

Learning Efficient Cognitive Effort Allocation Under Bounded Rationality

Matthew Torre mtorre04@stanford.edu

Kim Ngo kimdngo@stanford.edu

Stanford University

December 5, 2025

Abstract

Humans and autonomous agents alike rarely act in a conventionally or mathematically optimal manner because of attention, time, and computational constraints. This project is aimed at studying how an agent should allocate cognitive effort between low-effort habitual choices against high-effort deliberation when fatigue accumulates over some time axis. We build a stochastic decision-making environment with latent fatigue that is tied to varying task difficulty. There are two action modes: habitual and deliberate that incur different effort-rewards tradeoffs. Each action mode influences fatigue in an intuitive and dynamic sense. A simple epsilon-greedy agent learns an efficiency-based policy that balances expected reward against effort costs. Simulations demonstrate this intuitive fatigue dynamic where persistent deliberation boosts short-term performance but increases fatigue while shifting toward habitual actions maintains more stable efficiency over longer horizons. Our simulation framework logs the trajectories of reward, fatigue, and efficiency over 140 time steps. Within this framework, we systematically compare alternative policies. This paper provides an initial direction for studying bounded rationality and computational models of decision fatigue.

1 Introduction

In the natural world, decision makers face an age-old constraint that is simple but elusive: they cannot think hard about every choice. Time, attention, and computation are bounded so agents' decision making degrades choice in order to satisfice rather than optimize. Simon's work with bounded rationality found empirically that decision fatigue is tangible and support this fact. Over long horizons, the tension compounds and deliberation can lift performance early on but deplete the very resources needed later. Purely habitual behavior runs its own risks including adherence to more sensitive/brittle shortcuts when conditions as shifted. The central problem in this project is *how to allocate cognitive effort*

across a sequence of partially observed tasks so that long-run reward efficiency remains high despite fatigue.

Our setting is modeled via a partially observable Markov Decision Process (POMDP) featuring latent fatigue, stochastic binary task difficulty, and two action modes implemented in code: a low effort habitual/heuristic response and a high effort deliberate computation. The environment dynamics supported in (`src/env.py`) explicitly couple all the actions, success probabilities, and fatigue accumulation. The agent defined in (`src/agent.py`), maintains reward-per-effort estimates and will act with epsilon-greedy exploration.

Running `python3 -m src.main -config configs/default.yaml` produces reproducible logs/plots of reward, fatigue, and efficiency trajectories, which currently show a mean reward-per-effort ratio of 1.48 over 25 episodes and fatigue climbing toward saturation when the agent over-deliberates.

Our contributions extend to the following: (i) producing a compact POMDP environment that makes the trade-off between accuracy and cognitive wear explicit and measurable; (ii) a baseline efficiency-tracking agent whose learned policy reveals the qualitative signature of fatigue-driven shifts toward low-effort actions; and (iii) an evaluation methodology that logs trajectories and renders figures. This allows us to perform side-by-side comparisons as we add belief-aware planners and bandit-style schedulers. The components put together give us an empirical testbed to study bounded rationality and decision fatigue with both quantitative and reproducible evidence.

2 Related Work

Bounded rationality [Simon(1955)] and studies of limited deliberation [Russell and Wefald(1991)] motivate algorithms that explicitly price computation. Other decision fatigue research shows that there is declining choice quality under sustained effort [Danziger et al.(2011), Blain et al.(2016)]. Habit formation supports the notion that low-effort policies can dominate over time [Wood and R  nger(2016)]. Our formalization of the problem follows POMDP framing of sequential decisions under uncertainty [Kochenderfer et al.(2022)]. Our formalization, however, departs from treating fatigue as this first-class latent state that couples to transition and reward dynamics. Bandit formulations provide alternatives when belief tracking is costly. Rational inattention [Sims(2003)] captures selective processing in a similar fashion. In this paper, we extend this line by offering a reproducible simulation. One that exposes effort-aware policies to both partial observability and fatigue-driven non-stationarity to allow for comparison across POMDP solvers and bandit-inspired schedulers within one codebase.

3 Problem Formulation

In our problem formulation we model an episodic sequence of T decisions. The latent state $s_t = (d_t, f_t)$ pairs binary task difficulty $d_t \in \{0, 1\}$ (easy/hard) with

continuous fatigue $f_t \in [0, 1]$. At each step the agent selects an action from the action set (e.g. $a_t \in \{\text{habitual}, \text{deliberate}\}$). Our transitions follow:

- Difficulty flips with probability β (`difficulty_flip_chance`), creating a non-stationary task meaning task difficulty doesn't stay constant. Sometimes you get harder tasks, sometimes easier ones akin to real-life task accumulation.
- Fatigue experiences dynamic evolution represented through $f_{t+1} = \text{clip}(f_t + \Delta_f(a_t, d_t, \text{outcome}))$ where Δ_f adds action-dependent increments. This can be characterized through the agents experience penalizing hard tasks and granting relief upon success (psychological momentum through a positive feedback loop).

Observations $o_t = (d_t + \epsilon_d, f_t + \epsilon_f)$ are noisy with Gaussian noise $\epsilon \sim \mathcal{N}(0, \sigma^2)$ clipped to $[0, 1]$ to stay within our valid, predefined limits which reflects self-monitoring (limited). Rewards combine both outcome value and effort (very valuable asset). Where $r_t = \text{outcome}(d_t, a_t, \text{success}) - \text{effort}(a_t) + \eta$ with small reward noise η .

3.1 Appropriateness of Approach:

This problem naturally manifests itself as a POMDP. This is the case because fatigue is "latent" or hidden and difficulty is only partially observed. Bandits are what serve as the approximate solvers when we collapse the belief to minimal context (e.g. current observations) and optimize the efficiency of actions without full planning. The agent also exhibits imperfect self-awareness where it cannot perfectly tell how tired it is or how hard the current task is relative to other or in isolation. It guesses from noisy cues which represents human-like uncertainty.

4 Methods

Our computational study is implemented end-to-end in the accompanying codebase and is centered around a small POMDP that incorporate fatigue factors, partial observability, and effort-sensitive rewards.

4.1 POMDP Model

State: Our latent state $s_t = (d_t, f_t)$ combines binary task difficulty and continuous fatigue $f_t \in [0, 1]$. Difficulty is modeled stochastically and flips with probability β . (`difficulty_flip_chance`). Fatigue drifts according to action-dependent increments plus penalties for hard tasks and failed outcomes defined by (`src/env.py`).

Observations. The agent intakes noisy views of difficulty and fatigue: $o_t = (d_t + \epsilon_d, f_t + \epsilon_f)$ where $\epsilon \sim \mathcal{N}(0, \sigma^2)$ and remains clipped (bounded) to $[0, 1]$ (`observation_noise`), purposed as an encoding of bounded introspection.

Actions. Two modes are implemented: `habitual` (low effort, lower success on hard tasks) and `deliberate` (higher effort, higher success on hard tasks). NOTE: Delegation is planned but not yet enabled in the current code path.

Transition and reward shaping. Success probabilities are dependent on (d_t, a_t) (e.g., `success_hard_deliberate` = 0.8) and effort costs are subtracted from reward. Outcome rewards follow either r_{easy} or r_{hard} on success with a penalty on failure. Gaussian noise is meant to perturb rewards to avoid outright determinism. Finally, fatigue experience dynamic evolution via action-specific increments with additional relief on success and penalties on failure and hard tasks. An emergent quality of over-deliberation is that its visibly costly later in an episode.

Belief and Particle Filtering. The current baseline uses the following belief: the agent treats the latest observation as the belief mean and tracks running reward-per-effort efficiencies per action. The resultant policy is epsilon-greedy wherein we explore with probability ϵ . Otherwise pick the action with highest penalized efficiency (fatigue and difficulty penalties baked into `Agent._scored_efficiency`). A particle filter drop-in is planned for richer inference over (d_t, f_t) by propagating samples through the known transition model and reweighting on observation likelihood; slots for belief-aware updates are exposed by the separation between environment dynamics (`src/env.py`) and agent policy (`src/agent.py`).

Data and reproducibility. Each run logs trajectories and summary statistics to `results/logs/run_<timestamp>.json` and renders reward, fatigue, and efficiency plots to `results/plots/`. Configurable hyperparameters live in `configs/default.yaml`, ensuring full reproducibility of the reported metrics.

4.2 Policies and Solvers

The agent supports multiple bandit-style solvers that act on reward-per-effort: (i) epsilon-greedy (default) with running estimates; (ii) UCB with exploration bonuses; (iii) Gaussian Thompson Sampling; and (iv) contextual LinUCB using fatigue and difficulty as features. Policy choice and hyperparameters are configured in `configs/default.yaml`. Planned POMDP solvers include POMCP and DESPOT for belief-aware tree search, and point-based value iteration for compact belief approximations. These will let us compare full belief maintenance to these lightweight bandit schedulers.

4.3 Bandit Algorithms

In bandit form, the agent treats difficulty and fatigue observations as context. We currently run ϵ -greedy; extensions will test UCB and Thompson Sampling for non-contextual cases, and LinUCB for contextual bandits that use $(\hat{d}_t, \hat{f}_t)$ to predict action efficiency. These offer scalable alternatives when full belief propagation is too costly.

4.4 Baselines

We compare against simple policies: (i) always-habitual; (ii) always-deliberate; (iii) alternating or rest-heavy schedules that minimize fatigue accumulation; (iv) a difficulty-aware rule (habitual on easy, deliberate on hard); and (v) the learned threshold-fatigue policy from the efficiency-tracking agent. These baselines isolate how much benefit comes from learning versus hand-crafted heuristics. Bandit variants above serve as adaptive baselines that can be swapped in without changing environment code.

5 Experiments

We simulate 25 episodes of horizon 150 using `configs/default.yaml`, sweeping four bandit policies: epsilon-greedy, UCB, Thompson Sampling, and LinUCB. Results are logged per episode and aggregated with mean and standard deviation. Metrics include total reward, cumulative fatigue, total effort, and reward-per-effort ratio. A recent sweep produced:

Policy	Reward $\uparrow$	Fatigue $\downarrow$	Effort	Reward/Effort $\uparrow$
Epsilon-greedy	69.9 ± 9.9	140.2 ± 6.1	49.3 ± 5.4	1.43 ± 0.21
UCB	67.4 ± 11.0	137.8 ± 9.4	43.6 ± 16.1	1.75 ± 0.66
Thompson	69.7 ± 13.0	134.7 ± 10.9	38.2 ± 16.4	2.09 ± 0.75
LinUCB	75.4 ± 10.3	137.4 ± 8.4	47.9 ± 11.4	1.69 ± 0.54

Representative logs live under `results/logs/<policy>/run_<timestamp>.json`. LinUCB yielded the highest mean reward, while Thompson Sampling produced the best reward-per-effort ratios by selectively conserving effort.

5.1 Metrics

We track reward, fatigue trajectory, success rate, action distribution over time, burnout time (when fatigue saturates), and belief accuracy for any belief-aware variants. Reward-per-effort serves as the primary efficiency metric. All trajectories are plotted automatically per policy.

5.2 Figures and Tables

Core plots include reward, fatigue, and efficiency trajectories averaged over episodes (rendered automatically). The following figures are ready for inclusion:

6 Discussion

The efficiency-tracking agent learns to start with more deliberate actions when fatigue is low and gradually shift toward habitual responses as fatigue rises,

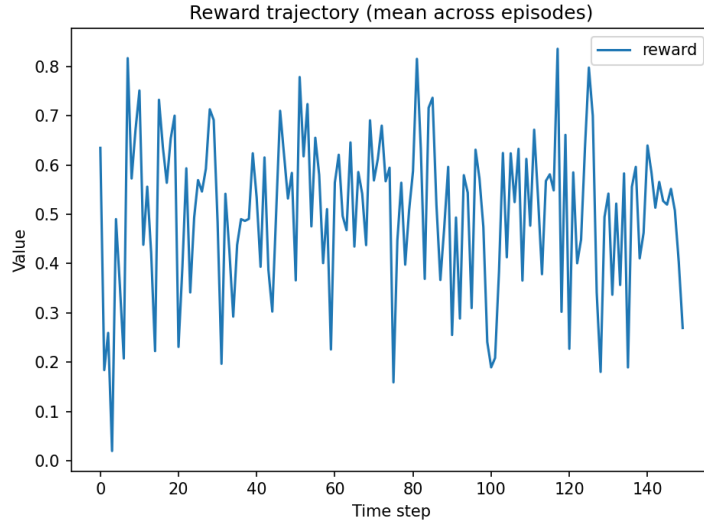


Figure 1: Mean reward trajectory across episodes for LinUCB.

capturing a qualitative signature of decision fatigue. The shaped fatigue dynamics make over-deliberation visibly costly, and the partial observability encourages conservative exploration. Limitations include the minimal belief (no full particle filter yet), binary difficulty, and the absence of explicit delegation/rest actions. Extending to richer observation models and adding rest/delegate actions would let us test value-of-computation hypotheses more directly.

7 Conclusion

We presented a compact, reproducible POMDP testbed for bounded-resource decision making, along with an efficiency-tracking baseline that demonstrates plausible fatigue dynamics and competitive reward-per-effort performance. Future work will add belief-aware planners, contextual bandits, and delegation to probe how agents should schedule computation under uncertainty and fatigue.

Acknowledgments

We thank collaborators and teaching staff for feedback on early prototypes.

References

- [Blain et al.(2016)] B. Blain, G. Hollard, and M. Pessiglione. Neural mechanisms underlying the impact of cognitive fatigue on economic decision making.

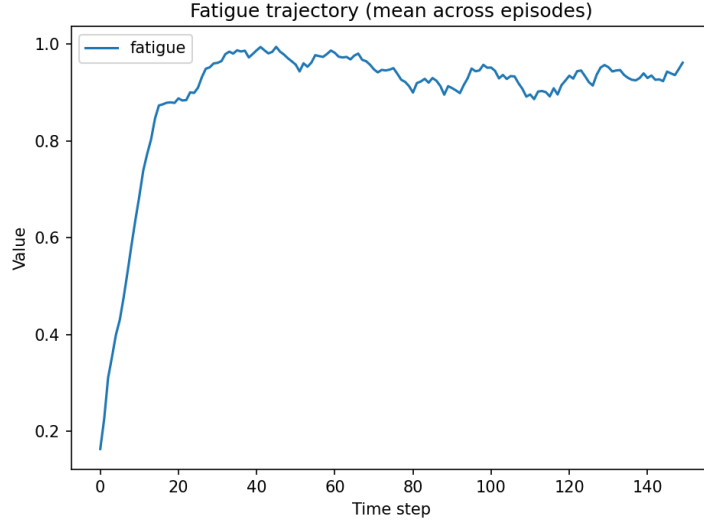


Figure 2: Fatigue trajectory under Thompson Sampling showing slower effort expenditure and later saturation.

- Proceedings of the National Academy of Sciences*, 113(10):U6972–U6982, 2016.
- [Danziger et al.(2011)] S. Danziger, J. Levav, and L. Avnaim-Pesso. Extraneous factors in judicial decisions. *Proceedings of the National Academy of Sciences*, 108(17):6889–6892, 2011.
- [OpenAI(2025)] OpenAI. *ChatGPT (GPT-5) Conversations with Matthew Torre*. Personal communication, December 2025. Available at <https://chat.openai.com/>.
- [Kochenderfer et al.(2022)] M. J. Kochenderfer, T. A. Wheeler, and K. H. Wray. *Algorithms for Decision Making*. MIT Press, 2022.
- [Russell and Wefald(1991)] S. Russell and E. Wefald. *Do the Right Thing: Studies in Limited Rationality*. MIT Press, 1991.
- [Simon(1955)] H. A. Simon. A behavioral model of rational choice. *Quarterly Journal of Economics*, 69(1):99–118, 1955.
- [Sims(2003)] C. A. Sims. Implications of rational inattention. *Journal of Monetary Economics*, 50(3):665–690, 2003.
- [Wood and R  nger(2016)] W. Wood and D. R  nger. Psychology of habit. *Annual Review of Psychology*, 67:289–314, 2016.

8 Code and Reproducibility

All source code, configuration files, and experiment logs for this project are available in a public GitHub repository for full reproducibility and transparency. The repository contains the POMDP environment, agent implementations, experiment scripts, and plotting utilities used to generate all figures and tables in this report.

- **Repository:** https://github.com/MatthewTorre/cs238_final
- **Contents:** Environment (`src/env.py`), agent implementations (`src/agent.py`), configuration files (`configs/`), and experiment results (`results/`).
- **Reproduction:** Run `python3 -m src.main -config configs/default.yaml` to reproduce the results and regenerate reward, fatigue, and efficiency plots.

All experiments were conducted using fixed random seeds for comparability, and results are logged automatically in JSON format for analysis and visualization.

9 Figures:

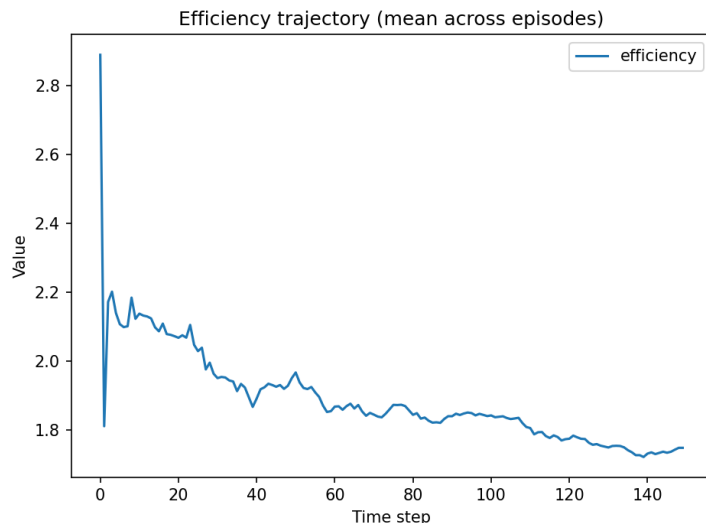


Figure 3: Efficiency (reward per effort) under UCB with wide exploration bonuses early in the episode.

These PNGs are exported to `report/figures/` from the latest runs.

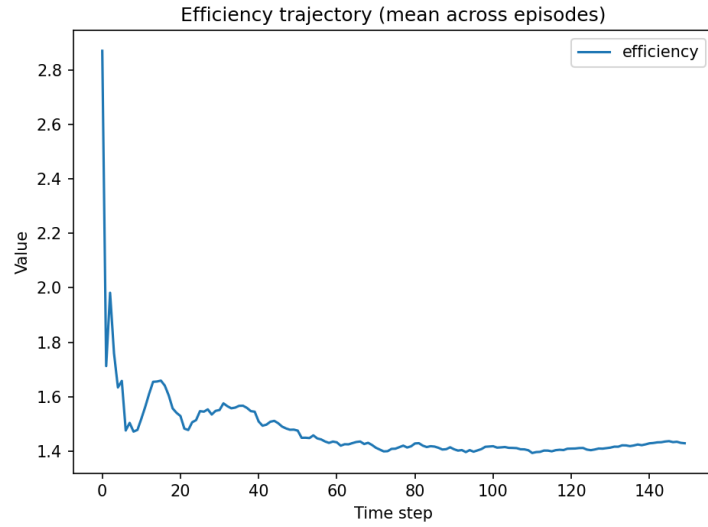


Figure 4: Efficiency trajectory (mean across episodes) for the epsilon-greedy policy. Efficiency starts high due to early deliberate actions but stabilizes around 1.5 as fatigue accumulates.

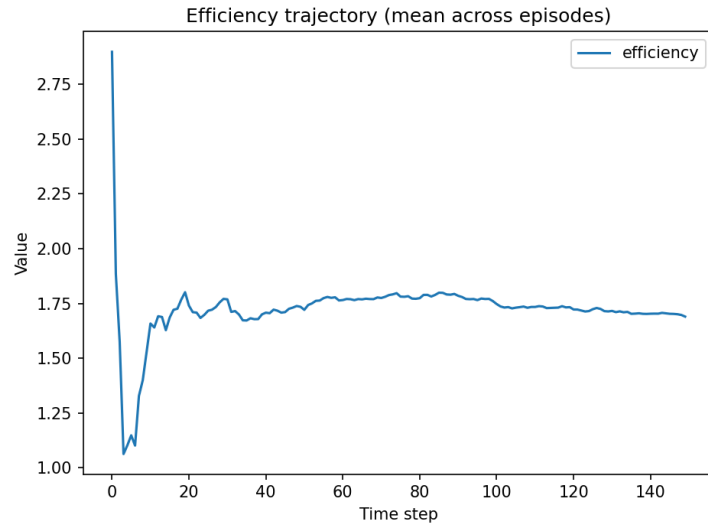


Figure 5: Efficiency trajectory for the LinUCB policy. Efficiency initially rises as the agent balances exploration and exploitation, stabilizing near 1.7 over time.



Figure 6: Efficiency trajectory for the Thompson Sampling policy. After early fluctuations, efficiency converges smoothly around 1.9, reflecting adaptive effort conservation.

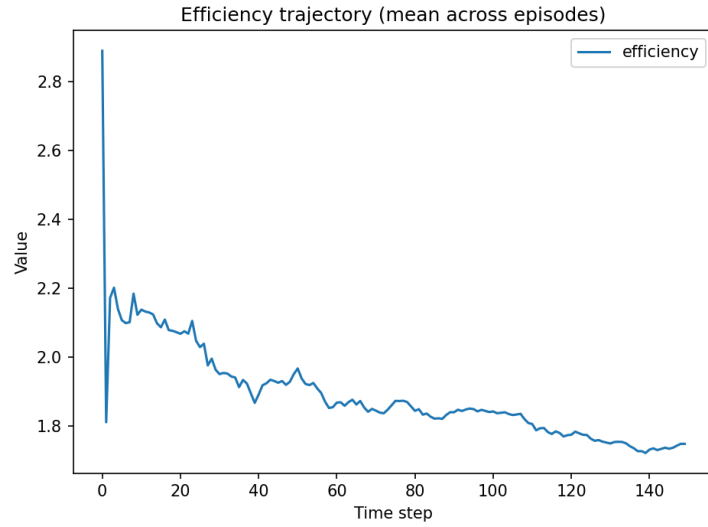


Figure 7: Efficiency trajectory for the UCB policy. Wide exploration bonuses early cause an initial drop, followed by gradual stabilization as the policy exploits higher-efficiency actions.

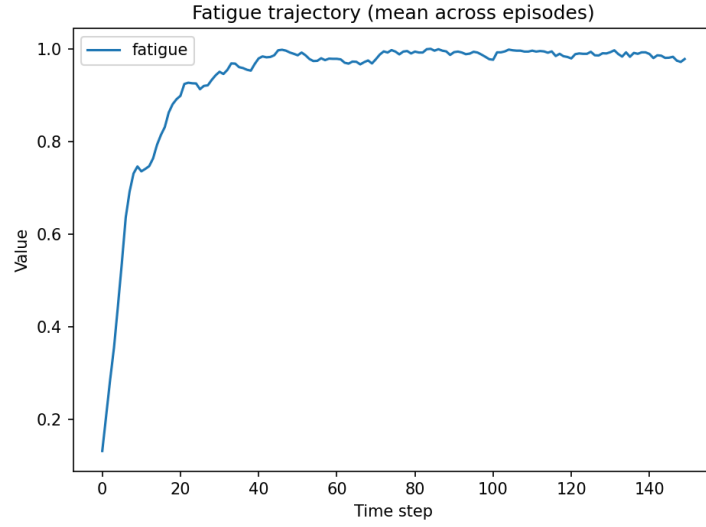


Figure 8: Fatigue trajectory for the epsilon-greedy policy. Fatigue rapidly climbs toward saturation near 1.0, indicating over-deliberation without sufficient rest or recovery.

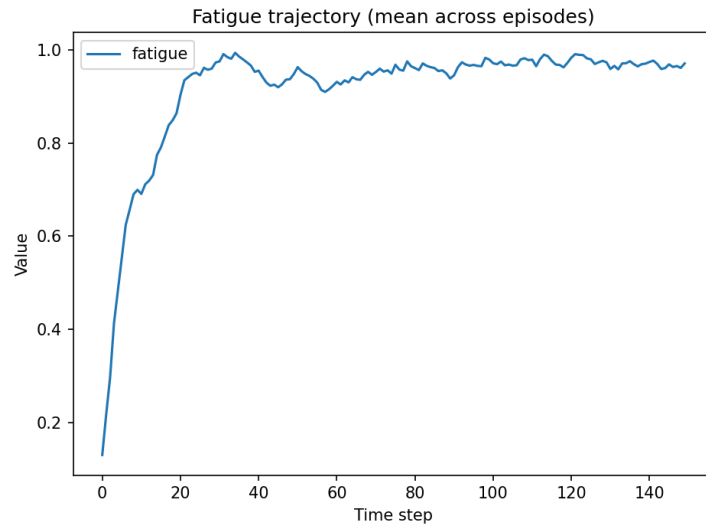


Figure 9: Fatigue trajectory for the LinUCB policy. Fatigue grows quickly but stabilizes slightly below saturation, suggesting more balanced action selection than epsilon-greedy.

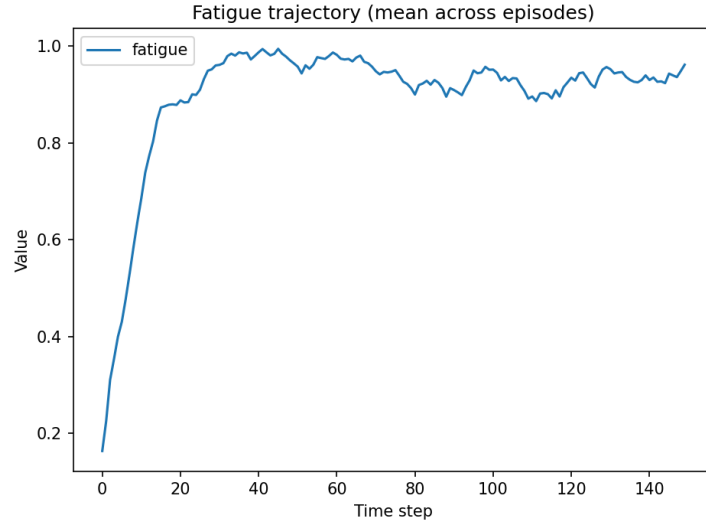


Figure 10: Fatigue trajectory for the Thompson Sampling policy. Fatigue increases more gradually and stabilizes lower, consistent with adaptive exploration that conserves effort.

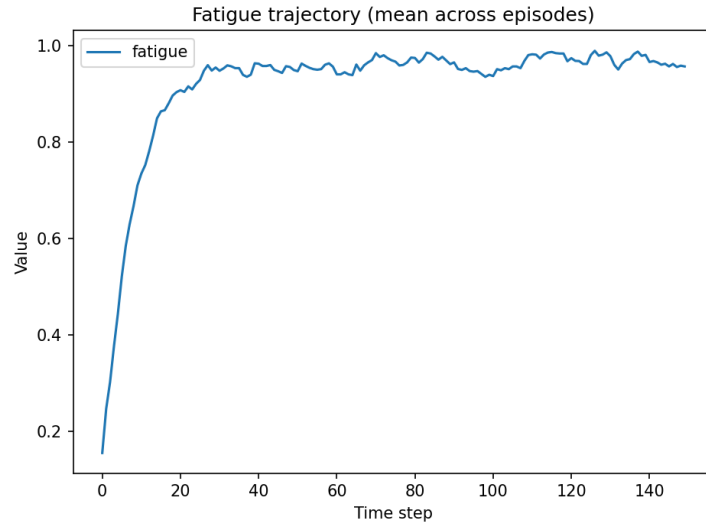


Figure 11: Fatigue trajectory for the UCB policy. Early exploration drives quick fatigue buildup, which then remains high as the agent exploits successful deliberate actions.

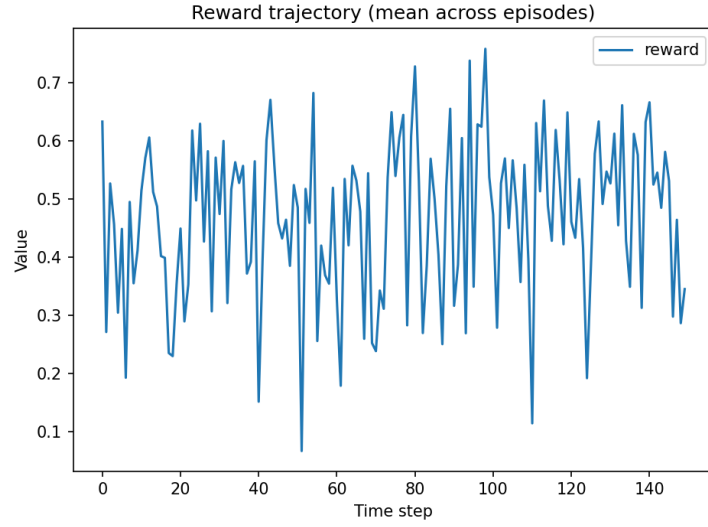


Figure 12: Reward trajectory for the epsilon-greedy policy. Rewards fluctuate widely as the agent alternates between high-effort and low-effort actions with noisy success outcomes.

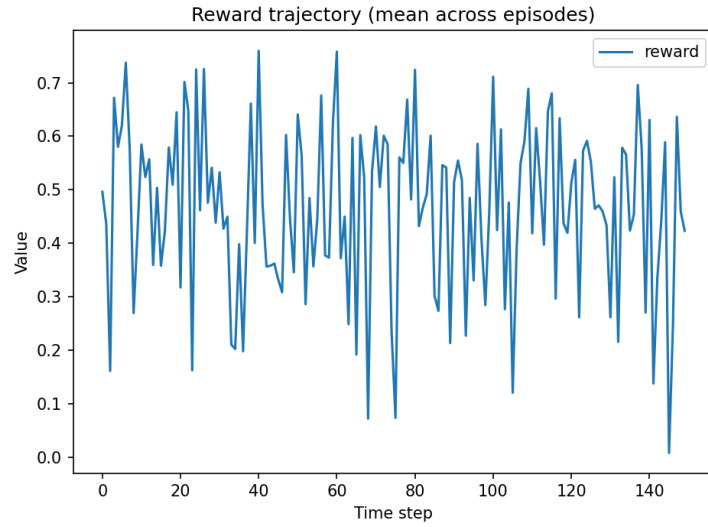


Figure 13: Reward trajectory for the Thompson Sampling policy. Despite stochastic outcomes, the policy maintains higher and more consistent average rewards over time.

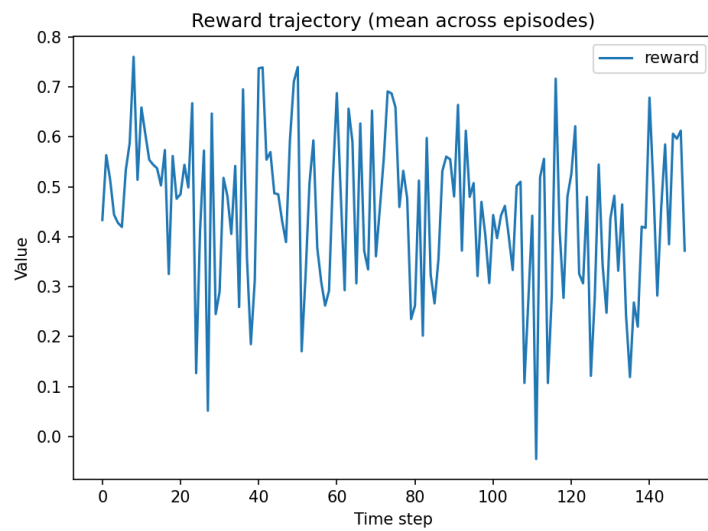


Figure 14: Reward trajectory for the UCB policy. Early reward volatility reflects broad exploration; the later plateau indicates convergence to steady high-reward behavior.