GUIDE: Guided Updates for In-context Decision Evolution in LLM-Driven Spacecraft Operations

Anonymous CVPR submission

Paper ID *****

Abstract

Large language models (LLMs) have been proposed as supervisory agents for spacecraft operations, but existing approaches rely on static prompting and do not improve across repeated executions. We introduce GUIDE, a non-parametric policy improvement framework that enables cross-episode adaptation without weight updates by evolving a structured, state-conditioned playbook of natural-language decision rules. A lightweight acting model performs real-time control, while offline reflection updates the playbook from prior trajectories. Evaluated on an adversarial orbital interception task in the Kerbal Space Program Differential Games environment, GUIDE's evolution consistently outperforms static baselines. Results indicate that context evolution in LLM agents functions as policy search over structured decision rules in real-time closed-loop spacecraft interaction.

1. Introduction

Spacecraft operations remain costly, high-risk, and slow to iterate. Unlike most terrestrial AI deployments, missions cannot easily collect large amounts of new data, reset the environment, or retrain policies after deployment. Yet many operational scenarios—exploration missions, autonomous Guidance, Navigation, and Control (GNC), on-orbit servicing, and non-cooperative interaction—require supervisory reasoning under uncertainty rather than only low-level control. This creates a gap between existing autonomy algorithms and operator-level decision making, motivating systems capable of reasoning under uncertainty and adaptability rather than relying on fixed policies.

Large language models (LLMs) have emerged as promising candidates for this supervisory layer [2, 12]. Beyond text generation, they exhibit reasoning, hierarchical agentic behavior, and structured decision making [15]. Modern LLM systems can also alter behavior at inference time through in-context learning and memory without updating weights, making them attractive where retraining is impractical and deployment constraints dominate.

Deploying an LLM as a supervisory decision agent in a real-time, closed-loop dynamical system introduces two coupled constraints. First, control actions must be generated within strict latency bounds while reasoning over delayed feedback, irreversible actuation, and adversarial interactions. Models capable of extended deliberation incur prohibitive latency for direct integration into the control loop, whereas models that meet real-time requirements cannot perform such reasoning during action selection. Second, interaction with reactive or unknown agents requires adaptation across encounters rather than one-shot decisions. Because such adaptation cannot occur inside the online loop, learning must be separated from action execution. This dual constraint motivates a teacher–student separation, in which a frontier reasoning model performs episodic offline reflection while a lightweight acting model executes real-time control.

Building on the paradigm of large language models as spacecraft operators [2, 3, 12], we introduce GUIDE (**G**uided **U**pdates for **I**n-context **D**ecision **E**volution), a non-parametric policy improvement framework that enables cross-episode adaptation without parameter updates. Instead of optimizing parameters, the system iteratively refines a state-conditioned playbook—a structured set of natural language rules encoding contextual behavior guidance—which functions as a non-parametric policy representation. The acting model remains fixed; policy improvement arises from updating this context rather than model weights, enabling adaptation to adversarial dynamics while preserving real-time execution. We show that structured in-context memory can serve as a learnable policy object improved across episodes without parameter updates.

To study this setting, we consider a real-time multi-agent Capture-the-Satellite scenario using the Kerbal Space Program Differential Games (KSPDG) environment [1]. The environment involves sequential decision-making under delayed observations and non-cooperative interactions, providing a closed-loop setting in which cross-episode adaptation can be evaluated under realistic dynamical constraints.

2. Related Work

Spacecraft autonomy requires sequential decision-making under delayed observations and irreversible dynamics, particularly in adversarial and non-cooperative scenarios [6, 10]. While classical GNC provides reliable stabilization and trajectory tracking when objectives and models are specified [3], it does not address strategic objective selection. This motivates a supervisory decision layer above conventional control, for which recent work has explored LLMs as operator-like decision modules for spacecraft tasks [2, 3, 12].

More broadly, LLM-agent research studies systems combining reasoning, memory, and tool use within action loops [14, 15], but these agents accumulate errors and exhibit prompt sensitivity in long-horizon tasks [14]. A key observation is that behavior can adapt at inference time through in-context learning (ICL): instead of weight updates, the model conditions on demonstrations and memory to modify its decisions [4, 5, 8]. This has been interpreted as inference-time meta-learning [4], and reflection mechanisms incorporate experience without gradient-based retraining [9, 11, 13].

Agentic Context Engineering (ACE) treats structured context as the persistent object of improvement across executions [16], with related self-evolving systems refining structured experience across runs [7, 17]. Our framework is closest in spirit to ACE but differs in objective and role: rather than using evolving context solely to guide behavior, the context itself becomes the learned decision representation that shapes the actions of an online agent. Unlike prior self-evolving agents, which are typically evaluated in episodic software environments, we study context evolution within a continuous closed-loop dynamical control system.

3. Method

We propose GUIDE, a framework for language-based decision agents in real-time closed-loop tasks (Fig. 1). The key idea is to learn state-conditioned behavioral guidance expressed in natural language (a *playbook*) and iteratively improve it using episodic reflection over executed trajectories. Instead of updating model parameters, the method performs a search over structured rules, while a resource-constrained language model executes actions online.

3.1. Problem Setting

We consider sequential decision problems defined over continuous spacecraft dynamics. At each timestep t, the agent observes state s_t and produces a control action a_t. State transitions are governed by orbital mechanics, and thrust actions irreversibly influence future trajectories.

Experiments use the KSPDG environment, which simulates adversarial orbital encounters under varying initial conditions and maneuver policies. We evaluate the Lady–Bandit–Guard (LBG) interception scenario, where the agent controls the bandit spacecraft B, attempting to minimize its distance to the lady spacecraft L while avoiding capture by the guard spacecraft G. At each timestep, the agent observes elapsed time t, spacecraft mass (m_{prop}, m_{total}), and CBCI-frame position and velocity vectors for all relevant spacecraft, yielding

$$s_{t,LBG} = \{t_{LBG}, m_{B,total}, m_{B,prop}, \vec{B}, \dot{\vec{B}}, \vec{L}, \dot{\vec{L}}, \vec{G}, \dot{\vec{G}}\},$$

where $\vec{B}, \dot{\vec{B}}$ denotes the bandit's position and velocity, $\vec{L}, \dot{\vec{L}}$ the lady's, and $\vec{G}, \dot{\vec{G}}$ the guard's. The control action a_t is a three-axis thrust vector with an associated burn duration [1].

Guard behavior varies across scenarios. In LG4, the Guard follows an aggressive pursuit policy, directly chasing the Bandit. In LG6 and LG7, the Guard adopts a defensive strategy, remaining near the Lady and positioning along likely interception trajectories. LG5 introduces stochastic switching between pursuit and defense. These regimes are not specified in prompts or observations and are not known a priori to the LLM policy generator.

3.2. Playbook Representation

Rather than learning a policy $\pi_\theta(a|s)$ through gradient optimization, action selection is conditioned on a language-structured context $\pi(a|s, P)$, where P is a dynamic playbook providing state-conditioned behavioral guidance.

The acting model receives a fixed baseline prompt encoding generic domain instructions (e.g., guard avoidance and prograde alignment for interception). This baseline remains constant across all experiments and is not modified during training.

The playbook P is initially empty and evolves over episodes. It is represented as a collection of bullets or natural language rules, optionally annotated with a state predicate $b_i = (\text{text}_i, \text{condition}_i(s))$, where a bullet becomes active when its condition evaluates to true (e.g., distance thresholds or approach conditions). Active bullets are injected into the prompt at runtime, augmenting the fixed baseline context and modifying the decision distribution.

The learned object in our framework is therefore not the model parameters nor the fixed baseline prompt, but the evolving playbook P.

3.3. Online Decision Agent

The acting language model executes control decisions at every timestep using the playbook-conditioned policy $\pi(a \mid s, P)$. To ensure real-time compliance while preserving structured reasoning, the model is decomposed into two sequential calls within each control cycle.

First, a bounded **reasoning agent** (fixed token budget) produces short-horizon intent conditioned on the current observation and the active playbook bullets. All strategic decision-making occurs within this reasoning step, and the

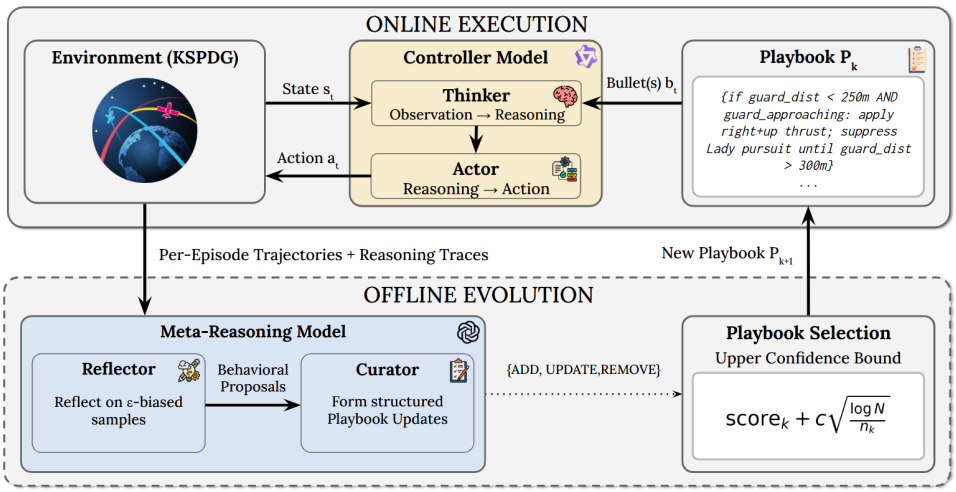

Figure 1. Closed-loop control with cross-episode context adaptation. A fixed acting model executes $\pi(a \mid s, P_k)$ online, while offline reflection updates the playbook via ADD/UPDATE/REMOVE and UCB selects improved versions.

playbook P directly shapes the model's deliberation through state-conditioned behavioral rules injected into context.

Second, an **actor agent** maps the state and the generated intent to a structured control output via a function call tool. This stage produces no free-form reasoning and is restricted to the control action format required by the environment.

This separation bounds reasoning latency while allowing the evolving playbook to shape strategic behavior; the actor simply enforces structured execution within the control interface.

3.4. Offline ϵ-Biased Reflection Sampling

After batches of episodes, trajectories are evaluated by the meta-reasoning LLM using task performance metrics and constraint violations. Reflection inputs are not chosen solely from failures. Instead, an ϵ-selection rule is used: with probability $1-\epsilon$, the system reflects on poorly performing trajectories, and with probability ϵ, it reflects on successful ones. Each reflection input consists of a short telemetry window together with the reasoning trace that produced the corresponding decisions.

A frontier language model (*Reflector*) analyzes the selected episode and proposes behavioral corrections. A second pass (*Curator*) converts these proposals into structured operations applied to a rule-based playbook, producing new rules for a new playbook version P_k:

$$\{\text{ADD}, \text{UPDATE}, \text{REMOVE}\},$$

Multiple playbook versions are maintained simultaneously and evaluated over complete episodes. Selection among playbooks uses an Upper Confidence Bound (UCB1) rule

$$\text{score}_k + c\sqrt{\frac{\log N}{n_k}},$$

where n_k is the number of evaluations of version k and N is the total number of evaluations. This favors playbooks that perform well while still allowing newly generated versions to be tried.

The acting model performs real-time control using the current playbook, while the frontier model operates only between episode batches to revise the playbook from accumulated trajectories. The two models, therefore, serve complementary roles: online execution and offline policy improvement.

4. Results & Discussion

We evaluate four policies across LG4, LG5, LG6, and LG7 scenarios: (1) LLM (static), (2) GUIDE (evolved LLM), (3) Linear Quadratic Regulator (LQR), and (4) Prograde alignment (see Table 1).

Performance is measured using the composite interception score:

$$S = d_{\min}^{LB\,2} + \frac{10^6}{d_{\min}^{BG} + 0.1}, \tag{1}$$

where $d_{\min}^{LB}$ and $d_{\min}^{BG}$ denote the minimum lady-bandit and bandit-guard distances. Lower scores are better. Two-sample t-tests ($\alpha = 0.05$) compare the baseline LLM and GUIDE.

The evolved LLM substantially improves upon the static LLM baseline across all scenarios (Table 1), with statistically significant gains in LG4, LG6, and LG7, and a non-significant trend in LG5. Improvements are largest in LG6-LG7, where the guard adopts a defensive position and interception requires conditional maneuvering, while LG4-LG5 primarily reward direct pursuit. Here, simple prograde alignment already provides a strong open-loop interception heuristic, leaving less room for adaptive improvement.

Table 1. Composite score comparison across LBG scenarios (lower is better). Mean ± std. N = 20

Policy	LG4	LG5	LG6	LG7
LLM (static v0)	$4.15\times10^5 \pm 7.72\times10^5$	$5.37\times10^5 \pm 9.30\times10^5$	$2.28\times10^7 \pm 1.84\times10^7$	$1.95\times10^7 \pm 1.28\times10^7$
GUIDE (best evolved)	$7.22\times10^4 \pm 3.53\times10^4$	$8.49\times10^4 \pm 4.07\times10^4$	$\mathbf{5.07\times10^4} \pm 2.38\times10^4$	$\mathbf{3.49\times10^4} \pm 2.33\times10^4$
p-value vs. v0	*0.043*	*0.143*†	<0.001	<0.001
LQR π	$1.18\times10^6 \pm 8.12\times10^5$	$9.87\times10^5 \pm 9.02\times10^5$	$4.78\times10^5 \pm 1.82\times10^5$	$1.74\times10^5 \pm 4.90\times10^4$
Prograde-Alignment π	$\mathbf{3.73\times10^4} \pm 6.97\times10^4$	$\mathbf{1.60\times10^4} \pm 1.44\times10^4$	$2.92\times10^7 \pm 1.56\times10^7$	$1.62\times10^7 \pm 1.05\times10^7$

† Not significant at α=0.05 (high v0 variance from outlier runs).

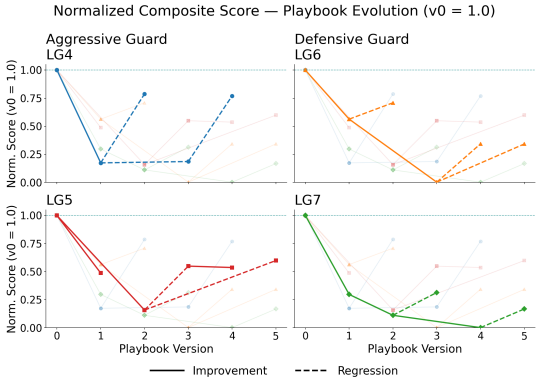

Figure 2. Evolution comparison.

Across all scenarios, the first playbook update consistently identifies the same failure mode: the agent maintains pursuit while the guard closes in to intercept the bandit. Introducing guard-avoidance rules that suspend pursuit and apply lateral thrusts to evade the guard, combined with higher-precision approach behaviors, reduces the composite score by 82%-99% within a few evolutions; Table 2 shows that this arises from both reduced lady-bandit distance and increased guard separation.

Table 2. GUIDE distance metrics (m). Lady: lady-bandit (↓ better). Guard: guard-bandit (↑ better).

	LG4		LG5		LG6		LG7	
Policy	Lady	Guard	Lady	Guard	Lady	Guard	Lady	Guard
LLM v0	237.31	13.34	310.95	**20.91**	291.83	8.41	258.20	8.07
GUIDE (best)	**28.14**	**17.58**	**11.44**	14.56	**88.69**	**30.49**	**52.16**	**29.43**

Improvements arise from structured context updates rather than parameter learning. The offline model encodes corrective strategy into state-conditioned playbook rules executed by the fixed acting model, functioning as a form of in-context distillation that reshapes $\pi(a \mid s, P)$. As shown in Fig. 3, activating a single guard-avoidance bullet changes the trajectory, suspending pursuit and inducing lateral escape before re-engagement.

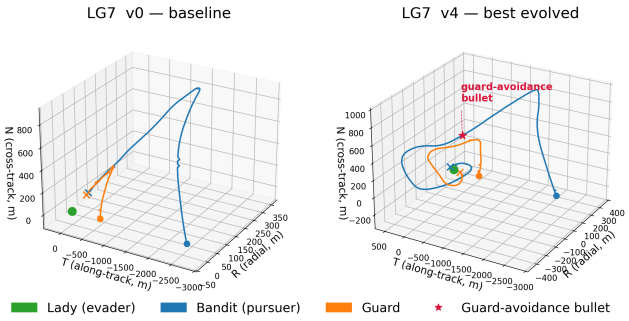

Figure 3. Hill-frame (RTN) trajectories for LG7 (v0 and best).

5. Conclusions & Future Work

GUIDE demonstrates that LLMs can adapt online to dynamic adversarial scenarios through structured context evolution, without weight updates. Beyond static spacecraft operation, context evolution enables cross-episode policy improvement under unpredictable agent interactions, which is most impactful when baseline policies are far from optimal.

Future work includes ablation studies, tighter low-level control integration, algorithmic candidate evolution alongside language policies, and larger-scale playbook evolution to assess convergence behavior.

6. Acknowledgements

This section is intentionally left blank to preserve anonymity during the double-blind review process.

References

[1] Ross E. Allen, Yaron Rachlin, Jessica Ruprecht, Sean Loughran, Jacob Varey, and Herbert Viggh. Spacegym: Discrete and differential games in non-cooperative space operations. In *2023 IEEE Aerospace Conference*, pages 1–12, 2023. 1, 2

[2] Alejandro Carrasco, Marco Nedungadi, Victor Rodriguez-Fernandez, and Richard Linares. *Visual Language Models as Operator Agents in the Space Domain*. AIAA, 2025. 1, 2

[3] Alejandro Carrasco, Victor Rodriguez-Fernandez, and

Richard Linares. Large language models as autonomous spacecraft operators in kerbal space program. *Advances in Space Research*, 76(6):3480–3497, 2025. 1, 2

[4] Julian Coda-Forno, Marcel Binz, Zeynep Akata, Matthew Botvinick, Jane X. Wang, and Eric Schulz. Meta-in-context learning in large language models, 2023. 2

[5] Damai Dai, Yutao Sun, Li Dong, Yaru Hao, Shuming Ma, Zhifang Sui, and Furu Wei. Why can gpt learn in-context? language models implicitly perform gradient descent as meta-optimizers, 2023. 2

[6] Andrew Harris, Thibaud Teil, and Hanspeter Schaub. Spacecraft decision-making autonomy using deep reinforcement learning. In *AAS/AIAA Astrodynamics Specialist Conference*, number AAS 19-447 in Advances in the Astronautical Sciences, Portland, Oregon, USA, 2019. American Astronautical Society. 2

[7] Tu Hu, Ronghao Chen, Shuo Zhang, Jianghao Yin, et al. Controlled self-evolution for algorithmic code optimization, 2026. 2

[8] Chengye Li, Haiyun Liu, and Yuanxi Li. Brewing knowledge in context: Distillation perspectives on in-context learning, 2025. 2

[9] Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. Self-refine: Iterative refinement with self-feedback, 2023. 2

[10] Cameron Mehlman, Joseph Abramov, and Gregory Falco. Cat-and-mouse satellite dynamics: Divergent adversarial reinforcement learning for contested multi-agent space operations. *arXiv preprint arXiv:2409.17443*, 2024. 2

[11] Joon Sung Park, Joseph O'Brien, Carrie Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. Generative agents: Interactive simulacra of human behavior, 2023. 2

[12] Victor Rodriguez-Fernandez, Alejandro Carrasco, Jason Cheng, Eli Scharf, Peng Mun Siew, and Richard Linares. Language models are spacecraft operators, 2024. 1, 2

[13] Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning, 2023. 2

[14] Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, Wayne Xin Zhao, Zhewei Wei, and Ji-Rong Wen. A survey on large language model based autonomous agents, 2023. 2

[15] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models, 2022. 1, 2

[16] Qizheng Zhang, Changran Hu, Shubhangi Upasani, Boyuan Ma, Fenglu Hong, Vamsidhar Kamanuru, Jay Rainton, Chen Wu, Mengmeng Ji, Hanchen Li, Urmish Thakker, James Zou, and Kunle Olukotun. Agentic context engineering: Evolving contexts for self-improving language models, 2026. 2

[17] Yuqing Zhou, Zhuoer Wang, Jie Yuan, Hong Wang, Samson Koelle, Ziwei Zhu, and Wei Niu. Wise-flow: Workflow-induced structured experience for self-evolving conversational service agents, 2026. 2

GUIDE: Guided Updates for In-context Decision Evolution in LLM-Driven Spacecraft Operations

Supplementary Material

7. Per-Version Performance Statistics

7.1. LG4 | Passive Lady, Active Guard

Table 3. LG4 per-version statistics.

Version	Mean score	$\bar{d}_{\text{Lady}}$ (m)	$\bar{d}_{\text{Guard}}$ (m)
v0	4.15×10^{5}	237.3	13.3
v1	$\mathbf{7.22\times10^{4}}$	**28.1**	**17.6**
v2	3.27×10^{5}	192.7	15.9
v3	7.72×10^{4}	36.0	18.4
v4	3.19×10^{5}	173.9	16.7

7.2. LG5 | Passive Lady, Faster Active Guard

Table 4. LG5 per-version statistics.

Version	Mean score	$\bar{d}_{\text{Lady}}$ (m)	$\bar{d}_{\text{Guard}}$ (m)
v0	5.37×10^{5}	311.0	20.9
v1	2.62×10^{5}	159.7	21.0
v2	$\mathbf{8.49\times10^{4}}$	**11.4**	**14.6**
v3	2.95×10^{5}	164.0	19.8
v4	2.88×10^{5}	164.5	38.0
v5	3.21×10^{5}	160.5	13.3

7.3. LG6 | Passive Lady, Blocking Guard

Table 5. LG6 per-version statistics.

Version	Mean score	$\bar{d}_{\text{Lady}}$ (m)	$\bar{d}_{\text{Guard}}$ (m)
v0	2.28×10^{7}	291.8	8.4
v1	1.29×10^{7}	164.8	9.0
v2	1.62×10^{7}	254.7	13.5
v3	$\mathbf{5.07\times10^{4}}$	**88.7**	**30.5**
v4	7.85×10^{6}	106.3	13.2
v5	7.82×10^{6}	123.1	11.5

7.4. LG7 | Passive Lady, Dual Guard

Table 6. LG7 per-version statistics.

Version	n	Mean score	$\bar{d}_{\text{Lady}}$ (m)	$\bar{d}_{\text{Guard}}$ (m)
v0	20	1.95×10^{7}	258.2	8.1
v1	10	5.79×10^{6}	57.4	11.9
v2	15	2.17×10^{6}	59.0	24.4
v3	5	6.13×10^{6}	55.7	13.8
v4	**20**	$\mathbf{3.49\times10^{4}}$	**52.2**	**29.4**
v5	5	3.27×10^{6}	142.0	38.0

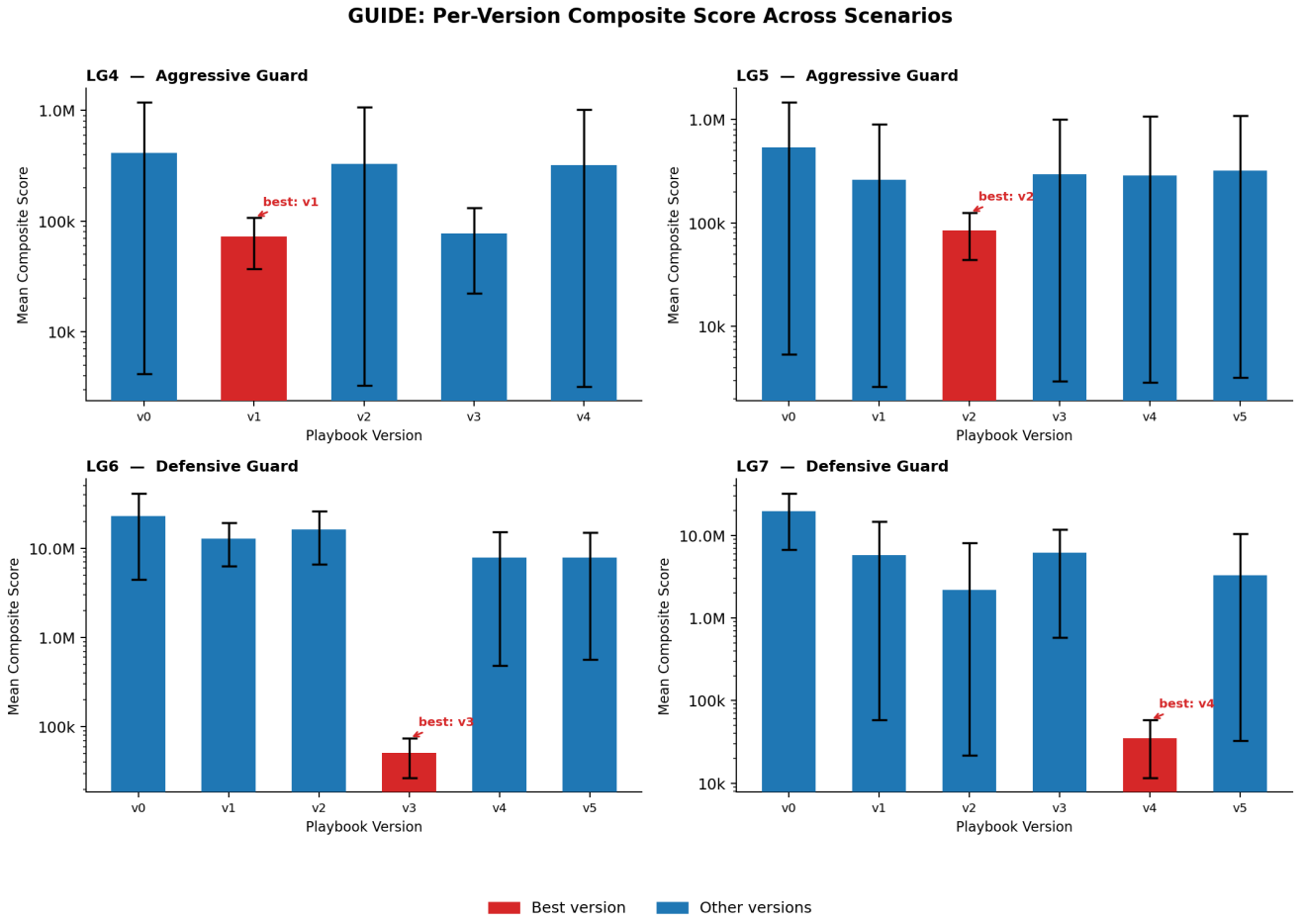

Figure 4. Per-version mean composite score (logarithmic scale, ± 1 std) for all four scenarios. Red bars denote the best-performing version.

8. GUIDE Playbook Structure and Examples

```
id:              <unique id, e.g. guard-avoidance-00001>
section:           <guard_avoidance | approach | ...>
type:            <constraint | rule>
text:            <NL instruction, 1-3 sentences>
conditions:
time:              {min:  <seconds>}      // ignore early orbit phase
guard_distance:    {max:  <metres>}   // guard proximity trigger
guard_approaching:   <bool>        // guard closing flag
target_distance:   {max:  <metres>}   // optional: Lady proximity
velocity:          {max:  <m/s>}      // optional: approach speed
states:            <list of (t, d_lady, v, d_guard) snapshots>
evidence:          <causal narrative written by LLM from failure episode>
episode_history:   [<episode ids that refined this bullet>]
occurrence_count:   <int>      // times this rule fired
```

Figure 5. GUIDE playbook bullet schema. The `conditions` block is a *symbolic guard*: the bullet text is injected into the LLM prompt only when all conditions evaluate to true on the current observation.

id: `guard-avoidance-00001` **section:** `guard_avoidance` **type:** `constraint` **occurrence_count:** 1

text: *"When the Guard is closing inside ∼220 m, stop all forward pursuit and instead apply continuous lateral and/or vertical evasive thrust (right/left and up/down only) until the Guard distance increases above this threshold."*

conditions: `time.min=35 s`, `guard_distance.max=220 m`, `guard_approaching=true`

evidence: *Guard closed from ∼230 m at $t \approx 125.4$ s to 16.9 m at $t = 137.0$ s while forward throttle (`fx=1.0`) was maintained, causing a proximity violation with `guard_distance=30.1 m`.*

Figure 6. **Example 1** — Simple `guard_avoidance` constraint, LG6 v2. Produced after Episode 2 where the Guard closed from 230 m to 17 m in 11.5 s while the Bandit maintained forward throttle.

id: `guard-avoidance-00001` **section:** `guard_avoidance` **type:** `constraint` **occurrence_count:** 3

text: *"After the initial phase ($t \geq 35$ s), apply a two-tiered guard-avoidance regime:*
(1) Caution zone *(`guard_distance` $\lesssim$ 230 m): Immediately stop using any forward or backward thrust that further reduces Guard range relative to the Lady pursuit path. Set longitudinal throttle to neutral (`fx=0`) unless a brief backward input is clearly increasing `guard_distance`. Prioritise lateral and vertical evasive thrust (right/left, up/down only) to start opening separation from the Guard.*
(2) Critical zone *(`guard_distance` $\lesssim$ 180 m, urgency at $\lesssim$ 160 m): Immediately freeze all forward and backward throttle (`ft=0`), overriding any Lady-approach logic. Focus exclusively on lateral and vertical manoeuvres until `guard_distance` reopens beyond ∼230 m.*
Do not resume forward pursuit of the Lady until the Guard is no longer approaching and `guard_distance` has opened beyond ∼230-260 m."

conditions: `time.min=35 s`, `guard_distance.max=230 m`, `guard_approaching=true`, `target_distance.max=900 m`, `velocity.max=25 m/s`

evidence: *In one case, the Guard closed from ∼300 m to 86.1 m while strong forward/braking inputs maintained a nose-to-Lady approach, causing a proximity violation. In another, the Guard closed from ∼270 m to 15.5 m while forward throttle was held nearly continuously, causing a critical proximity violation. Both show that continuing longitudinal motion toward the Lady while the Guard is closing inside a few hundred metres leads to unsafe proximity.*

Figure 7. **Example 2** — Tiered `guard_avoidance` constraint, LG7 v4 (`episode_history`: episodes 1, 2, 4). The two-zone structure emerged across three failure episodes; each zone threshold was calibrated by a different episode and synthesised into a single protocol.

id: `approach-00002` **section:** `approach` **type:** `constraint` **occurrence_count:** 2

text: *"When within ∼120 m of the Lady and still approaching faster than ∼18 m/s, switch to backward throttle or zero forward throttle until relative speed is reduced below 18 m/s to avoid overshoot."*

conditions: `target_distance.max=120 m`, `velocity.min=18 m/s`, `approaching=true`, `time.min=35 s`

evidence: *Bandit was 76 m from Lady at 23 m/s with no braking applied, resulting in overshoot.*

Figure 8. **Example 3** — `approach` braking constraint, LG4 v2. Addresses terminal-phase kinematics when guard-avoidance permits Lady approach but closure speed is too high.