}_{X|\mu,\beta}\big[V_{t+1}(\mu', \beta', \max(m, X))\big] \Big),$$

with (μ', β') the posterior after observing X. Note that X doesn't depend on m and is distributed by a normal distribution $N(\mu, \sigma^2)$ where σ^2 doesn't depend on μ, so if this μ is shifted by Δ, then the distribution of X is also shifted to $X + \Delta$. Observing $X + \Delta$ from $(\mu + \Delta, \beta)$ gives

$$\mu'_{\text{shifted}} = \frac{\kappa_t(\mu + \Delta) + (X + \Delta)}{\kappa_t + 1} = \frac{\kappa_t \mu + X + (\kappa_t + 1)\Delta}{\kappa_t + 1} = \frac{\kappa_t \mu + X}{\kappa_t + 1} + \Delta = \mu' + \Delta,$$

$$\beta'_{\text{shifted}} = \beta + \frac{\kappa_t}{2(\kappa_t + 1)}\big((X + \Delta) - (\mu + \Delta)\big)^2 = \beta + \frac{\kappa_t}{2(\kappa_t + 1)}(X - \mu)^2 = \beta',$$

so the posterior mean shifts by Δ while β' is unchanged. Combining these with $\max(m + \Delta, X + \Delta) = \max(m, X) + \Delta$ and expanding the value-function recursion at the shifted state allows us to write $V_t(\mu + \Delta, \beta, m + \Delta)$ as

$$\begin{aligned}
&\max\Big\{m + \Delta, \; -c + \mathbb{E}_{X+\Delta|\mu+\Delta,\beta}\big[V_{t+1}(\mu'_{\text{shifted}}, \beta'_{\text{shifted}}, \max(m + \Delta, X + \Delta))\big]\Big\} \\
&= \max\Big\{m + \Delta, \; -c + \mathbb{E}_{X|\mu,\beta}\big[V_{t+1}(\mu' + \Delta, \beta', \max(m, X) + \Delta)\big]\Big\} \\
&= \max\Big\{m, \; -c + \mathbb{E}_{X|\mu,\beta}\big[V_{t+1}(\mu', \beta', \max(m, X))\big]\Big\} + \Delta \\
&= V_t(\mu, \beta, m) + \Delta.
\end{aligned}$$

This completes the induction. □

A.5 Token Cost Calibration

To set the cost of exploration under the token cost regime, we normalize the turn cost according to how much an LLM thinks when turn costs are in place. More specifically, for each cost level $c \in \{0.15\%, 0.5\%\}$, which corresponds to $c \in \{0.03, 0.1\}$, and agent a, we find the average number of tokens used for each decision $r_{a,c}$ by the reasoning agent across its 200 game episodes in Section 4.2. The costs per token in the token cost games are then set at $c_{\text{tok}} \in \{\frac{0.03}{r_{a,0.03}}, \frac{0.1}{r_{a,0.1}}\}$. This allows the agent to fully recover their performance from the turn cost games if they reason in the same manner, but it also provides them with the ability to adapt their reasoning to the current decision and improve final performance by lowering incurred costs or taking advantage of reduced reasoning costs to explore further. Thus, calibrating costs in this manner *should* be purely to the agent's benefit in maximizing their net reward in the game.

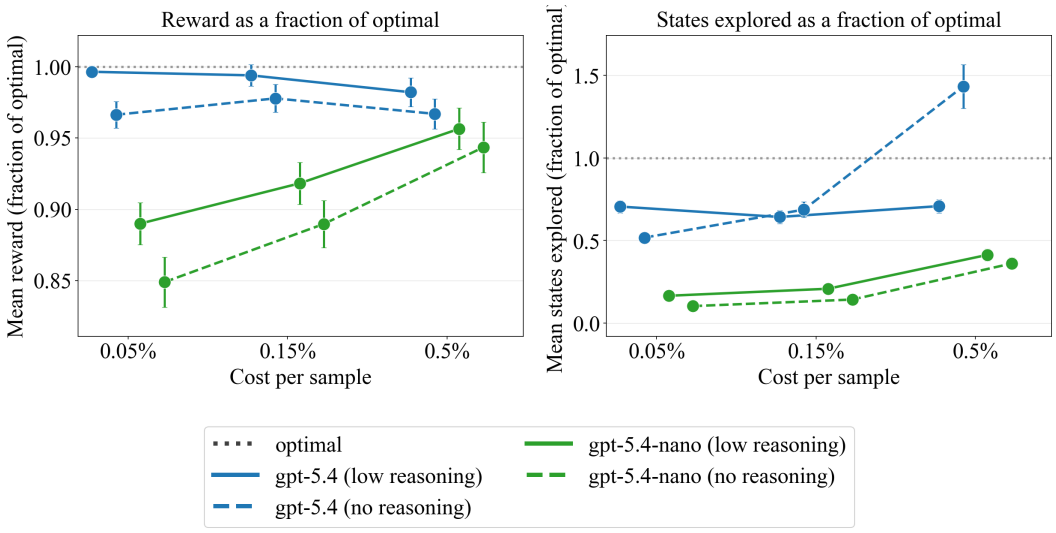


Figure 6: OpenAI model performance as a fraction of the optimal policy: mean net reward (left) and mean states explored (right) across sample costs $c \in \{0.05\%, 0.15\%, 0.5\%\}$. Solid lines are reasoning variants, dashed lines non-reasoning. Error bars are 95% confidence intervals of these fractional estimates.

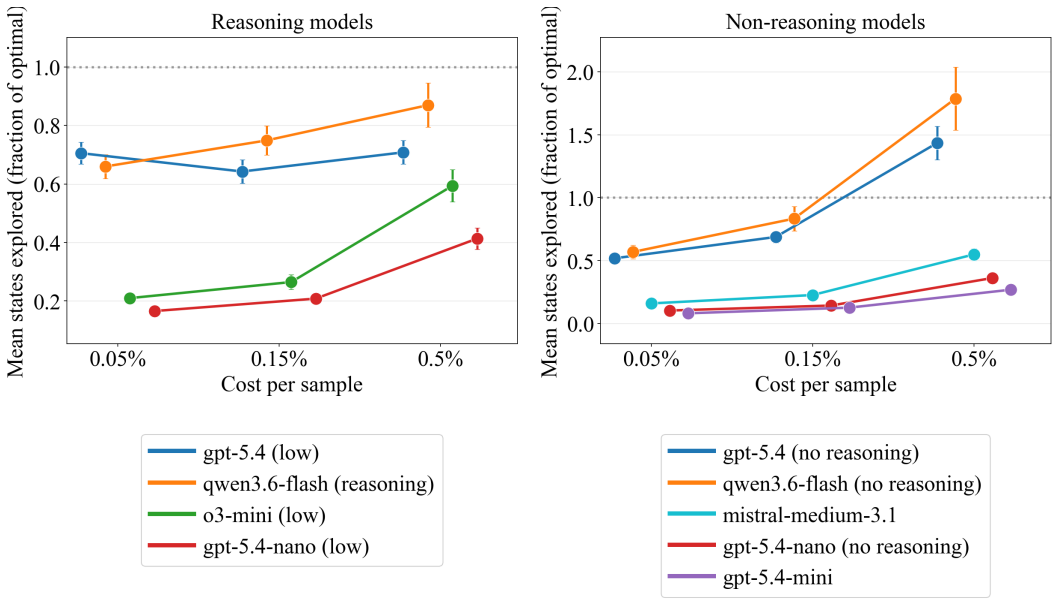


Figure 7: Mean states explored as a fraction of the optimal policy for reasoning models (left) and non-reasoning models (right) across sample costs $c \in \{0.05\%, 0.15\%, 0.5\%\}$. Error bars are 95% confidence intervals of these fractional estimates.

A.6 Plots Normalized to the Optimal Policy

To compare models on a common scale, we also present net reward and states explored normalized to the optimal policy in Section 4.2 Search Game Results in Figure 6 and Figure 7. Each model's value is expressed as a fraction of the Bayes-optimal benchmark facing the same draws, so a value of one corresponds to matching the optimal policy.

Table 2: Mean net reward and mean states explored across models, reasoning vs. non-reasoning variants, and turn vs. token sampling cost. Means are computed over 200 episodes for each sampling cost. Net rewards are normalized by the prior mean of 20. – marks costs that were not run.

	Mean normalized net reward			Mean states explored		
Model	c=0.05%	c=0.15%	c=0.5%	c=0.05%	c=0.15%	c=0.5%
gpt-5.4 (low)	1.605	1.535	1.395	62.4	38.7	17.2
qwen3.6-flash (reasoning)	1.585	1.510	1.415	58.3	45.1	21.1
gpt-5.4 (no reasoning)	1.560	1.510	1.375	45.7	41.3	34.7
qwen3.6-flash (no reasoning)	1.510	1.470	1.295	50.2	50.1	43.3
o3-mini (low)	1.470	1.440	1.385	18.5	15.9	14.4
mistral-medium-3.1	1.440	1.425	1.380	14.1	13.6	13.3
gpt-5.4-nano (low)	1.435	1.415	1.360	14.6	12.5	10.0
gpt-5.4-nano (no reasoning)	1.370	1.370	1.340	9.1	8.6	8.8
gpt-5.4-mini (no reasoning)	1.360	1.355	1.330	7.1	7.6	6.5
inkling-small (high)	1.563	1.520	1.420	42.8	36.0	18.7
inkling-small (no reasoning)	1.350	1.395	1.360	7.9	9.9	9.5
nemotron-3-ultra (medium)	–	1.481	1.420	–	29.6	19.5
nemotron-3-ultra (no reasoning)	–	1.511	1.188	–	91.5	92.0
deepseek-v4-flash (think high)	1.563	1.517	1.408	40.8	30.6	16.2
deepseek-v4-flash (no reasoning)	1.420	1.458	1.302	37.8	43.0	42.0
llama-3.1-nemotron-ultra-253b (reasoning)	1.513	1.446	1.289	43.6	32.0	39.2
llama-3.1-nemotron-ultra-253b (no reasoning)	1.377	1.360	1.336	8.6	7.9	7.7
llama-3.3-nemotron-super-49b (no reasoning)	1.249	1.241	1.247	3.7	3.3	3.5
Token cost (implicit pricing)						
gpt-5.4 (low)	–	1.502	1.378	–	38.4	19.8
gpt-5.4-nano (low)	–	1.436	1.384	–	15.3	15.8
qwen3.6-flash (reasoning)	–	1.517	1.324	–	63.3	51.4
inkling-small (high)	–	1.521	1.400	–	40.3	22.7
nemotron-3-ultra (medium)	–	1.511	1.394	–	40.0	25.5
deepseek-v4-flash (think high)	–	1.442	1.340	–	15.1	8.2

A.7 Mean Reward and States Explored by Model

We report the mean net reward with the reward normalized by $\mu^* = 20$ and mean states explored across sample costs $c \in \{0.05\%, 0.15\%, 0.5\%\}$ and averaged over the $n = 200$ episodes run for each cost. We also report reasoning models from implicit token pricing. Nemotron-3-ultra and the token-cost arms were run only on $c \in \{0.15\%, 0.5\%\}$.

A.8 Reasoning Effort Levels

Figure 8 compares GPT-5.4 Nano across reasoning effort levels using 200 games at each sampling cost, $c \in \{0.05\%, 0.15\%, 0.5\%\}$. We estimate the cost elasticity of exploration as the negative slope of the log-log relationship between states explored and sampling cost, with effort and seed fixed effects, and average normalized net reward across the three costs. Both elasticity and reward rise monotonically from no reasoning through high effort: elasticity increases from 0.02 to 0.48, approaching but remaining below the Bayes-optimal benchmark of 0.62. Thus, greater reasoning effort produces more cost-sensitive exploration and moves performance closer to the optimal policy.

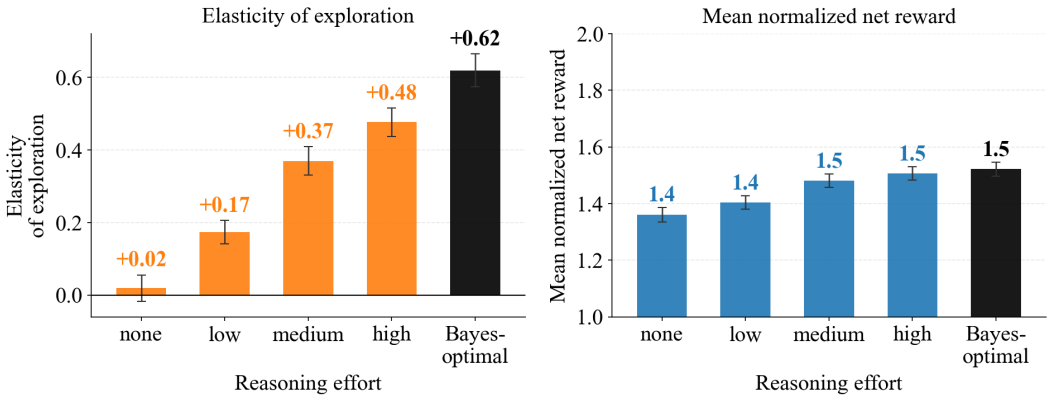


Figure 8: GPT-5.4 Nano across reasoning effort levels, with the Bayes-optimal policy (black) as the rightmost bar in each panel. Left: cost elasticity of exploration. Right: net reward (normalized by the prior mean of 20) averaged over the three sampling costs $c \in \{0.05\%, 0.15\%, 0.5\%\}$ (the axis does not start at zero). Both elasticity and reward rise monotonically from no reasoning through high effort, and both remain below the optimal benchmark. Error bars are 95% CIs: ± 1.96 seed-clustered standard errors from a single regression of $\log(\text{states})$ on a per-effort $\log(\text{cost})$ slope with effort and seed fixed effects for the elasticity, and normal intervals for the cost-averaged reward.

A.9 Net Reward Decomposition

To understand what is driving the performance shifts of agents between sample costs, we decompose agents' difference in net reward at different costs into the change in exploration costs they incur and the change in the best reward they observe. If an agent receives net reward r_i at cost c_i from a maximum realized reward m_i and number of states explored s_i, then we can decompose their net reward change from r_1 to r_2 by

$$r_2 - r_1 = \underbrace{m_2 - m_1}_{\text{change in reward}} + \underbrace{c_1 s_1 - c_2 s_2}_{\text{net cost effect}}.$$

Figure 9 displays this decomposition for agents going from $c = 0.5\%$ to $c = 0.05\%$ and shows that reasoning models' and the optimal policy's boost in performance is much less driven by the net cost effect than for non-reasoning models. This is due to the fact that reasoning models and the optimal policy explore more as the cost falls while non-reasoning models are nearly cost-invariant in this respect. These results explicitly demonstrate that reasoning models' heightened cost sensitivity allows their behavior between costs to shift more similarly to the optimal policy than their non-reasoning counterparts.

A.10 Cost Elasticity Decomposition under Token Pricing

The gap between the cost elasticities of the explicit and implicit arms can be decomposed using Equation 4. Because the price per token is normalized to the explicit arm's tokens per turn, the one-for-one effect of the price cancels between the two arms, and $\varepsilon_C^E - \varepsilon_C^I = -(\varepsilon_r^I - \varepsilon_r^E) + (\varepsilon_\tau^E - \varepsilon_\tau^I)$. The first term captures how tokens per turn respond to cost relative to the price normalization, and the second term captures the difference in exploration elasticities.

Panel (a) of Figure 10 compares the overall cost elasticities under explicit and implicit pricing for each model. Panel (b) focuses on the within-model difference between those elasticities and its decomposition, with the black bar showing their total, corresponding to the gap between the blue and orange bars in Panel (a).

A.11 Gross Reward Elasticity

Figure 11 reports the gross reward elasticity of cost, $\varepsilon_m = -d \log m / d \log c$, where m is the best reward observed before exploration costs are subtracted. The explicit minus implicit difference splits

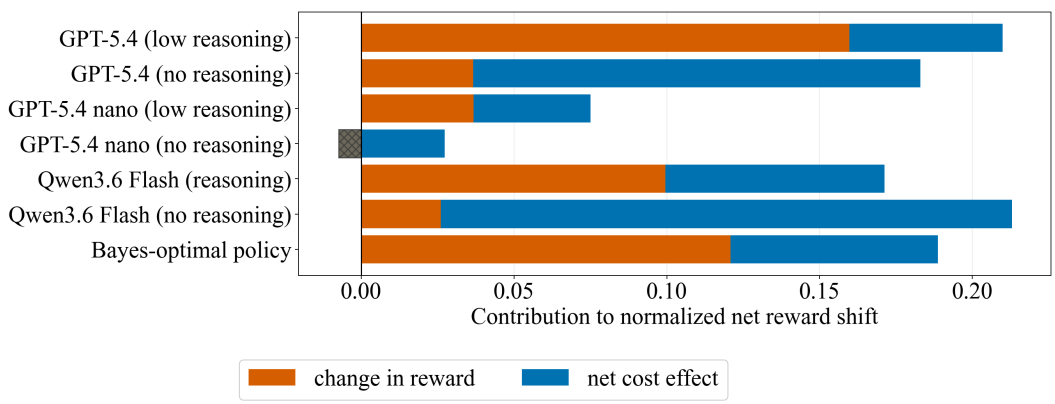


Figure 9: Decomposition of the average change in normalized net reward as the sampling cost falls from $c = 0.5\%$ to $c = 0.05\%$. Orange bars show the change in the best observed reward and blue bars the net cost effect; their sum is the total change in net reward. Within each model family, the reasoning variant is shown above the non-reasoning variant. Crosshatching marks the region in which components with opposite signs cancel. Reward contributions are normalized by the prior mean $\mu^* = 20$.

into an exploration channel, the change in the expected best of the draws taken under each game's data-generating process implied by the change in exploration, summed turn by turn, and a residual, which reflects which draws the agent chooses to stop on.

A.12 Semi-Elasticities of Gross Reward, Search Cost and Net Reward

Because net reward is gross reward minus search cost, $U = m - C$ (Equation 1), its semi-elasticity with respect to the per-sample cost is the sum $dU/d\log c = dm/d\log c - dC/d\log c$. Figure 12 reports the three terms, with rewards and costs expressed per prior mean reward.

A.13 Dollar Cost of Experiments across Models

Table 3 reports the reasoning cost of one simulation (a single game of up to $T = 100$ turns) for the reasoning variant of every model in Figure 3, under both explicit (turn) and implicit (token) pricing, together with DeepSeek-V4-Flash (Think High). Prompt tokens are not included in the calculation. All entries are calculate from 200 games at each of the $c \in \{0.15\%, 0.5\%\}$ costs of exploration. Output tokens are priced at OpenRouter's latest output rate from September 2026.

B WebShop Experiments

B.1 Experimental Configuration

WebShop is an environment where agents are trained and evaluated on shopping according to user instructions. Agents on WebShop are provided with a user instruction that contains a series of constraints on product options, attributes, and price. They have the ability to query products in a search bar, inspect products for their characteristics, and choose whether or not to buy a product. In contrast to the original work, we charge a series of exploration costs for viewing a product and its characteristics, and these costs match the exploration costs from the primary experiments relative to the average price in that simulation's search results. We instruct the LLM to try and adhere to the price constraint given, inclusive of these costs. In the original work, agents were required to construct a search query based on the user instruction, but we provide them with a query and the corresponding search results *a priori* in order to achieve a closer comparison to our primary experiments. We select 200 random user instructions and construct a query for each instruction with Claude Opus 5. We ensure that there is at least one product that fully matches the constraints, including our cost-inclusive price constraint, in the search results. Exploration costs for token cost configurations are set in

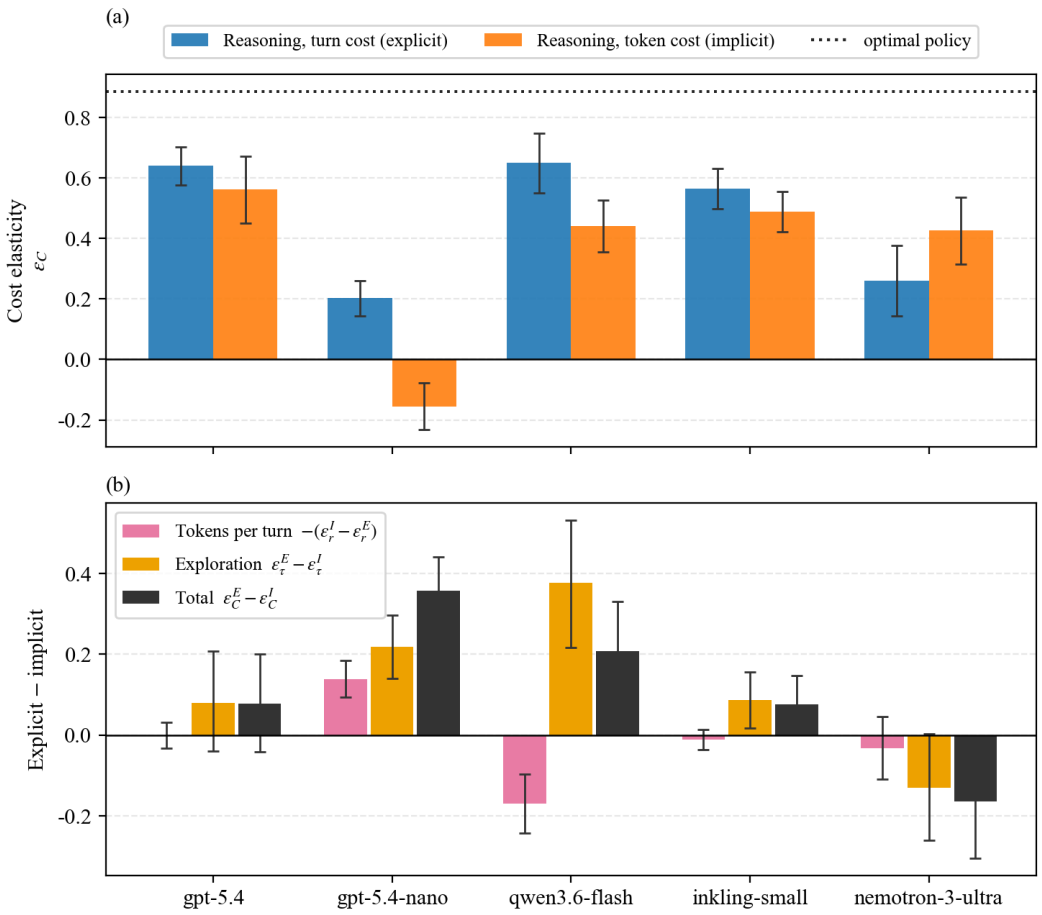


Figure 10: (a) Cost elasticity $\varepsilon_C = 1 - d\log C/d\log c$ of total exploration cost for each model's reasoning variant under explicit (turn, blue) and implicit (token, orange) pricing. The blue bars equal the blue bars of Figure 3, since $\varepsilon_C^E = \varepsilon_\tau^E$. The orange bars show the implicit-pricing cost elasticity $\varepsilon_\tau^I + \varepsilon_r^I - \varepsilon_r^E$ (Equation 4); the dotted line is the Bayes-optimal policy. (b) The explicit minus implicit difference, decomposed as in Equation 4 into the response of tokens per turn relative to the price normalization, $-(\varepsilon_r^I - \varepsilon_r^E)$, and the difference in exploration elasticities, $\varepsilon_\tau^E - \varepsilon_\tau^I$. Error bars in (a) are 95% confidence intervals from standard errors clustered by seed (Appendix F); intervals in (b) are from a paired bootstrap over seeds.

the same manner as described in Section A.5 but instead across WebShop simulations for a given model-cost combination. Prompts used for generating search queries and prompts shown to agents in the game are provided in Appendix E.5.

B.2 Full Results

Table 4 reports the mean number of products viewed for every model and pricing arm shown in Figure 5. Each entry averages over the 200 games run at the relevant cost.

B.3 Cost Elasticity Decomposition under Token Pricing

Figure 13 applies the decomposition of Equation 4 to WebShop, as Figure 10 does for the primary experiments. Here τ is the number of products reviewed and r the billed tokens per product reviewed. The price per token is normalized to the explicit arm's tokens per response at each cost, so $\varepsilon_C^E = \varepsilon_\tau^E$ and $\varepsilon_C^I = (\varepsilon_r^I - \varepsilon_r^E) + \varepsilon_\tau^I$ hold exactly. Like Figure 5 and Table 4, this figure uses the extended set of

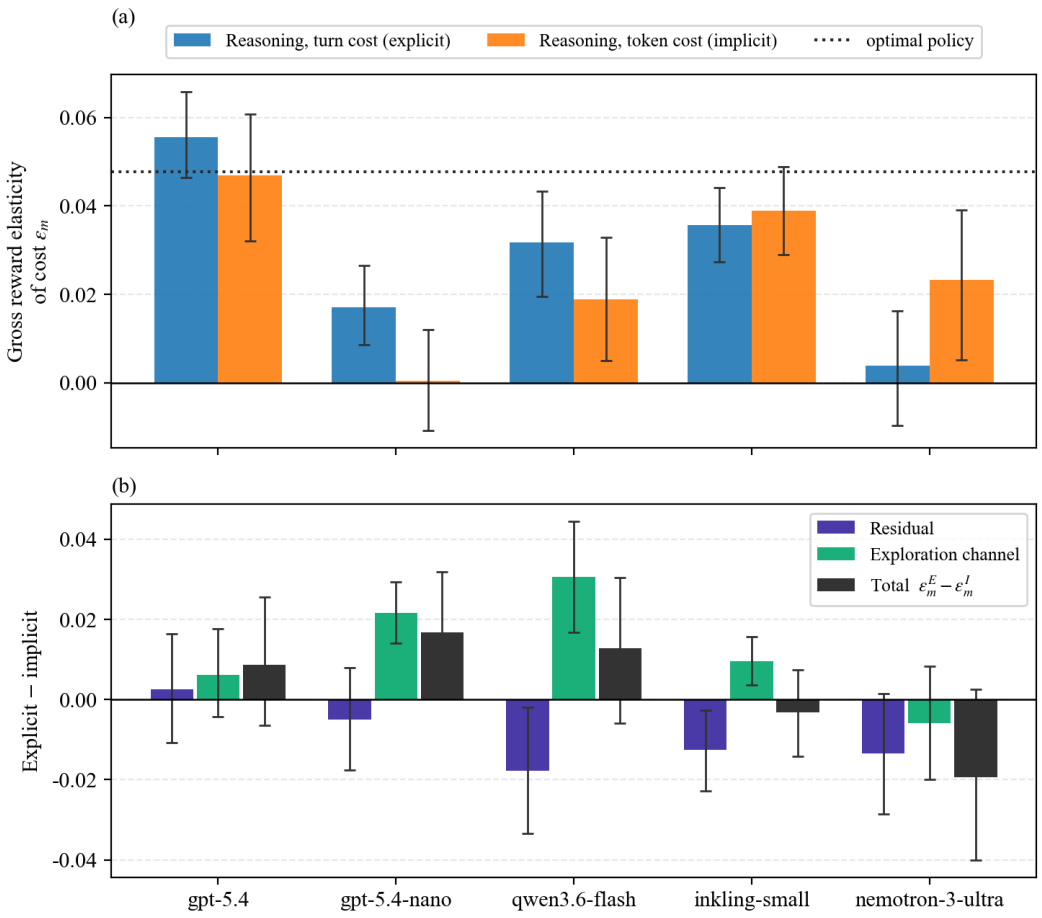


Figure 11: (a) Gross reward elasticity of cost $\varepsilon_m = -d\log m/d\log c$ for each model's reasoning variant under explicit (turn, blue) and implicit (token, orange) pricing; the dotted line is the Bayes-optimal policy. (b) The explicit minus implicit difference, split into the exploration channel and the residual. Elasticities are seed-matched log changes between $c = 0.15\%$ and 0.5% over the same 200 seeds; error bars are 95% intervals from a paired bootstrap over seeds.

200 instructions (the original 50 and 150 more) for the five reasoning arms. Elasticities are estimated over all three cost levels.

C Testing for Understanding

C.1 Update Priming Prompt Ablation

In our LLM system prompt we instruct that

> **System prompt**
>
> Your confidence intervals on the distribution should shrink as you observe more samples, and you should update your prior over mu and sigma^2 with each new value you observe.

in order to induce the priors we want the model to use properly. To test whether this instruction has a tangible effect on model performance, we run an experiment with this sentence ablated from the system prompt on a set of non-reasoning, reasoning, and scratchpad reasoning models (details in

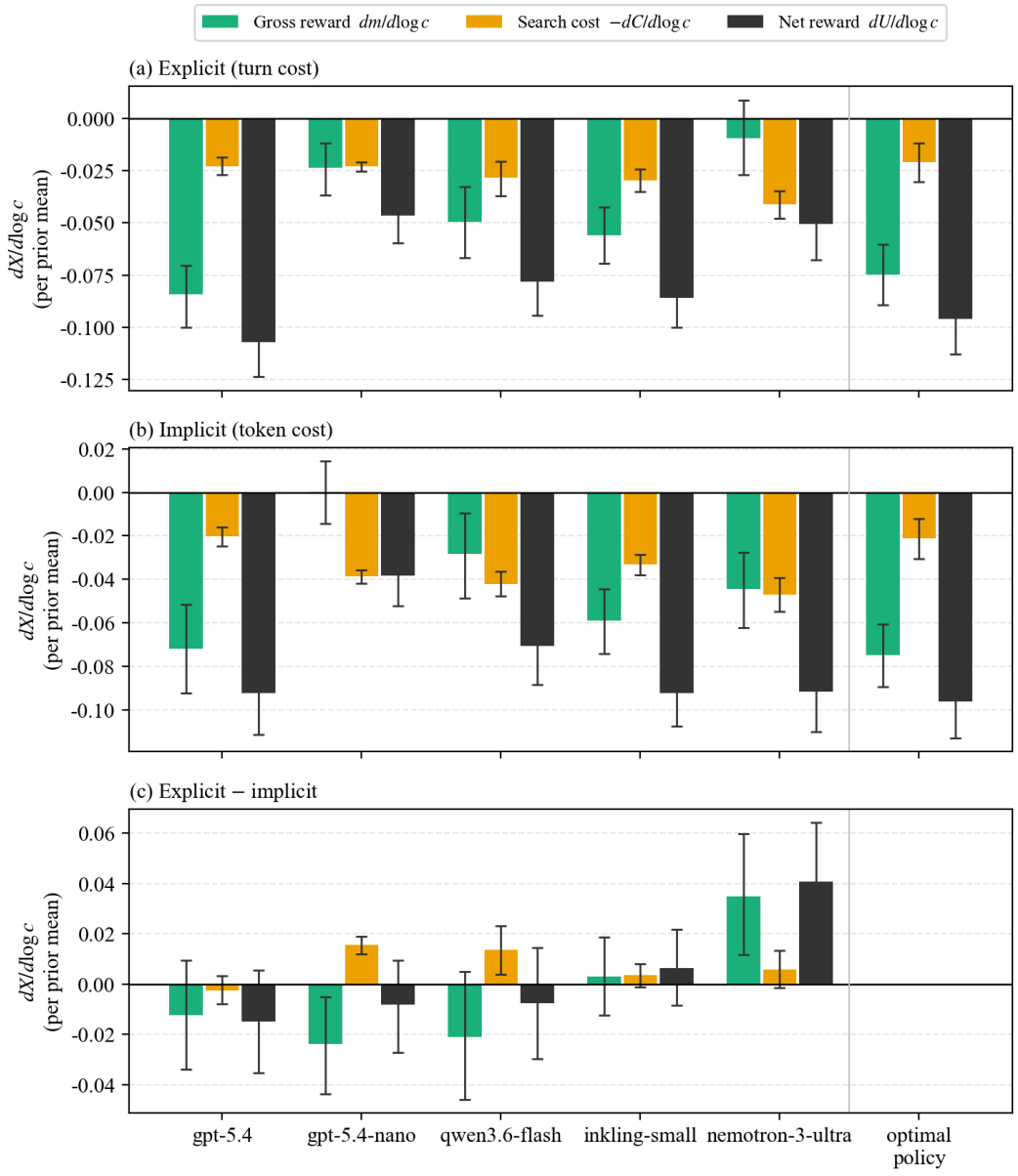


Figure 12: Semi-elasticities with respect to the per-sample cost, $dX/d\log c$, of gross reward m, search cost C (entered as $-dC/d\log c$), and net reward $U = m - C$, which add exactly, for (a) explicit and (b) implicit pricing and (c) their difference. Rewards and costs are expressed per prior mean reward. Costs are those charged in the game: $c\tau$ under explicit pricing and the token charge under implicit pricing. The last group is the Bayes-optimal policy. Estimated between $c = 0.15\%$ and 0.5% over the same 200 seeds; error bars are 95% intervals from a paired bootstrap over seeds.

Table 3: Average number of reasoning tokens and reasoning dollar cost per simulation for reasoning models. Values are computed from the $c \in \{0.15\%, 0.5\%\}$ costs of exploration to keep a shared set of costs over turn vs. token cost agents. Final dollar costs are based upon the most recent API pricing from OpenRouter (September 2026, US dollars per million output tokens).

Model	Reasoning	Pricing	Output price ($/1M)	Reasoning tokens per sim.	Reasoning $ per sim.
gpt-5.4	Low	Turn	15.00	15,233	0.2285
gpt-5.4	Low	Token	15.00	9,399	0.1410
gpt-5.4-nano	Low	Turn	1.25	4,133	0.0052
gpt-5.4-nano	Low	Token	1.25	4,276	0.0053
qwen3.6-flash	On	Turn	1.125	114,158	0.1284
qwen3.6-flash	On	Token	1.125	95,496	0.1074
inkling-small	High	Turn	1.20	100,911	0.1211
inkling-small	High	Token	1.20	85,606	0.1027
nemotron-3-ultra	Medium	Turn	2.40	111,967	0.2687
nemotron-3-ultra	Medium	Token	2.40	157,851	0.3788
deepseek-v4-flash	Think High	Turn	0.097	58,930	0.0057
deepseek-v4-flash	Think High	Token	0.097	76,926	0.0075

Table 4: Mean products viewed for the WebShop experiments across models, reasoning vs. non-reasoning, and turn vs. token sampling cost. Means are computed over 200 instructions.

	Mean products viewed		
Model and arm	c=0.05%	c=0.15%	c=0.5%
qwen3.6-flash, non-reasoning (turn)	45.6	45.8	46.8
qwen3.6-flash, reasoning (turn)	32.6	30.8	27.9
qwen3.6-flash, reasoning (token)	32.5	30.0	26.7
gpt-5.4, no reasoning (turn)	42.1	41.3	38.0
gpt-5.4, low reasoning (turn)	39.5	37.4	33.3
gpt-5.4, low reasoning (token)	25.7	23.6	20.0
gpt-5.4-nano, no reasoning (turn)	45.6	45.9	45.6
gpt-5.4-nano, low reasoning (turn)	33.6	30.8	31.2
gpt-5.4-nano, low reasoning (token)	29.4	25.9	22.3
nemotron-3-ultra, no reasoning (turn)	49.6	49.6	49.6
nemotron-3-ultra, medium reasoning (turn)	36.6	35.5	32.3
nemotron-3-ultra, medium reasoning (token)	34.9	33.6	29.9
inkling-small, no reasoning (turn)	36.3	34.6	36.7
inkling-small, high reasoning (turn)	37.5	36.9	35.4
inkling-small, high reasoning (token)	25.5	24.0	22.2

Appendix D.1). The results are displayed in Figure 14. For all models except for Qwen3.6 Flash (reasoning), the ablation has an inconsistent or negligible effect on both reward and exploration. For Qwen3.6 Flash (reasoning), we see a drop in both. In Table 5, we additionally see that this model uses considerably fewer tokens when the instruction is removed, suggesting that it is not reasoning as much about updating its beliefs. Its cost elasticity of exploration computed over the tested exploration costs drops only from 0.65 to 0.52, however, and we see that it obtains similar net reward. More broadly, this indicates that the update priming instruction is not a major driver of model performance, although it may have some effect, and it should not affect the interpretation of our results substantially.

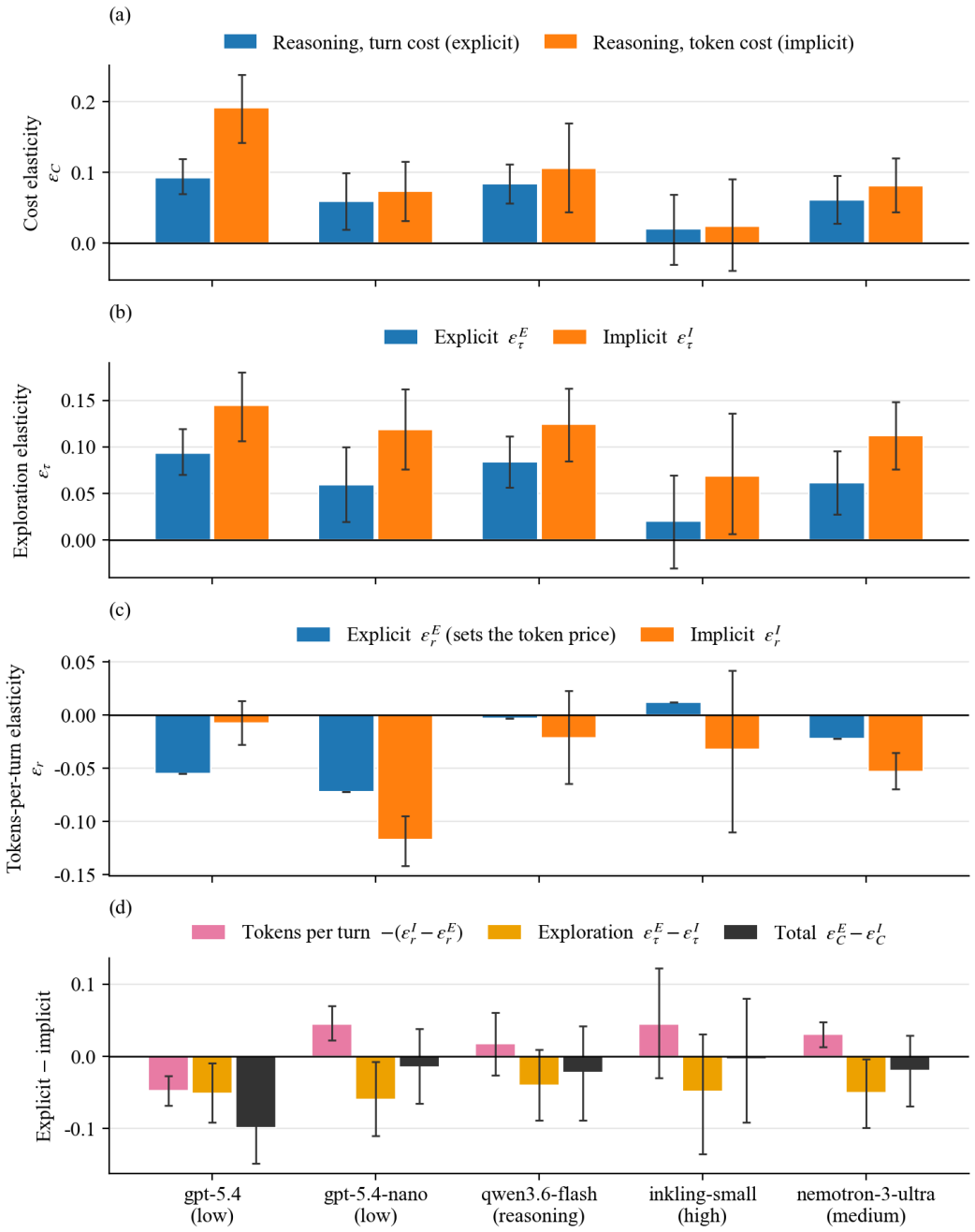


Figure 13: Cost elasticity decomposition in WebShop for each model's reasoning variant under explicit (turn, blue) and implicit (token, orange) pricing, with $\varepsilon_x = -d\log x/d\log c$. (a) Cost elasticity $\varepsilon_C = 1 - d\log C/d\log c$ of total exploration cost. (b) Exploration elasticity ε_τ, with τ the number of products reviewed. (c) Tokens-per-turn elasticity ε_r; ε_r^E is read from the quoted per-token prices and sets the price schedule. (d) The explicit minus implicit difference $\varepsilon_C^E - \varepsilon_C^I$, split into $-(\varepsilon_r^I - \varepsilon_r^E)$ and $\varepsilon_\tau^E - \varepsilon_\tau^I$. Each elasticity is the mean over the 200 instructions of the within-instruction slope of the log quantity on $\log c$ across $c \in \{0.05\%, 0.15\%, 0.5\%\}$; error bars are 95% intervals from a paired bootstrap over instructions. Because ε_r^E comes from the fixed price schedule, its intervals are near zero.

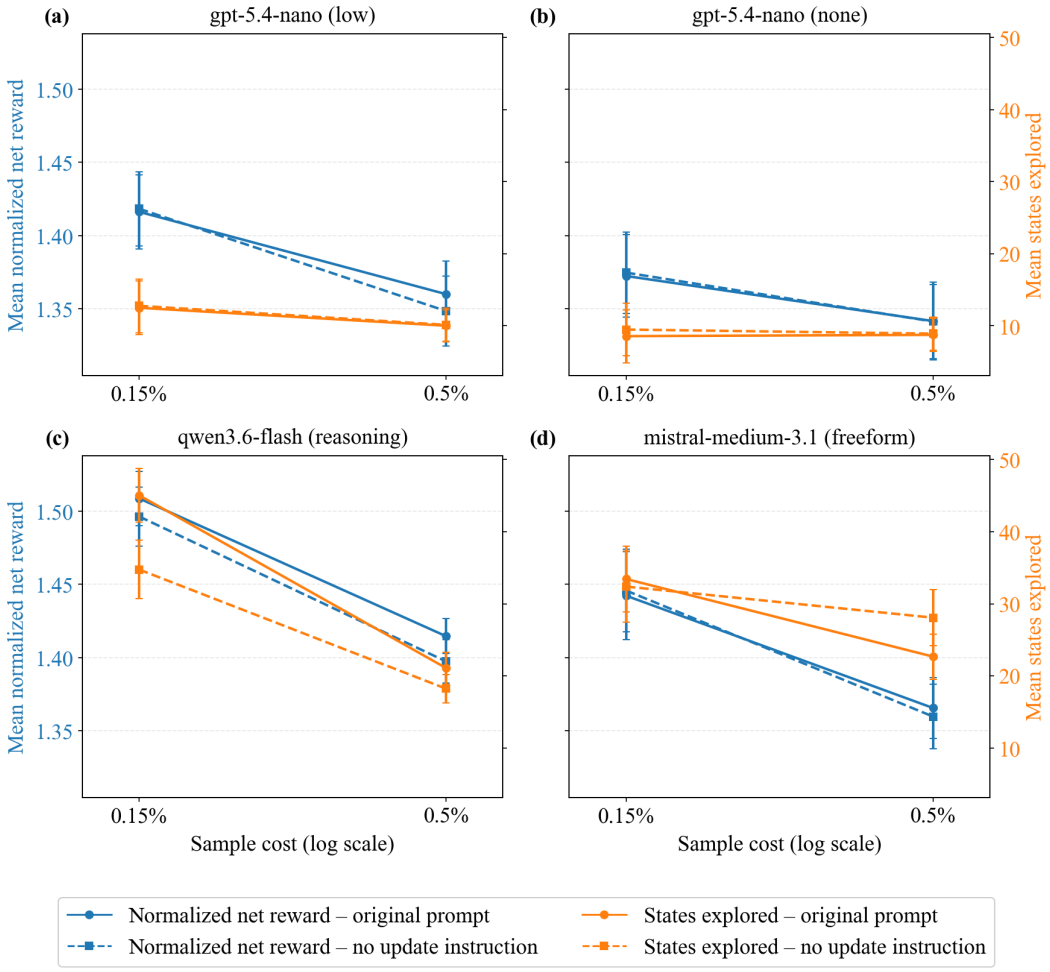


Figure 14: Mean net reward normalized by the prior mean of 20 (left axis, blue) and mean states explored (right axis, orange) across 200 games at sample costs $c \in \{0.15\%, 0.5\%\}$ for (a) GPT-5.4 nano (low), (b) GPT-5.4 nano (no reasoning), (c) Qwen3.6 Flash (reasoning), and (d) Mistral Medium 3.1 (freeform scratchpad). Solid lines show the original prompt and dashed lines show the prompt with the belief-updating instruction removed. Error bars are obtained in the same manner as in Figure 2.

C.2 Tests for Environmental Understanding

To assess how well LLMs understand the task environment, we ask them six questions in which they are instructed to update their beliefs based on observations of varying degrees of likelihood under the initial belief state[4]. These questions are asked using the same system prompt as the abstract scenario which describes the environment in the actual game. Complete questions and the system prompt are shown in Appendix E.6. Table 6 displays the results for two examples: GPT-5.4 (low) can perform these updates perfectly while GPT-5.4 nano (no reasoning) achieves only 79% accuracy. More broadly, we find that reasoning models are always able to perform the belief updating correctly while non-reasoning models sometimes fail. GPT-5.4 nano (no reasoning) particularly struggles with updating the β parameter, which dictates the scale of the prior variance and has the only nonlinear update amongst the belief parameters. This performance difference between models suggests that non-reasoning models tend to be less capable at performing the tasks necessary to maintain an effective decision-making strategy and may fail to comprehend the data generating process as a whole.

C.3 Continuation Value Investigation

To understand if agents truly exhibit recursive reasoning, we must test more explicitly whether they value exploration beyond the gains it provides to net reward from that turn alone. The value of exploration can be decomposed into three distinct components: improvements in the best reward from the immediate step, improvements from continued exploration, and improvements in the agent's beliefs about the underlying reward distribution. To isolate the first two components, we run an ablation where we explicitly tell the LLM the reward distribution and construct a *myopic policy* which only values the immediate increase in reward to compare the LLM to in this modified environment. Specifically, the myopic policy is identical to the optimal policy except with $Q_t^{\text{explore}}(\pi_t, m_t) = -c + \mathbb{E}_{X_{t+1}\sim p(\cdot|\pi_t)}\left[\max\{m_t, X_{t+1}\}\right]$ and π_t having a perfect belief about the reward distribution.

In order for the myopic and optimal policies to diverge in practice when Bayesian learning is not present, we let the cost of exploration decline exponentially over each turn and set the time horizon to 200 turns. We then set the exploration cost and its rate of decline based upon the first reward drawn in the simulation so that the optimal policy always explores at the first turn while the myopic policy stops. We test which policy LLMs agree with on that turn and show the results in Table 7. These results initially indicate that most models are not myopic, even those who perform poorly and are cost insensitive in the main experiments.

This experiment does not symmetrically compare the myopic and optimal policies, however, since the myopic policy always stops on the first turn, whereas the optimal policy always explores. To test whether some of the models which side with the optimal policy in Table 7 truly value continued exploration, we run an additional experiment where we switch the exploration cost back to a constant so that the optimal and myopic policies agree and make the cost exceedingly high or low so that the best action is clear. These costs are defined such that the difference between the value of stop and value of explore is $\pm 0.5\sigma$ for σ the standard deviation of the reward distribution. The results are shown in Table 8 and indicate that some of the models which appeared to side with the optimal policy in Table 7, such as mistral-medium-3.1, are actually just prone to early exploration. Other models like gpt-5.4 (low) look to truly value continued exploration based upon their performance in both experiments. Overall, these results reasonably demonstrate that models which perform well on both experiments are reasoning recursively and those which perform poorly on the first but well on the second are not, but we cannot make as conclusive a claim for models which perform poorly on the second experiment. It is additionally important to note that though these experiments isolate the behavior of interest, they may not generalize to our main experiments since they are performed with a known reward distribution and on only the first turn.

C.4 Decision-Level Value Analysis

To better evaluate models' explore-or-stop decisions beyond net reward, we define the value loss when the model chooses the lower-value action,

$$\ell_t = \max\left\{Q_t^*(\text{Explore}),\, Q_t^*(\text{Stop})\right\} - Q_t^*\left(a_t^{LLM}\right),$$

[4]Environmental understanding parameters and their belief updating rules are defined in Appendix A.2.

Table 5: Mean completion tokens per decision (including hidden reasoning tokens) for each model at each sample cost on the original prompt and with the belief-updating instruction removed. Entries marked "-" are shown for non-reasoning models which do not produce any reasoning tokens.

	Original prompt		No update instruction	
Model	$c = 0.15\%$	$c = 0.5\%$	$c = 0.15\%$	$c = 0.5\%$
gpt-5.4-nano (low)	371	362	328	362
gpt-5.4-nano (no reasoning)	-	-	-	-
mistral-medium-3.1 (freeform)	669	1019	647	828
qwen3.6-flash (reasoning)	3117	4174	2611	2942

Table 6: Per-parameter one-shot accuracy of normal-inverse-gamma posterior updates on the data generating process (DGP) comprehension probe. Each question observes one (Q1–Q3) or five (Q4–Q6) samples whose values are common, borderline high, or extremely high under the initial beliefs. Responses are judged correct if they are within 1% of the true analytic updated value. A check mark (✓) in the table indicates that the model updated that parameter ($\mu_0, \kappa, \alpha, \beta$) correctly while a cross (×) indicates an incorrect update.

			GPT-5.4 (low)				GPT-5.4 nano (no reasoning)			
Row	Samples	Likelihood	μ_0	κ	α	β	μ_0	κ	α	β
1	1	Common	✓	✓	✓	✓	✓	✓	✓	✓
2	1	Borderline high	✓	✓	✓	✓	✓	✓	✓	✓
3	1	Extreme high	✓	✓	✓	✓	✓	✓	✓	✓
4	5	Common	✓	✓	✓	✓	✓	✓	✓	×
5	5	Borderline high	✓	✓	✓	✓	✓	✓	✓	×
6	5	Extreme high	✓	✓	✓	✓	×	✓	×	×

and the absolute value gap, which measures how much value separates the better action from the worse one,

$$\Delta_t = |Q_t^*(\text{Explore}) - Q_t^*(\text{Stop})| \,.$$

In Figure 15, we show the expected value loss when those decisions differ from the Bayes-optimal policy. Across all three sampling costs, each reasoning variant incurs less mean value loss than its non-reasoning counterpart. In Figure 16, we further find that while the accuracy of LLMs in choosing the optimal action generally rises with the absolute value gap, reasoning variants generally lead at small and moderate gaps, while both variants approach very high accuracy at large gaps.

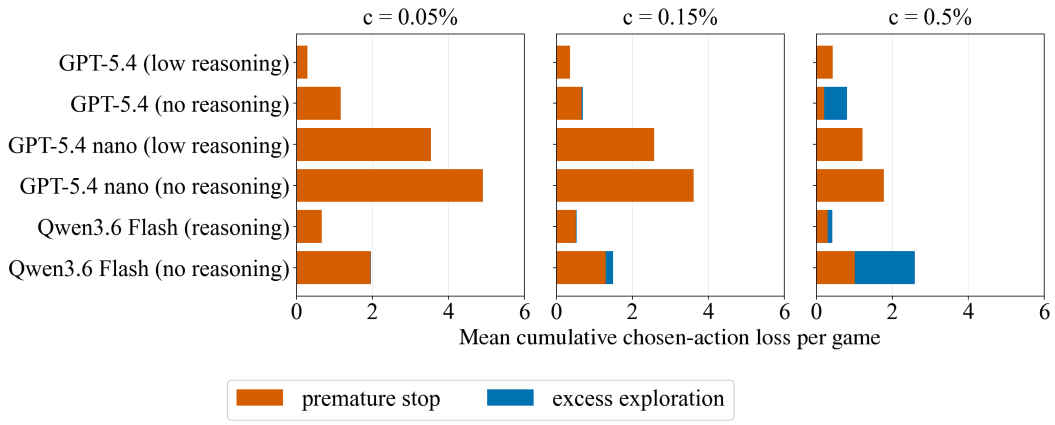


Figure 15: Chosen-action value loss across sampling costs. Value loss is computed as a sum of the value gap under the optimal policy for all agent decisions which disagree with the optimal policy. This is summed within games and averaged across games to produce the results.

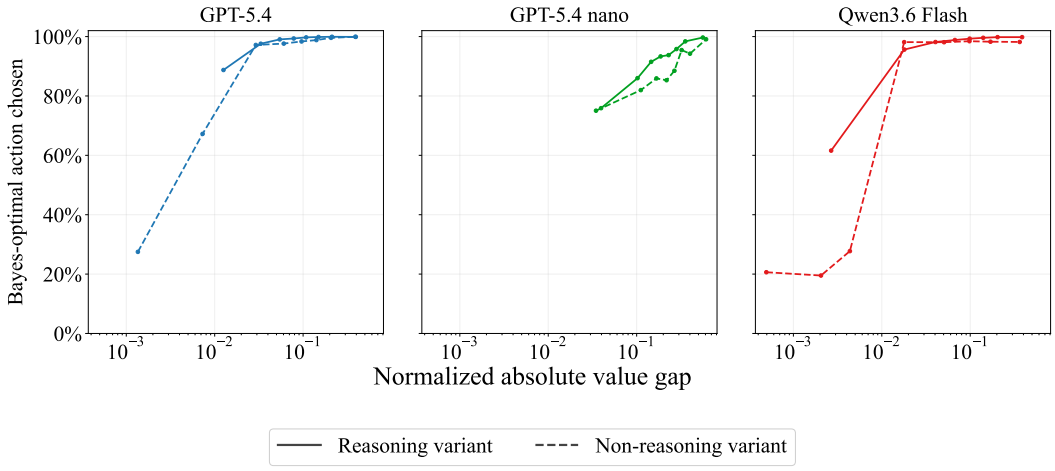


Figure 16: Percentage of agent actions which follow the Bayes-optimal policy, separated by the gap between the value of EXPLORE and STOP under the optimal policy. Results are pooled across sampling costs and placed into eight equal-frequency bins of $\Delta_t/20$. Bin boundaries are model specific. Solid lines show results for reasoning models whereas dashed lines show non-reasoning variants.

Table 7: Fraction of declining-cost games in which each model's first-turn action is explore, agreeing with the optimal policy over the myopic policy. Each of the 200 games tested for each model fixes a known reward distribution $N(20, \sigma^2)$ at each of ten different standard deviations $\sigma \in \{1, 2, 3, 5.27, 7.5, 10, 15, 20, 30, 50\}$ with 20 initial draws.

Model	Explore fraction
gpt-5.4-nano (no reasoning)	1.000
mistral-medium-3.1	1.000
gpt-5.4 (low)	0.995
gpt-5.4-mini (no reasoning)	0.940
qwen3.6-flash (no reasoning)	0.935
gpt-5.4 (no reasoning)	0.930
qwen3.6-flash (reasoning)	0.685
gpt-5.4-nano (low)	0.390
o3-mini (low)	0.385

Table 8: Accuracy in the constant-cost control games ($n = 200$ games per condition for each model). The same games as Table 7 are run with a constant exploration cost, under which the optimal and myopic policies always agree, so there is a single correct action.

Model	Explore accuracy (low cost)	Stop accuracy (high cost)
gpt-5.4 (low)	1.000	0.975
gpt-5.4-nano (low)	1.000	0.975
qwen3.6-flash (reasoning)	1.000	0.840
gpt-5.4 (no reasoning)	0.965	0.400
o3-mini (low)	0.975	0.375
gpt-5.4-mini (no reasoning)	0.985	0.015
gpt-5.4-nano (no reasoning)	1.000	0.005
mistral-medium-3.1	1.000	0.000
qwen3.6-flash (no reasoning)	1.000	0.000

C.5 Joint Test of Constrained Recursive Reasoning

The experiments in previous sections examine cost sensitivity, belief updating, and continuation value largely separately. We next test whether LLMs still choose the Bayes-optimal action when all three matter at the same decision. Using states from the main experiment, we select those where separate distortions to the posterior mean, posterior uncertainty, sample cost, and continuation value each reverse the Bayes-optimal action. We calibrate each distortion to produce a typical shift of 0.10c, 0.15c, or 0.20c in the value gap of EXPLORE relative to STOP. We then test whether the LLM's recorded EXPLORE/STOP decision agrees with the Bayes-optimal policy in these states.

Figure 17 shows the difference in reasoning and non-reasoning models' agreement with the optimal policy at $c = 0.5\%$. We report only the sample-cost condition of $c = 0.5\%$ because, at costs $c = 0.05\%$ and $c = 0.15\%$, several models consistently reach few or no states satisfying the selection criteria, making comparisons uneven. When the cost of exploration is low, it is rare that EXPLORE will not be chosen even if the agent has very pessimistic beliefs or a low continuation value since the cost of another draw is extremely low. States which qualify generally occur later at turns after many models have already stopped. The figure shows that reasoning models generally outperform their non-reasoning variants. The figure also decomposes agreement according to whether the optimal policy recommends Explore or Stop. All reasoning models are much better at stopping. These results show some initial behavioral evidence that reasoning models are likely to consider cost sensitivity, belief updating, and continuation value jointly in our search game.

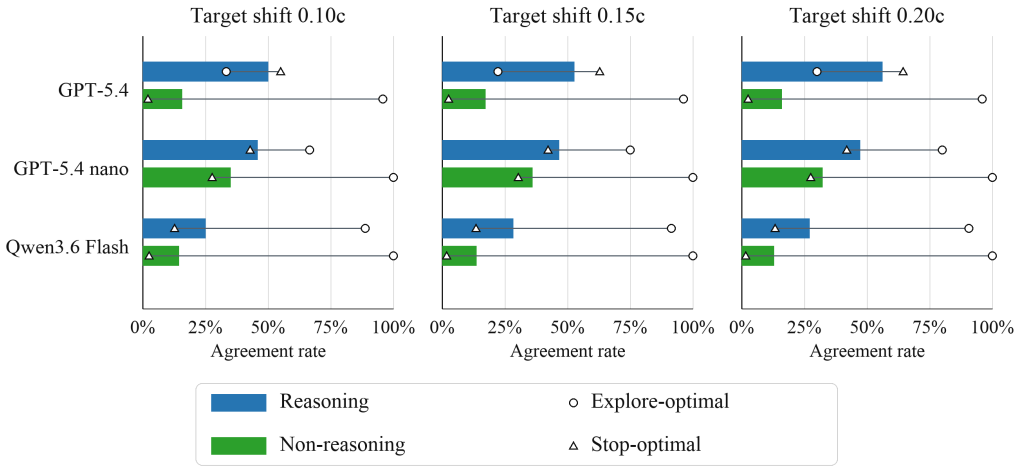


Figure 17: Agreement rate with the optimal policy for states which satisfy joint diagnostic constraints. Results shown are for the sampling cost $c = 0.5\%$ on reasoning vs. non-reasoning variants. Results are decomposed by whether the optimal action is STOP or EXPLORE.

C.6 Equivalent Parameter Sensitivities to Cost

In our primary experiments, we study agents' cost sensitivity, but there are other game parameters to which the model's exploration should be elastic, such as belief parameters and time horizon. The β parameter in particular, which dictates the prior magnitude of the variance of the reward, can be scaled by $\frac{1}{\gamma^2}$ to create an identical transformation to the optimal policy as scaling the cost by γ. Given this property, agents should have identical policies under these equivalent transformations. We test this by using the $c = 0.15\%$ game configuration and scaling β up and down such that the optimal policy matches the policy for costs $c = 0.05\%$ and $c = 0.5\%$. We simulate using Qwen3.6 Flash and DeepSeekV4 Flash as these are the cheapest reasoning models which exhibit some cost sensitivity and report the results in Figure 18. We see that for each model the reasoning variants more closely track their original behavior under the equivalent scaling, demonstrating that these models are adjusting their behavior to the game environment in a consistent manner beyond just cost.

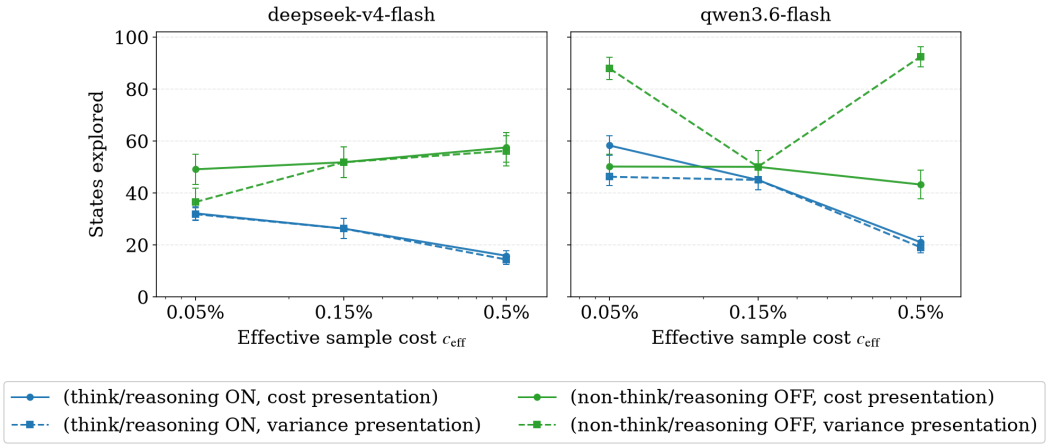


Figure 18: Each panel plots states explored against the effective sample cost c_{eff} for reasoning vs. non-reasoning variants of the listed models. Solid lines show exploration for models at varying costs. Dashed lines show identical results for $c_{\text{eff}} = 0.15\%$ but show exploration for models anchored from the $c_{\text{eff}} = 0.15\%$ cell but with their prior variances scaled such that the optimal policy is transformed to match the policy for either $c_{\text{eff}} = 0.05\%$ or $c_{\text{eff}} = 0.5\%$. Results are over 200 simulations at each configuration and error bars are 95% confidence intervals which are constructed similarly to Figure 2.

D Extensions

D.1 Scratchpad Reasoning Experiments

We additionally run experiments with LLMs that are prompted to produce a scratchpad of reasoning before making their decision. There are two types of scratchpad prompts, *structured*, which instructs the LLM to reason through its decisions with a provided template, and *freeform*, which prompts the LLM to reason without any fixed structure. Full prompts are shown in Appendix E.2. Figure 19 reports the sample cost elasticity of exploration and Figure 20 reports the mean states explored and mean net reward (normalized by the prior mean of 20) for selected models under their reasoning, non-reasoning, and scratchpad variants. The increase in performance and elasticity under the freeform scratchpad prompt shows that reasoning itself improves agents' ability to perform the task. This indicates that the heightened performance of native reasoning models is not fully inherited from capabilities learned during training that are independent from the ability to reason itself. The mistral-medium-3.1 results particularly demonstrate this since that is a model which was trained entirely without reasoning and still sees performance improvements when prompted to reason. The results under the structured prompt still look to generally improve performance but aren't as conclusively strong, suggesting that templated reasoning may actually impede the model's ability to reason effectively.

D.2 Alternative Prompt Scenarios

We run our experiments on the same experimental configuration as before except with a set of alternative prompts which feature more real-world shopping, investment, and hiring scenarios that better reflect the economic decisions an LLM would make in practice. We use Qwen3.6 Flash for its token cost efficiency and to assess whether its difference in elasticities between reasoning variants in the first experiment continues in these contexts. The results in Figure 21 show that the higher cost sensitivity of reasoning models persists across every scenario, which suggests that our main results generalize more widely and are not purely an artifact of the abstract prompt scenario. The elasticity under the abstract scenario is also much closer than the alternatives to that of the optimal policy, indicating that the LLM may struggle to apply its recursive reasoning capabilities to real-world environments or that the abstract nature of the prompt primes the necessary mathematical reasoning. The full per-scenario prompts are reported in Appendix E.1.

The full results for Qwen3.6 Flash across the four prompt scenarios (abstract, shopping, hiring, investment) are shown below. Figure 22 shows mean normalized net reward and Figure 23 shows the

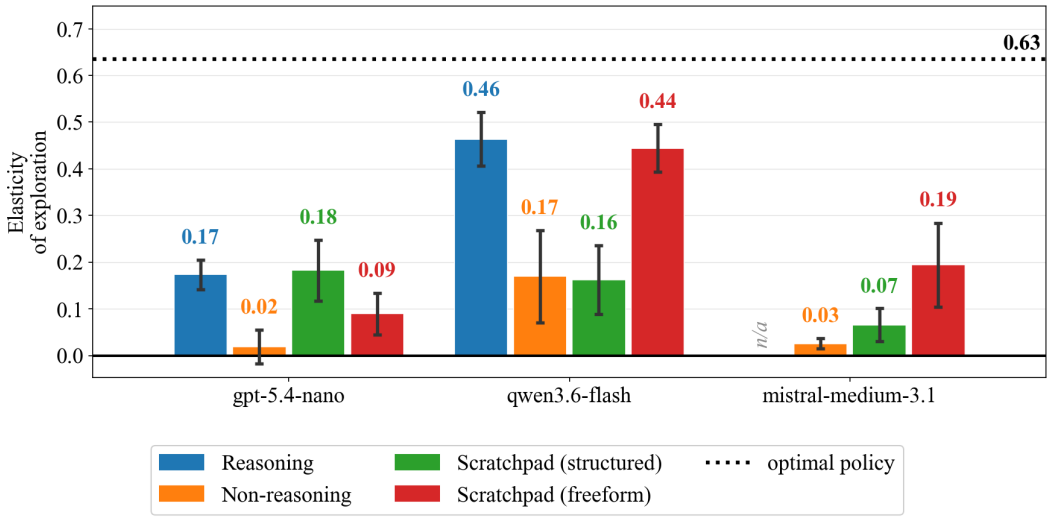


Figure 19: Sample cost elasticity of exploration for each model under its reasoning, non-reasoning, structured scratchpad, and freeform scratchpad prompts. The elasticity is the negative slope of the log-log relationship between the number of states explored and the cost per sample, estimated from 200 runs at each of three costs in one regression over every bar with variant and seed fixed effects, and the dotted line shows the elasticity of the optimal policy. Error bars are obtained in the same manner as in Figure 3. Variants marked n/a were not run.

mean number of states explored, in each case across sample costs $c \in \{0.05\%, 0.15\%, 0.5\%\}$ and split by reasoning and non-reasoning.

E Prompts

E.1 Prompt Scenarios

We construct four qualitatively different scenarios to test how the LLMs' decisions are affected by the context in which the game is presented.

- **Abstract:** The game is framed as a theoretical decision-making problem where the agent samples i.i.d. random variables from an unknown distribution and must commit to one realized value. They are told each additional random variable observed incurs a cost.
- **Investment:** The LLM is told they are an investment-selection agent helping a user pick one stock and that they can view the true return of stocks. The cost for exploration reflects the cost of keeping the user's money uninvested for an additional period.
- **Shopping:** The LLM plays the role of a shopper choosing one grocery brand to buy for their user. Each brand has a true user satisfaction rating which is observed when the user tastes the product from that brand. The cost for exploration reflects the cost of asking the user to sample another brand themselves.
- **Hiring:** The LLM acts as a hiring manager at a firm selecting a job applicant to fill a role. Interviewing candidates provides the LLM with a true performance score for the candidate. The cost for exploration reflects the time it takes to complete each additional interview which leaves the role open for an additional period.

For each scenario, we reproduce below the full system prompt and the two user-prompt forms used in our continuous conversation setting: the prompt shown at $t = 0$ which reports the free initial observation and the prompt shown after each paid exploration. The exploration cost, current turn, number of turns remaining, total accumulated cost, distribution parameters, and observed rewards are shown using brace-delimited placeholders (e.g., `{p99_value}`). Item labels, such as tickers, brand names, candidate names, or variable labels, are shown as generic placeholders.

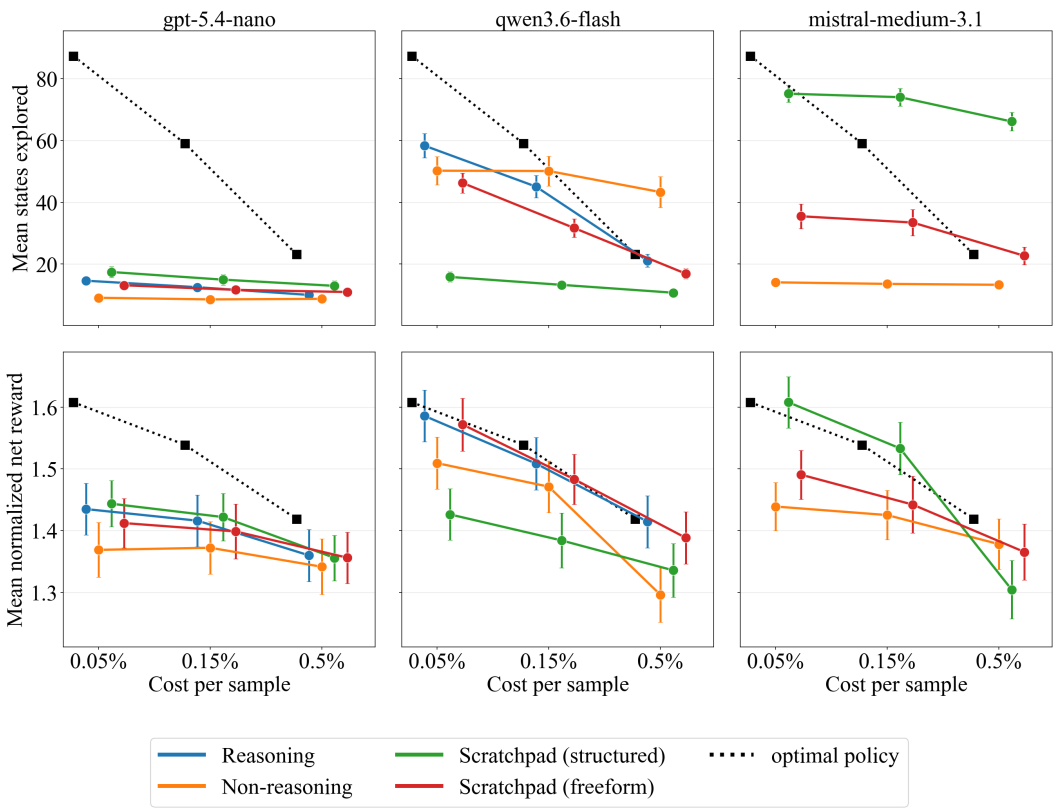


Figure 20: Mean states explored (top row) and mean net reward normalized by the prior mean of 20 (bottom row) across sample costs $c \in \{0.05\%, 0.15\%, 0.5\%\}$ for each model under its reasoning, non-reasoning, structured scratchpad, and freeform scratchpad prompts. The dotted line shows the optimal policy. Error bars are obtained in the same manner as in Figure 2.



Figure 21: Sample cost elasticity of exploration for Qwen3.6 Flash across the four prompt scenarios (abstract, shopping, hiring, investment), split into reasoning and non-reasoning variants. The elasticity is the negative slope of the log-log relationship between the number of states explored and the cost per sample, estimated from 200 runs at each of three costs in one regression over every bar with variant and seed fixed effects, and the dotted line shows the elasticity of the optimal policy defined in Section 3.2. Error bars are obtained in the same manner as in Figure 3.

E.1.1 ABSTRACT

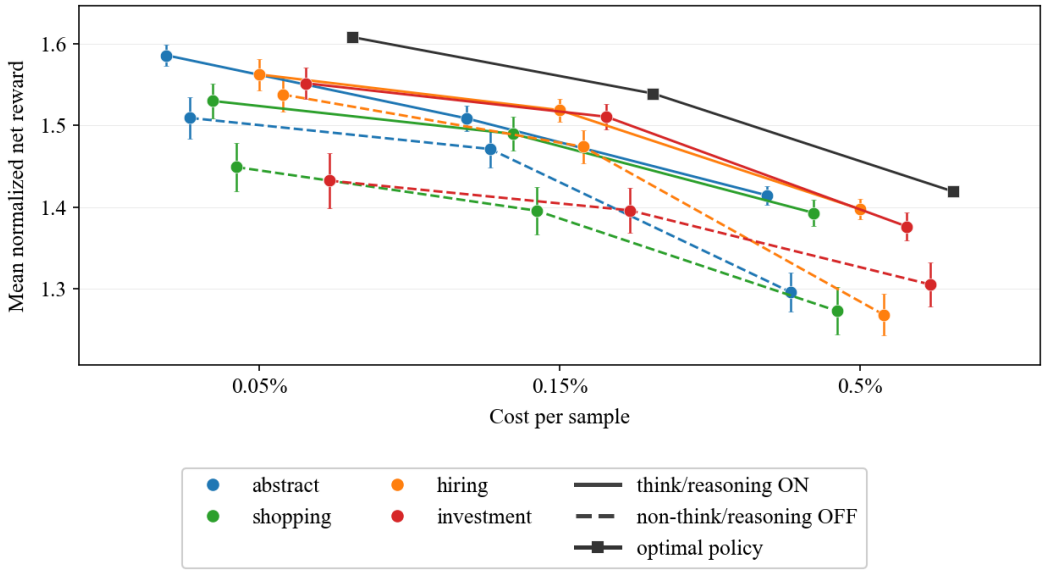


Figure 22: Mean net reward (normalized by the prior mean of 20) for qwen3.6-flash across 200 games with a maximum of 100 turns over the four prompt scenarios and sample costs $c \in \{0.05\%, 0.15\%, 0.5\%\}$. Solid lines are the reasoning variant and dashed lines the non-reasoning variant. The optimal policy defined in Section 3.2 is run on 100,000 games and shown in black. Error bars are obtained in the same manner as in Figure 2.

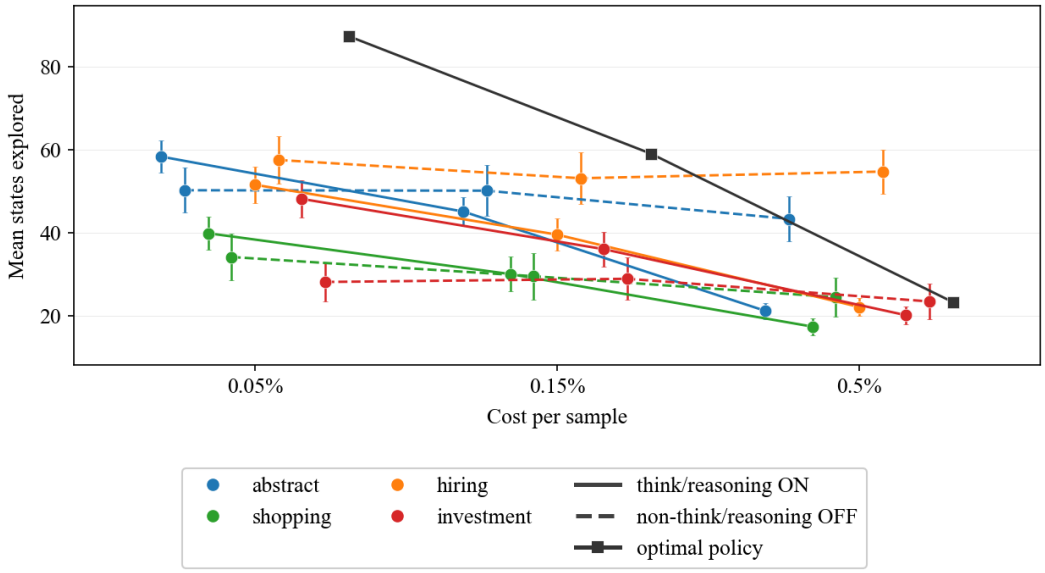


Figure 23: Mean number of states explored for qwen3.6-flash across 200 games with a maximum of 100 turns over the four prompt scenarios and sample costs $c \in \{0.05\%, 0.15\%, 0.5\%\}$. Solid lines are the reasoning variant and dashed lines the non-reasoning variant. The optimal policy defined in Section 3.2 is run on 100,000 games and shown in black. Error bars are obtained in the same manner as in Figure 2.

System prompt

You are an analyst sampling i.i.d. random variables from an unknown distribution and must commit to one realized value. Each random variable has a true value. You can see the true value of any random variable you have already observed. Each additional random variable you observe costs `{sample_cost}` units. The first random variable has already been observed for free. The value of each random variable comes from a process with four named parameters:

- mu0 = `{mu0}`: the prior mean of mu (the long-run mean of the observations).
- kappa = `{kappa}`: the prior precision-multiplier for mu; larger kappa pulls mu tighter to mu0.
- alpha = `{alpha}`: the shape parameter of the InverseGamma prior on sigma^2 (the variance of the observations).
- beta = `{beta}`: the scale parameter of the InverseGamma prior on sigma^2.

These four parameters drive the data-generating process in the following order:

1. sigma^2 is generated from InverseGamma(alpha=`{alpha}`, beta=`{beta}`). This sets the variance of the data; the variance quartiles under this prior are Q1=`{variance_q1}`, median=`{variance_median}`, Q3=`{variance_q3}`.
2. Conditional on sigma^2, mu is generated from Normal(mean=mu0=`{mu0}`, variance=sigma^2/kappa = sigma^2/`{kappa}`). This sets the mean of the data, and couples it to the realized sigma^2 through kappa.
3. Each observation is then generated independently from Normal(mean=mu, variance=sigma^2). Steps 1-2 happen once at the start of the game; step 3 repeats each time you observe a value.

Before any observations, an initial value of one random variable has 1st-percentile `{p1_value}` and 99th-percentile `{p99_value}`. Your confidence intervals on the distribution should shrink as you observe more samples, and you should update your prior over mu and sigma^2 with each new value you observe. At each decision, either output STOP to lock in the random variable with the highest observed value so far, or output EXPLORE to pay the cost and observe one additional random variable. If you output STOP, your final payoff is the highest observed value across all observed random variables minus all costs incurred so far. Treat utility as fully linear in payoff, with no risk aversion or any other curvature. Your goal is to maximize expected final payoff for your user. To inform each decision, weigh your current beliefs about the distribution of values, the highest observed value so far, the cost of each additional random variable, and how many paid observing opportunities remain. Respond with exactly one token: either EXPLORE or STOP.

User prompt at $t = 0$

The first random variable you observed was X1 (value `{observed_value}`).
Decision `{decision_number}`.
Paid observing opportunities remaining: `{remaining_opportunities}`.
Total cost incurred so far: `{total_cost}`.
Output EXPLORE to pay `{sample_cost}` and observe one new random variable, or output STOP to lock in the random variable with the highest observed value now. Required output: exactly one token, either EXPLORE or STOP.

User prompt at $t > 0$

You chose EXPLORE. New random variable observed: X2 (value `{observed_value}`).
Decision `{decision_number}`.
Paid observing opportunities remaining: `{remaining_opportunities}`.
Total cost incurred so far: `{total_cost}`.
Output EXPLORE to pay `{sample_cost}` and observe one new random variable, or output STOP to lock in the random variable with the highest observed value now. Required output: exactly one token, either EXPLORE or STOP.

E.1.2 INVESTMENT

System prompt

You are an investment-selection agent helping a user pick one stock. Each stock has a true return. You can see the true return of any stock you have already viewed. Each additional stock you view costs `{sample_cost}` dollars. The first stock has already been viewed for free; each additional view reflects the cost of holding the user's money uninvested for another period. The return of each stock comes from a process with four named parameters:

- mu0 = `{mu0}`: the prior mean of mu (the long-run mean of the observations).
- kappa = `{kappa}`: the prior precision-multiplier for mu; larger kappa pulls mu tighter to mu0.
- alpha = `{alpha}`: the shape parameter of the InverseGamma prior on sigma^2 (the variance of the observations).
- beta = `{beta}`: the scale parameter of the InverseGamma prior on sigma^2.

These four parameters drive the data-generating process in the following order:

1. sigma^2 is generated from InverseGamma(alpha=`{alpha}`, beta=`{beta}`). This sets the variance of the data; the variance quartiles under this prior are Q1=`{variance_q1}`, median=`{variance_median}`, Q3=`{variance_q3}`.
2. Conditional on sigma^2, mu is generated from Normal(mean=mu0=`{mu0}`, variance=sigma^2/kappa = sigma^2/`{kappa}`). This sets the mean of the data, and couples it to the realized sigma^2 through kappa.
3. Each observation is then generated independently from Normal(mean=mu, variance=sigma^2). Steps 1-2 happen once at the start of the game; step 3 repeats each time you observe a value.

Before any observations, an initial return of one stock has 1st-percentile `{p1_value}` and 99th-percentile `{p99_value}`. Your confidence intervals on the distribution should shrink as you observe more samples, and you should update your prior over mu and sigma^2 with each new value you observe. At each decision, either output STOP to lock in the stock with the highest observed return so far, or output EXPLORE to pay the cost and view one additional stock. If you output STOP, your final payoff is the highest observed return across all viewed stocks minus all costs incurred so far. Treat utility as fully linear in payoff, with no risk aversion or any other curvature. Your goal is to maximize expected final payoff for your user. To inform each decision, weigh your current beliefs about the distribution of returns, the highest observed return so far, the cost of each additional stock, and how many paid viewing opportunities remain. Respond with exactly one token: either EXPLORE or STOP.

User prompt at $t = 0$

The first stock you viewed was TICKER (return `{observed_return}`).
Decision `{decision_number}`.
Paid viewing opportunities remaining: `{remaining_opportunities}`.
Total cost incurred so far: `{total_cost}`.
Output EXPLORE to pay `{sample_cost}` and view one new stock, or output STOP to lock in the stock with the highest observed return now. Required output: exactly one token, either EXPLORE or STOP.

User prompt at $t > 0$

You chose EXPLORE. New stock viewed: TICKER (return `{observed_return}`).
Decision `{decision_number}`.
Paid viewing opportunities remaining: `{remaining_opportunities}`.
Total cost incurred so far: `{total_cost}`.
Output EXPLORE to pay `{sample_cost}` and view one new stock, or output STOP to lock in the stock with the highest observed return now. Required output: exactly one token, either EXPLORE or STOP.

E.1.3 SHOPPING

System prompt

You are a shopper choosing one grocery brand to buy for your user. Each brand has a true user satisfaction rating. You can see the true user satisfaction rating of any brand you have already tasted. Each additional brand you taste costs `{sample_cost}` dollars. The first brand has already been tasted for free; each additional tasting reflects the cost of asking the user to sample yet another brand before buying. The user satisfaction rating of each brand comes from a process with four named parameters:

- mu0 = `{mu0}`: the prior mean of mu (the long-run mean of the observations).
- kappa = `{kappa}`: the prior precision-multiplier for mu; larger kappa pulls mu tighter to mu0.
- alpha = `{alpha}`: the shape parameter of the InverseGamma prior on sigma^2 (the variance of the observations).
- beta = `{beta}`: the scale parameter of the InverseGamma prior on sigma^2.

These four parameters drive the data-generating process in the following order:

1. sigma^2 is generated from InverseGamma(alpha=`{alpha}`, beta=`{beta}`). This sets the variance of the data; the variance quartiles under this prior are Q1=`{variance_q1}`, median=`{variance_median}`, Q3=`{variance_q3}`.
2. Conditional on sigma^2, mu is generated from Normal(mean=mu0=`{mu0}`, variance=sigma^2/kappa = sigma^2/`{kappa}`). This sets the mean of the data, and couples it to the realized sigma^2 through kappa.
3. Each observation is then generated independently from Normal(mean=mu, variance=sigma^2). Steps 1-2 happen once at the start of the game; step 3 repeats each time you observe a value.

Before any observations, an initial user satisfaction rating of one brand has 1st-percentile `{p1_value}` and 99th-percentile `{p99_value}`. Your confidence intervals on the distribution should shrink as you observe more samples, and you should update your prior over mu and sigma^2 with each new value you observe. At each decision, either output STOP to lock in the brand with the highest observed user satisfaction rating so far, or output EXPLORE to pay the cost and taste one additional brand. If you output STOP, your final payoff is the highest observed user satisfaction rating across all tasted brands minus all costs incurred so far. Treat utility as fully linear in payoff, with no risk aversion or any other curvature. Your goal is to maximize expected final payoff for your user. To inform each decision, weigh your current beliefs about the distribution of user satisfaction ratings, the highest observed user satisfaction rating so far, the cost of each additional brand, and how many paid tasting opportunities remain. Respond with exactly one token: either EXPLORE or STOP.

User prompt at $t = 0$

The first brand you tasted was BRAND (user satisfaction rating `{observed_rating}`).
Decision `{decision_number}`.
Paid tasting opportunities remaining: `{remaining_opportunities}`.
Total cost incurred so far: `{total_cost}`.
Output EXPLORE to pay `{sample_cost}` and taste one new brand, or output STOP to lock in the brand with the highest observed user satisfaction rating now. Required output: exactly one token, either EXPLORE or STOP.

User prompt at $t > 0$

You chose EXPLORE. New brand tasted: BRAND (user satisfaction rating `{observed_rating}`).
Decision `{decision_number}`.
Paid tasting opportunities remaining: `{remaining_opportunities}`.
Total cost incurred so far: `{total_cost}`.
Output EXPLORE to pay `{sample_cost}` and taste one new brand, or output STOP to lock in the brand with the highest observed user satisfaction rating now. Required output: exactly one token, either EXPLORE or STOP.

E.1.4 HIRING

System prompt

You are a hiring manager at a firm selecting one job applicant to fill an open role. Each candidate has a true performance score. You can see the true performance score of any candidate you have already interviewed. Each additional candidate you interview costs `{sample_cost}` dollars. The first candidate has already been interviewed for free; each additional interview reflects the cost of leaving the role open another period. The performance score of each candidate comes from a process with four named parameters:

- mu0 = `{mu0}`: the prior mean of mu (the long-run mean of the observations).
- kappa = `{kappa}`: the prior precision-multiplier for mu; larger kappa pulls mu tighter to mu0.
- alpha = `{alpha}`: the shape parameter of the InverseGamma prior on sigma^2 (the variance of the observations).
- beta = `{beta}`: the scale parameter of the InverseGamma prior on sigma^2.

These four parameters drive the data-generating process in the following order:

1. sigma^2 is generated from InverseGamma(alpha=`{alpha}`, beta=`{beta}`). This sets the variance of the data; the variance quartiles under this prior are Q1=`{variance_q1}`, median=`{variance_median}`, Q3=`{variance_q3}`.
2. Conditional on sigma^2, mu is generated from Normal(mean=mu0=`{mu0}`, variance=sigma^2/kappa = sigma^2/`{kappa}`). This sets the mean of the data, and couples it to the realized sigma^2 through kappa.
3. Each observation is then generated independently from Normal(mean=mu, variance=sigma^2). Steps 1-2 happen once at the start of the game; step 3 repeats each time you observe a value.

Before any observations, an initial performance score of one candidate has 1st-percentile `{p1_value}` and 99th-percentile `{p99_value}`. Your confidence intervals on the distribution should shrink as you observe more samples, and you should update your prior over mu and sigma^2 with each new value you observe. At each decision, either output STOP to lock in the candidate with the highest observed performance score so far, or output EXPLORE to pay the cost and interview one additional candidate. If you output STOP, your final payoff is the highest observed performance score across all interviewed candidates minus all costs incurred so far. Treat utility as fully linear in payoff, with no risk aversion or any other curvature. Your goal is to maximize expected final payoff for your user. To inform each decision, weigh your current beliefs about the distribution of performance scores, the highest observed performance score so far, the cost of each additional candidate, and how many paid interviewing opportunities remain. Respond with exactly one token: either EXPLORE or STOP.

User prompt at $t = 0$

The first candidate you interviewed was CANDIDATE (performance score `{observed_score}`).
Decision `{decision_number}`.
Paid interviewing opportunities remaining: `{remaining_opportunities}`.
Total cost incurred so far: `{total_cost}`.
Output EXPLORE to pay `{sample_cost}` and interview one new candidate, or output STOP to lock in the candidate with the highest observed performance score now. Required output: exactly one token, either EXPLORE or STOP.

User prompt at $t > 0$

You chose EXPLORE. New candidate interviewed: CANDIDATE (performance score `{observed_score}`).
Decision `{decision_number}`.
Paid interviewing opportunities remaining: `{remaining_opportunities}`.
Total cost incurred so far: `{total_cost}`.
Output EXPLORE to pay `{sample_cost}` and interview one new candidate, or output STOP to lock in the candidate with the highest observed performance score now. Required output: exactly one token, either EXPLORE or STOP.

E.2 Scratchpad Prompts

Agents with scratchpad reasoning capabilities modify the abstract scenario system prompt from Appendix E.1.1 by appending one of the following blocks to the prompt's end, depending on the selected scratchpad variant.

E.2.1 Structured Scratchpad

System prompt

Do not answer immediately: first reason step by step in a written scratchpad, then give your final answer. Structure every response exactly as follows, with each labeled line on its own line, in this order:

SCRATCHPAD:

STATE: the highest observed value so far, the number of random variables observed so far, and the number of paid observing opportunities remaining.

BELIEFS: your updated estimates of the mean and standard deviation of the value distribution, given every value observed so far.

NEXT DRAW: the probability that one more random variable beats the current best value, and the expected size of the improvement if it does.

VALUE OF EXPLORING: the expected gain in final payoff from observing one more random variable, compared with its cost of `{sample_cost}` units, taking the remaining opportunities into account.

DECISION: one sentence naming the action that maximizes expected final payoff.

FINAL ANSWER: <EXPLORE or STOP>

Scratchpad rules:

- Keep each labeled line to at most two short sentences.
- Fill in every line in the order shown; never skip, merge, or reorder lines.
- The last line must be exactly "FINAL ANSWER: EXPLORE" or "FINAL ANSWER: STOP", with nothing after it.

Note on the conversation transcript: to save space, your earlier replies are shown with the scratchpad omitted – only their FINAL ANSWER line was kept. Do not imitate that abbreviated form. Every new reply must contain the complete SCRATCHPAD template, ending with its FINAL ANSWER line.

E.2.2 Freeform Scratchpad

System prompt

Do not answer immediately: first reason in a written scratchpad, then give your final answer. Begin your response with the line "SCRATCHPAD:" and write your reasoning under it. The scratchpad is space for you to think before deciding; what you reason about, how much you write, and how you structure it are entirely up to you. End your response with one final line that is exactly "FINAL ANSWER: EXPLORE" or "FINAL ANSWER: STOP", with nothing after it. Every response must contain your written reasoning before the final line; never reply with only the final line. Note on the conversation transcript: to save space, your earlier replies are shown with the scratchpad omitted – only their FINAL ANSWER line was kept. Do not imitate that abbreviated form. Every new reply must still contain a SCRATCHPAD with your written reasoning, ending with its FINAL ANSWER line.

E.3 Continuation Value Prompts

The continuation value experiments use the following two system prompts in place of the standard ones shown in Appendix E.1.1. We report only the initial user prompt as the experiment only deals with the agent's first decision.

E.3.1 DECLINING-COST EXPERIMENT

System prompt

You are an analyst sampling i.i.d. random variables from a fully known distribution and must commit to one realized value. Each random variable has a true value. You can see the true value of any random variable you have already observed. Observing additional random variables is NOT free, and the cost falls each turn. The k-th paid observation costs

cost(k) = `{initial_cost}` * exp(-`{decline_rate}` * (k - 1)) units.

For example:

- observation 1: cost `{cost_1}` units
- observation 2: cost `{cost_2}` units
- observation 3: cost `{cost_3}` units
- observation 4: cost `{cost_4}` units
- observation 5: cost `{cost_5}` units
- observation 10: cost `{cost_10}` units
- observation 20: cost `{cost_20}` units
- observation 50: cost `{cost_50}` units
- observation 100: cost `{cost_100}` units
- observation 200: cost `{cost_200}` units

The full schedule is known in advance and never changes. The first random variable has already been observed for free. Every value is drawn independently from a normal distribution with mean `{mean}` and standard deviation `{standard_deviation}`. These are the TRUE parameters, known with certainty: there is nothing to learn about the distribution from observations. At each decision, either output STOP to lock in the random variable with the highest observed value so far, or output EXPLORE to pay the current cost and observe one additional random variable. If you output STOP, your final payoff is the highest observed value across all observed random variables minus all costs incurred so far. Treat utility as fully linear in payoff, with no risk aversion or any other curvature. Your goal is to maximize expected final payoff for your user. To inform each decision, weigh the known distribution of values, the highest observed value so far, the declining cost schedule, and how many paid observing opportunities remain. Respond with exactly one token: either EXPLORE or STOP.

User prompt at $t = 0$

The first random variable you observed was X1 (value `{observed_value}`).
Decision `{decision_number}`.
Paid observing opportunities remaining: `{remaining_opportunities}`.
Total cost incurred so far: `{total_cost}`.
Output EXPLORE to pay `{initial_cost}` and observe one new random variable, or output STOP to lock in the random variable with the highest observed value now. Required output: exactly one token, either EXPLORE or STOP.

E.3.2 CONSTANT-COST CONTROL

System prompt

You are an analyst sampling i.i.d. random variables from a fully known distribution and must commit to one realized value. Each random variable has a true value. You can see the true value of any random variable you have already observed. Observing each additional random variable costs a FIXED `{sample_cost}` units – the cost is the same on every turn and never changes. The first random variable has already been observed for free. Every value is drawn independently from a normal distribution with mean `{mean}` and standard deviation `{standard_deviation}`. These are the TRUE parameters, known with certainty: there is nothing to learn about the distribution from observations. At each decision, either output STOP to lock in the random variable with the highest observed value so far, or output EXPLORE to pay the cost and observe one additional random variable. If you output STOP, your final payoff is the highest observed value across all observed random variables minus all costs incurred so far. Treat utility as fully linear in payoff, with no risk aversion or any other curvature. Your goal is to

maximize expected final payoff for your user. To inform each decision, weigh the known distribution of values, the highest observed value so far, the fixed cost per observation, and how many paid observing opportunities remain. Respond with exactly one token: either EXPLORE or STOP.

User prompt at $t = 0$

The first random variable you observed was X1 (value `{observed_value}`).
Decision `{decision_number}`.
Paid observing opportunities remaining: `{remaining_opportunities}`.
Total cost incurred so far: `{total_cost}`.
Output EXPLORE to pay `{sample_cost}` and observe one new random variable, or output STOP to lock in the random variable with the highest observed value now. Required output: exactly one token, either EXPLORE or STOP.

E.4 Token Cost Prompts

Token-cost experiments use the abstract scenario prompt shown below. This contains slight modifications to the abstract scenario prompt from Appendix E.1.1 in order to communicate to the agents that their tokens will be charged a search cost rather than the observations themselves.

System prompt

You are an analyst sampling i.i.d. random variables from an unknown distribution and must commit to one realized value. Each random variable has a true value. You can see the true value of any random variable you have already observed. You pay for every completion token you produce in this task, including any hidden reasoning tokens: each token costs `{token_cost}` units. Your EXPLORE or STOP answer costs one token regardless of which choice you take. The first random variable has already been observed for free. The value of each random variable comes from a process with four named parameters:

- mu0 = `{mu0}`: the prior mean of mu (the long-run mean of the observations).
- kappa = `{kappa}`: the prior precision-multiplier for mu; larger kappa pulls mu tighter to mu0.
- alpha = `{alpha}`: the shape parameter of the InverseGamma prior on sigma^2 (the variance of the observations).
- beta = `{beta}`: the scale parameter of the InverseGamma prior on sigma^2.

These four parameters drive the data-generating process in the following order:

1. sigma^2 is generated from InverseGamma(alpha=`{alpha}`, beta=`{beta}`). This sets the variance of the data; the variance quartiles under this prior are Q1=`{variance_q1}`, median=`{variance_median}`, Q3=`{variance_q3}`.
2. Conditional on sigma^2, mu is generated from Normal(mean=mu0=`{mu0}`, variance=sigma^2/kappa = sigma^2/`{kappa}`). This sets the mean of the data, and couples it to the realized sigma^2 through kappa.
3. Each observation is then generated independently from Normal(mean=mu, variance=sigma^2). Steps 1-2 happen once at the start of the game; step 3 repeats each time you observe a value.

Before any observations, an initial value of one random variable has 1st-percentile `{p1_value}` and 99th-percentile `{p99_value}`. Your confidence intervals on the distribution should shrink as you observe more samples, and you should update your prior over mu and sigma^2 with each new value you observe. At each decision, either output STOP to lock in the random variable with the highest observed value so far, or output EXPLORE to observe one additional random variable. If you output STOP, your final payoff is the highest observed value across all observed random variables minus all costs incurred so far. Treat utility as fully linear in payoff, with no risk aversion or any other curvature. Your goal is to maximize expected final payoff for your user. To inform each decision, weigh your current beliefs about the distribution of values, the highest observed value so far, the token cost of the additional responses that further observing entails, and how many paid observing opportunities remain. Respond with exactly one token: either EXPLORE or STOP.

User prompt at $t = 0$

The first random variable you observed was X1 (value `{observed_value}`).
Decision `{decision_number}`.
Paid observing opportunities remaining: `{remaining_opportunities}`.
Completion tokens charged so far: `{tokens_charged}`.
Total token charges incurred so far: `{total_token_charges}`.
Output EXPLORE to observe one new random variable, or output STOP to lock in the random variable with the highest observed value now. Either way, every completion token of this response costs `{token_cost}` units. Required output: exactly one token, either EXPLORE or STOP.

User prompt at $t > 0$

You chose EXPLORE. New random variable observed: X2 (value `{observed_value}`).
Decision `{decision_number}`.
Paid observing opportunities remaining: `{remaining_opportunities}`.
Completion tokens charged so far: `{tokens_charged}`.
Total token charges incurred so far: `{total_token_charges}`.
Output EXPLORE to observe one new random variable, or output STOP to lock in the random variable with the highest observed value now. Either way, every completion token of this response costs `{token_cost}` units. Required output: exactly one token, either EXPLORE or STOP.

E.5 WEBSHOP PROMPTS

WebShop agents are shown prompts that follow the template below. Customer instructions and appropriate cost blocks are integrated into the system prompt and correspondingly applied to turn message formatting.

System prompt

You are a personal shopping assistant. A customer has asked you to purchase one item on their behalf. Their request, in their own words:

`{customer_instruction}`

The store has `{number_of_products}` products available for a request of this kind. You will review them one at a time, in the order the store returns them. For each product you are shown its title and its price.

Objective

Purchase the product that best satisfies the customer's request. Every element of it counts: the type of item, each feature they specified, the options they named, and their price limit. The store will rarely stock an exact match, so satisfy as much of the request as the available products allow.

You do not need to choose sizes, colours or other options yourself. If you purchase a product, the customer receives it with the options they asked for wherever that product offers them.

Search costs

`{cost_block}`

Each turn

Respond with one line and nothing else:

- NEXT – review the next product.
- BUY <number> – purchase a product you have reviewed, identified by the number it was shown with, for example BUY 7.

You may purchase any product you have already reviewed, not only the one in front of you; returning to an earlier product costs nothing. There are `{number_of_products}` products in total, and you must

make a purchase once you reach the last one. Keep your deliberation to yourself: the response is a single line, either NEXT or BUY followed by a number.

E.5.1 Explicit Per-View Cost Block

System prompt

Reviewing each additional product costs ${search_cost} in time and fees, and the customer bears that cost in addition to the price of the item. The total price they pay is therefore:

total price = price of the item purchased

+ ${search_cost} per product reviewed after the first

The first product is reviewed at no cost. The customer's price limit applies to that total price, not to the item's price alone: the longer you continue reviewing products, the lower the price of the item must be to remain within the limit. Weigh this against the possibility that a later product is a closer match.

E.5.2 Per-Token Cost Block

System prompt

Reviewing a product carries no fee of its own. What costs money is the thinking you do before you answer: every hidden reasoning token you produce costs ${token_price} in time and fees, and the customer bears that cost in addition to the price of the item. The total price they pay is therefore:

total price = price of the item purchased

+ ${token_price} per hidden reasoning token you generate

Your NEXT or BUY answer is billed as one further token at that same price, whichever of the two you choose, so what reviewing a further product costs the customer is whatever you spend thinking before you answer it. The customer's price limit applies to that total price, not to the item's price alone: the longer you continue reviewing products, the lower the price of the item must be to remain within the limit. Weigh this against the possibility that a later product is a closer match.

E.5.3 Turn Messages

Explicit per-view cost

Product {product_number} of {number_of_products}
{product_title}
Price: ${product_price}
Products reviewed: {product_number} of {number_of_products}. Search costs incurred: ${search_costs_incurred}.

Per-token cost

Product {product_number} of {number_of_products}
{product_title}
Price: ${product_price}
Products reviewed: {product_number} of {number_of_products}. Reasoning tokens charged so far: {tokens_charged}.
Token charges incurred so far: ${token_charges_incurred}.

Final-turn suffix

This is the final product; there are none left to review. Respond with BUY and the number of whichever of the `{number_of_products}` products you have reviewed you want to buy.

Correction message

That response could not be parsed. Respond with one line: either NEXT to review the next product, or BUY <number> to purchase one of the products you have already reviewed (1–`{products_reviewed}`). `{optional_error_explanation}`

E.5.4 Search Query Generation Prompt

We provide the below prompt to Opus 5 subagents in order to create search queries for WebShop user instructions. Each agent has their own batch of instructions (referred to as "goals" in the Output section of the prompt) to create queries for. `{GOAL_IDS}` and `{OUT_PATH}` are substituted in for each agent.

You write WebShop search queries. For each goal index in this batch you must produce ONE search query that (a) a real shopper could plausibly have typed after reading that goal's instruction, and (b) retrieves the goal's reference product inside the top-50 Lucene/BM25 results.

Your batch: goal indices `{GOAL_IDS}`
Write your results to: `{OUT_PATH}`

Rules for the query

1. **Derive it from the instruction, not from the reference product.** Read the instruction first, then write the query the way a shopper would: lowercase, 2–6 words, product type plus the one or two attributes that matter.
2. **Every word must come from the instruction's own vocabulary** (plural or singular forms of an instruction word are fine, e.g. "pillows" for "pillow"). Never copy brand names, model numbers, sizes, colors or phrasing that appear only in the reference product's title. Never put the asin in the query.
3. **No price terms** ("under 30 dollars", "cheap") and no filler ("buy", "best", "for me", "find").
4. The query should be a *category-level* query that returns a broad slate of comparable products, not a near-unique description of one item. If a query returns fewer than ~30 results, it is too narrow – broaden it.
5. Prefer the shortest query that works.

Loop, per goal

1. Fetch the goal. Read the instruction.
2. Write a candidate query under the rules above and check it.
3. If `found` is true, you are done – but if the query was long, try trimming a word and keep the shorter query when it still finds the reference product.
4. If `found` is false, diagnose: look at what the top-10 results are about and which instruction words are pulling them there. Re-weight toward other instruction words – swap the head noun for the instruction's own synonym, drop an over-specific modifier, or add a second instruction word that discriminates. You may look at the reference title to understand *why* the query missed, but the replacement query still has to obey rule 2.
5. Stop after at most 8 checks per goal. If nothing works, record the best attempt (highest-recall, most natural) with `"found": false`.

Output

Write `{OUT_PATH}` as a JSON array, one object per goal in this batch:

```
[
  {
    "goal_idx": 71,
    "instruction": "<verbatim instruction text>",
    "query": "<final query>",
    "found": true,
    "rank": 7,
    "n_results": 50,
    "reference_asin": "B0743JKHBV",
    "attempts": ["first query", "second query"],
    "note": "one line: why this query, or why nothing worked"
  }
]
```

`attempts` lists every query you checked, in order (the final one included). Do not edit any other file. Report back a one-line summary per goal: `<goal_idx>: "<query>" -> rank <r>/<n>`.

E.6 Environmental Understanding Prompts

Here we provide the questions which models are tested on in Table 6. For each question, the system prompt is shown alongside the relevant user prompt. Note that the system prompt is very similar to the portion of the system prompt describing the data-generating process used in the abstract scenario of the game shown in Appendix E.1.1. All six ask the model for the posterior update of $(\mu_0, \kappa, \alpha, \beta)$ under the rules of Appendix A.2.

System prompt

You will be given a description of a probabilistic process that produces numerical values. Read it carefully and answer the question that follows. Reason briefly and end with a clear, concrete answer.

Process description: The prior on this data-generating process has four named parameters:

- mu0 = 20: the prior mean of mu (the long-run mean of the observations).
- kappa = 1: the prior precision-multiplier for mu; larger kappa pulls mu tighter to mu0.
- alpha = 10: the shape parameter of the InverseGamma prior on sigma^2 (the variance of the observations).
- beta = 250: the scale parameter of the InverseGamma prior on sigma^2.

These four parameters drive the data-generating process in the following order:

1. sigma^2 is generated from InverseGamma(alpha=10, beta=250). This sets the variance of the data; the variance quartiles under this prior are Q1=20.984, median=25.857, Q3=32.359.
2. Conditional on sigma^2, mu is generated from Normal(mean=mu0=20, variance=sigma^2/kappa = sigma^2/1). This sets the mean of the data, and couples it to the realized sigma^2 through kappa.
3. Each observation X is then generated independently from Normal(mean=mu, variance=sigma^2). Steps 1-2 happen once at the start of the game; step 3 repeats each time you observe a value.

Before any observations, an initial draw has 1st-percentile 2.125 and 99th-percentile 37.875. Your confidence intervals on the distribution should shrink as you observe more samples, and you should update your prior over mu and sigma^2 with each new value you observe.

User prompts

Each user prompt opens with the observation line(s) specific to that question (listed below), followed by a blank line and then the shared parameter-update request:

Shared user-prompt suffix

The four prior parameters of the process are:

- mu0: the prior mean of mu
- kappa: the prior precision-multiplier for mu (mu | sigma^2 ∼ Normal(mu0, sigma^2 / kappa))
- alpha: the InverseGamma shape parameter for sigma^2
- beta: the InverseGamma scale parameter for sigma^2

Given the observation(s) above, what exactly would you update each of these four parameters to? For each parameter, give an explicit new numeric value (use the form mu0=..., kappa=..., alpha=..., beta=...) and a one-sentence explanation of how you got it.

Q1 — single observation, common value

Suppose you observed a single i.i.d. value from the process above, and that value is a very common value: X1 = 20.

Q2 — single observation, borderline high value

Suppose you observed a single i.i.d. value from the process above, and that value is a borderline high value: X1 = 32.

Q3 — single observation, extremely high value

Suppose you observed a single i.i.d. value from the process above, and that value is an extremely high value: X1 = 70.

Q4 — five observations, common values

Suppose you observed these five i.i.d. values from the process above, and they are five very common values: [18.5, 21.0, 19.5, 22.5, 20.0].

Q5 — five observations, borderline high values

Suppose you observed these five i.i.d. values from the process above, and they are five borderline high values: [30.0, 33.0, 28.0, 35.0, 31.0].

Q6 — five observations, extremely high values

Suppose you observed these five i.i.d. values from the process above, and they are five extremely high values: [65.0, 70.0, 75.0, 60.0, 80.0].

F Estimation of Elasticities and Confidence Intervals

We construct elasticities for exploration across the figures via an OLS regression over all agents a, game seeds j, and costs c_i as

$$\log(\tau_{a,j,i}) \;=\; \alpha_a + \gamma_j + \beta_a \log(c_i) + u_{a,j,i},$$

with $\tau_{a,j,i}$ the agent's number of states explored for each seed at each cost, α_a an agent fixed effect, and γ_j a random seed fixed effect. In the case of WebShop, instructions replace the random seeds. The elasticity is then computed as $\hat{\varepsilon}_a = -\hat{\beta}_a$. Standard errors for elasticity estimates are clustered

Table 9: Sample cost elasticity of exploration for different regression specifications. The outcome variable is $\log(\text{states explored})$ and coefficients are the slopes $\hat{\beta}_a$ with respect to $\log(c)$ for each model $\times$ pricing combination; the elasticity plotted in our figures is the negative of the coefficient, $\hat{\varepsilon}_a = -\hat{\beta}_a$. All combinations are estimated in a single regression with the listed fixed effects. Standard errors are clustered by seed and shown in parentheses. $^{*}p < 0.10$, $^{**}p < 0.05$, $^{***}p < 0.01$.

Model	Pricing	(1)	(2)	(3)
gpt-5.4 (none / low)	Turn, non-reasoning	-0.1795^{***}	-0.1795^{***}	-0.1795^{***}
		(0.0390)	(0.0396)	(0.0551)
	Turn	-0.6404^{***}	-0.6404^{***}	-0.6404^{***}
		(0.0313)	(0.0318)	(0.0443)
	Token	-0.5609^{***}	-0.5609^{***}	-0.5609^{***}
		(0.0572)	(0.0581)	(0.0808)
gpt-5.4-nano (none / low)	Turn, non-reasoning	0.0066	0.0066	0.0066
		(0.0352)	(0.0358)	(0.0498)
	Turn	-0.2022^{***}	-0.2022^{***}	-0.2022^{***}
		(0.0286)	(0.0291)	(0.0405)
	Token	0.0156	0.0156	0.0156
		(0.0314)	(0.0319)	(0.0444)
qwen3.6-flash (off / on)	Turn, non-reasoning	-0.2988^{***}	-0.2988^{***}	-0.2988^{**}
		(0.0898)	(0.0914)	(0.1271)
	Turn	-0.6496^{***}	-0.6496^{***}	-0.6496^{***}
		(0.0496)	(0.0504)	(0.0701)
	Token	-0.2722^{***}	-0.2722^{***}	-0.2722^{***}
		(0.0652)	(0.0664)	(0.0923)
inkling-small (off / high)	Turn, non-reasoning	-0.0214^{*}	-0.0214	-0.0214
		(0.0129)	(0.0131)	(0.0182)
	Turn	-0.5654^{***}	-0.5654^{***}	-0.5654^{***}
		(0.0329)	(0.0335)	(0.0465)
	Token	-0.4791^{***}	-0.4791^{***}	-0.4791^{***}
		(0.0327)	(0.0332)	(0.0462)
nemotron-3-ultra (off / medium)	Turn, non-reasoning	0.0060	0.0060	0.0060
		(0.0367)	(0.0374)	(0.0520)
	Turn	-0.2608^{***}	-0.2608^{***}	-0.2608^{***}
		(0.0578)	(0.0588)	(0.0818)
	Token	-0.3647^{***}	-0.3647^{***}	-0.3647^{***}
		(0.0436)	(0.0444)	(0.0617)
Agent FE		Yes	Yes	No
Seed FE		No	Yes	No
Agent $\times$ seed FE		No	No	Yes
Fixed-effect columns		15	214	3,000
Observations		6,000	6,000	6,000
Clusters (seed)		200	200	200
R^2		0.457	0.559	0.816

by seed and shared across agents. The error bars that we plot for elasticities are 95% confidence intervals computed as $\hat{\varepsilon}_a \pm 1.96\,\widehat{\text{se}}(\hat{\beta}_a)$.

We report the coefficients behind the elasticities shown in our plots ($\hat{\varepsilon}_a = -\hat{\beta}_a$) in column (2) of Table 9 for the abstract sequential search game and column (2) of Table 10 for the WebShop game. We additionally show estimates for other regressions with different sets of fixed effects to show that our estimates and standard errors are robust to other defensible regression specifications.

Table 10: Sample cost elasticity of exploration for different regression specifications. The outcome variable is $\log(\text{states explored})$ and coefficients are the slopes $\hat{\beta}_a$ with respect to $\log(c)$ for each model $\times$ pricing combination; the elasticity plotted in our figures is the negative of the coefficient, $\hat{\varepsilon}_a = -\hat{\beta}_a$. All combinations are estimated in a single regression with the listed fixed effects. Standard errors are clustered by user instruction and shown in parentheses. $^{*}p < 0.10$, $^{**}p < 0.05$, $^{***}p < 0.01$.

Model	Pricing	(1)	(2)	(3)
gpt-5.4 (none / low)	Turn, non-reasoning	-0.0645^{***}	-0.0645^{***}	-0.0645^{***}
		(0.0150)	(0.0152)	(0.0184)
	Turn	-0.0931^{***}	-0.0931^{***}	-0.0931^{***}
		(0.0127)	(0.0128)	(0.0155)
	Token	-0.1446^{***}	-0.1446^{***}	-0.1446^{***}
		(0.0195)	(0.0197)	(0.0238)
gpt-5.4-nano (none / low)	Turn, non-reasoning	-0.0019	-0.0019	-0.0019
		(0.0265)	(0.0268)	(0.0325)
	Turn	-0.0593^{***}	-0.0593^{***}	-0.0593^{**}
		(0.0206)	(0.0208)	(0.0252)
	Token	-0.1190^{***}	-0.1190^{***}	-0.1190^{***}
		(0.0226)	(0.0228)	(0.0276)
qwen3.6-flash (off / on)	Turn, non-reasoning	0.0392^{**}	0.0392^{**}	0.0392^{*}
		(0.0190)	(0.0192)	(0.0232)
	Turn	-0.0839^{***}	-0.0839^{***}	-0.0839^{***}
		(0.0143)	(0.0145)	(0.0175)
	Token	-0.1242^{***}	-0.1242^{***}	-0.1242^{***}
		(0.0201)	(0.0203)	(0.0246)
inkling-small (off / high)	Turn, non-reasoning	0.0209	0.0209	0.0209
		(0.0440)	(0.0445)	(0.0539)
	Turn	-0.0203	-0.0203	-0.0203
		(0.0258)	(0.0261)	(0.0315)
	Token	-0.0686^{**}	-0.0686^{**}	-0.0686^{*}
		(0.0337)	(0.0340)	(0.0412)
nemotron-3-ultra (off / medium)	Turn, non-reasoning	0.0000	0.0000	0.0000
		(0.0000)	(0.0000)	(0.0000)
	Turn	-0.0616^{***}	-0.0616^{***}	-0.0616^{***}
		(0.0170)	(0.0172)	(0.0208)
	Token	-0.1121^{***}	-0.1121^{***}	-0.1121^{***}
		(0.0190)	(0.0192)	(0.0233)
Agent FE		Yes	Yes	No
Instruction FE		No	Yes	No
Agent $\times$ instruction FE		No	No	Yes
Fixed-effect columns		15	214	3,000
Observations		9,000	9,000	9,000
Clusters (instruction)		200	200	200
R^2		0.120	0.490	0.787