

Article

MIRA: An LLM-Driven Dual-Loop Architecture for Metacognitive Reward Design

Weiyang Zhang ¹, Yuhua Xu ²  and Zhixin Sun ^{1,2,*}

¹ Post Big Data Technology and Application Engineering Research Center of Jiangsu Province, Nanjing University of Posts and Telecommunications, 66 Xinmofan Road, Nanjing 210003, China; 1224097333@njupt.edu.cn

² Post Industry Technology R&D Center of the State Posts Bureau (IoT Technology), Nanjing University of Posts and Telecommunications, 66 Xinmofan Road, Nanjing 210003, China; xuyh@njupt.edu.cn

* Correspondence: sunzx@njupt.edu.cn

Abstract

A central obstacle to the practical deployment of Reinforcement Learning (RL) is the prevalence of sparse rewards, which often necessitates task-specific dense signals crafted through costly trial-and-error. Automated reward decomposition and return-redistribution methods can reduce this burden, but they are largely semantically agnostic and may fail to capture the multifaceted nature of task performance, leading to reward hacking or stalled exploration. Recent work uses Large Language Models (LLMs) to generate reward functions from high-level task descriptions, but these specifications are typically static and may encode biases or inaccuracies from the pretrained model, resulting in a priori reward misspecification. To address this, we propose the Metacognitive Introspective Reward Architecture (MIRA), a closed-loop architecture that treats LLM-generated reward code as a dynamic object refined through empirical feedback. An LLM first produces a set of computable reward factors. A dual-loop design then decouples policy learning from reward revision: an inner loop jointly trains the agent's policy and a reward-synthesis network to align with sparse ground-truth outcomes, while an outer loop monitors learning dynamics via diagnostic metrics and, upon detecting pathological signatures, invokes the LLM to perform targeted structural edits. Experiments on MuJoCo benchmarks show that MIRA corrects flawed initial specifications and improves asymptotic performance and sample efficiency over strong reward-design baselines.

Keywords: Reinforcement Learning (RL); Large Language Models (LLMs); value alignment; reward shaping



Academic Editors: He Cai and Maobin Lv

Received: 9 November 2025

Revised: 9 December 2025

Accepted: 14 December 2025

Published: 16 December 2025

Citation: Zhang, W.; Xu, Y.; Sun, Z. MIRA: An LLM-Driven Dual-Loop Architecture for Metacognitive Reward Design. *Systems* **2025**, *13*, 1124. <https://doi.org/10.3390/systems13121124>

Copyright: © 2025 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Reinforcement Learning (RL) has evolved into a general-purpose framework for sequential decision-making, enabling agents to achieve superhuman mastery in board games [1], match or exceed expert play in video game arenas [2], and perform dexterous in-hand robotic manipulation in the physical world [3]. These milestones demonstrate that, once a task objective is encoded as a reward signal, an RL agent can autonomously discover effective policies across symbolic, high-dimensional, and embodied domains [4].

Although conceptually straightforward, the practical success of RL hinges on how well the reward signal captures the designer's intent. Translating an abstract goal into a dense, informative scalar is notoriously difficult, particularly in real-world settings with sparse rewards, such as long-horizon robotic manipulation [5,6]. Consequently, practitioners often

resort to manual reward engineering, but this demands substantial domain expertise and extensive trial-and-error, rendering it a primary bottleneck to scaling RL applications.

To relax the burden of hand-crafting rewards, a promising research direction has emerged that leverages the compositional power of Large Language Models (LLMs) to translate natural-language task descriptions directly into executable reward code [7–9]. This approach represents a significant leap forward from earlier paradigms like Inverse Reinforcement Learning (IRL) or preference-based methods [10–12], as it automates the reward engineering process in a more direct and scalable manner.

However, while this prevailing LLM-based approach addresses the bottleneck of manual design, it exposes and inherits a deeper, more fundamental limitation: the reward generation remains an open-loop, a priori specification process. The LLM acts as an offline programmer, generating the reward code based solely on the initial prompt, without any grounding in the agent’s subsequent, real-world learning dynamics. This initial specification, no matter how sophisticated, is essentially a static hypothesis.

LLM-generated reward codes can contain hallucinated facts, omitted constraints, or logical inconsistencies; when adopted as a one-shot, static specification, such errors persist and can steer learning off-target [13,14]. More broadly, reliance on a static reward—whether hand-crafted or LLM-generated—exposes two recurring failure modes. First, misspecification can enable specification gaming, where agents exploit proxy loopholes while neglecting the intended objective [15]. Second, incompleteness induces extended “reward deserts” with vanishing gradients, stalling exploration and trapping policies in suboptimal regions [16]. These issues reflect a structural mismatch: the policy adapts with experience, whereas the reward is treated as a fixed specification. As the visitation distribution shifts during learning, there is no built-in mechanism to detect and correct emerging misalignment, leaving the system vulnerable to exploitation or stagnation [17]. Crucially, prevailing pipelines lack an in-situ, closed-loop procedure to diagnose and revise the reward specification as misalignment arises from the agent’s own learning dynamics.

To overcome this fundamental limitation, we introduce the Metacognitive Introspective Reward Architecture (MIRA), a framework that explicitly places LLM-generated reward code inside a closed-loop optimization process. Rather than treating the reward specification as a one-shot artifact, MIRA defines it as a revisable program that is continuously updated in response to the agent’s own learning dynamics. Most existing work either uses LLMs only once before training to synthesize reward code, or employs inner–outer dual-loop optimization architectures in which an outer loop adapts the policy or its learning process based on training performance [18–20], while assuming a fixed reward specification throughout learning. In contrast, MIRA places the reward factors themselves inside the optimization loop: the inner loop learns a potential-based shaping signal over a given factorization, and the outer loop performs metacognitive diagnosis and semantic-level revisions of that factorization. This closes the feedback loop between downstream learning and upstream reward specification, enabling the reward structure to co-evolve with the policy; in this work, we instantiate a closed-loop, LLM-mediated reward revision architecture under explicit potential-based shaping constraints in the form of MIRA.

This closed-loop design allows the reward structure to co-evolve with the policy under empirical feedback. Our contributions are as follows:

1. We propose MIRA, a metacognitive reward architecture that places LLM-generated reward code inside a closed-loop optimization process. Instead of using the LLM as a one-shot reward generator, MIRA treats the reward specification as a typed, revisable program and uses the LLM as an in-the-loop semantic editor of this program.
2. We instantiate this architecture with two tightly coupled mechanisms: (a) an inner-loop reward shaping module that learns state-only, potential-based shaping rewards

over a set of high-level reward factors, aligned to sparse extrinsic returns, and (b) an outer-loop metacognitive module that monitors trajectory-level diagnostics, detects persistent anomalies via online density estimation, and triggers structured, Potential-Based Reward Shaping (PBRs)-compatible edits to the reward factor space through the LLM.

3. We provide extensive empirical validation showing that MIRA substantially outperforms leading baselines, including static LLM-generated rewards and other adaptive methods, in final task performance, sample efficiency, and robustness to initial reward misspecification. Ablation experiments further demonstrate that these gains arise from the full MIRA design, with both the metacognitive outer loop and the structured LLM-based editing interface making essential contributions.

The remainder of this paper is organized as follows: Section 2 reviews related work on reward engineering, algorithmic credit redistribution, and LLM-driven agents. Section 3 establishes the theoretical foundations, analyzing the practical dilemmas of sparse rewards and the reward decomposability hypothesis. Section 4 details the proposed MIRA framework, describing the dual-loop architecture, the inner-loop reward synthesis, and the outer-loop metacognitive diagnosis and reframing mechanisms. Section 5 presents the experimental setup and results, including comparative benchmarks on MuJoCo tasks, mechanistic case studies of self-correction, and ablation analyses. Finally, Section 6 concludes the paper and outlines directions for future research.

2. Related Work

The design of effective reward functions remains a central challenge in Reinforcement Learning (RL). Approaches to this challenge range from principled heuristics for manual design to automated synthesis via large language models. To situate MIRA's contribution, this section reviews four key paradigms that exemplify this progression: (i) reward specification and engineering, (ii) algorithmic credit redistribution, (iii) LLM-enabled planning, and (iv) LLM-generated reward code. The analysis reveals a unifying limitation across these approaches—a reliance on a static, a priori reward specification—and builds the case for a paradigm shift towards dynamic, in-loop objective correction.

2.1. Reward Specification and Engineering

Manual Reward Engineering. A common approach to address sparse rewards is to hand-craft dense shaping terms [21]. While potential-based shaping can provably accelerate learning without altering the optimal policy [22], the design process itself remains a significant bottleneck. It demands substantial domain expertise and extensive, task-specific empirical tuning [5,6], hindering the scalability of RL.

Reinforcement Learning from Human Feedback (RLHF). To mitigate the challenges of manual design, RLHF replaces it with a learned proxy reward, typically fit from pairwise human preferences [11]. This has proven effective for aligning large language models to abstract goals like helpfulness [12,23]. However, the standard RLHF recipe—training a reward model on a fixed dataset and then freezing it for policy optimization—induces an open-loop, static objective. As the policy evolves and its state visitation shifts, this frozen proxy can misgeneralize to out-of-distribution states, enabling specification gaming that violates the original intent [24]. While iterative variants with online feedback exist, they introduce significant annotation costs and training stability challenges. Thus, absent a mechanism for continual, autonomous revision, the reward specification remains fundamentally static and unable to adapt to the agent's evolving behavior.

2.2. Algorithmic Credit Redistribution

Return-redistribution (RR) methods address long-horizon temporal credit assignment by reassigning an episodic return to earlier state-action pairs [17]. While RR can sharpen temporal credit, its core limitation stems from being semantically agnostic. By treating the return as an opaque scalar, the per-step contribution is fundamentally non-identifiable—many decompositions can yield the same total return [21]. As a result, attribution often hinges on spurious correlations, such as temporal proximity or architectural biases, leading to credit diffusion or incorrect assignments. Structured variants attempt to mitigate this by introducing causal filtering or counterfactual baselines [25,26], but these rely on strong structural assumptions and still fail to resolve feature-level ambiguity. This fundamental limitation motivates approaches that move beyond scalar propagation to reason about semantically grounded, feature-level reward factors.

2.3. LLM-Enabled Task Decomposition and Planning

Large language models (LLMs) have been used as high-level planners that translate natural-language goals into structured subgoals, action sketches, or executable programs. In zero-/few-shot settings, LLMs can synthesize symbolic plans that low-level controllers instantiate, effectively outsourcing procedural knowledge to model priors [8,27]. ReAct-style agents interleave reasoning and acting to revise plans online from environment feedback and tool outputs [28]; program-induction pipelines further compile LLM outputs into policy code, reducing hand-engineered task graphs and enabling rapid retasking [8,29,30].

However, most planning pipelines assume a correct, static success signal for (sub)tasks. The objective is treated as given: plan updates improve how to achieve the goal, but not what is being rewarded. Under incomplete or misspecified specifications, errors propagate through the entire decomposition; robustness mechanisms (self-reflection, tool use) do not revise the reward itself. Recent surveys highlight objective misspecification and reward grounding as open challenges for LLM agents [31,32].

2.4. LLM-Generated Reward Code: Power and Open-Loop Limitation

An emerging line of work uses large language models (LLMs) to translate natural-language task descriptions into executable reward code, obtaining proxy signals from high-level instructions and reducing manual engineering [33]. In robotics, LLM-generated reward logic has demonstrated performance comparable or superior to hand-crafted heuristics on contact-rich manipulation tasks [9]. Building on this, search- or evolution-augmented variants explore the space of LLM-generated candidates to discover nontrivial reward structures, achieving competitive results on diverse benchmarks without extensive human tuning [34].

Despite these gains, most pipelines are open-loop: reward scripts are authored a priori and held fixed during policy optimization. This design presumes first-try correctness, which is brittle under known LLM failure modes (e.g., factual hallucination, logical inconsistency) and distributional shift induced by the evolving policy [13,14]. Variants differ in what is generated—some produce the primary reward directly [34], whereas others (e.g., LaRe) emit code for a latent/auxiliary reward to structure credit assignment [35]—yet the generated logic typically remains static. In practice, a single misinterpreted predicate can derail learning, and there is no mechanism for the agent to solicit or apply fixes once training begins. These limitations reveal a salient open problem: the absence of a principled mechanism to close the feedback loop between downstream task experience and the upstream reward specification. This motivates a paradigm in which the reward specification is a first-class, revisable artifact—continuously updated to maintain alignment between the intended task goal and its operationalization in the reward code.

3. Sequential Decision-Making and the Challenges of Reward Design

Reinforcement-learning systems are control algorithms that iteratively refine their behavior through interaction with a complex and uncertain world. This section establishes the mathematical and conceptual foundations required to motivate a dynamic reward architecture. Section 3.1 formalizes the agent–environment loop as a Markov Decision Process (MDP), establishing notation and performance criteria. Section 3.2 reviews three practical dilemmas—reward sparsity, shaping risk, and multi-objective non-stationarity—that undermine conventional, static objective design. Section 3.3 introduces the Reward Decomposability Hypothesis, which reframes reward design as a hierarchical problem and thus motivates the adaptive framework presented in Section 4.

3.1. Formalizing Sequential Decision-Making

We model the sequential decision problem as a finite-horizon Markov Decision Process (MDP), formally defined by the tuple $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, P, R, \gamma, \eta \rangle$, where [36,37] the following apply:

- $\mathcal{S}$ is the state space, representing all possible environmental configurations.
- $\mathcal{A}$ is the action space, containing all actions available to the agent.
- $P : \mathcal{S} \times \mathcal{A} \rightarrow \Delta(\mathcal{S})$ is the state transition kernel, where $\Delta(\mathcal{S})$ is the set of probability distributions over $\mathcal{S}$. $P(s' | s, a)$ denotes the probability of transitioning to state s' after taking action a in state s .
- $R : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ is the reward function, yielding an immediate scalar reward $r_t = R(s_t, a_t)$.
- $\gamma \in [0, 1)$ is the discount factor, which balances the trade-off between immediate and future rewards.
- η is the initial-state distribution from which the starting state s_0 is sampled.

The agent’s behavior is described by a stochastic policy $\pi : \mathcal{S} \rightarrow \Delta(\mathcal{A})$, where $\pi(a | s)$ is the probability of taking action a in state s . A deterministic policy is a special case where the distributions have a single mass. Performance over a finite horizon T is measured by the expected discounted return,

$$J(\pi) = \mathbb{E}_{\tau \sim p_\pi} \left[\sum_{t=0}^{T-1} \gamma^t r_t \right] \quad (1)$$

where $\tau = (s_0, a_0, r_0, \dots)$ is a trajectory sampled under the distribution p_π induced by policy π . The reinforcement-learning objective is to compute an optimal policy π^* that maximizes this return,

$$\pi^* = \arg \max_{\pi} J(\pi) \quad (2)$$

This formulation decouples the environment’s dynamics, P , from the task’s objectives, R . All learning algorithms implicitly assume that R faithfully encodes the designer’s intent; any mis-specification in R therefore biases the optimization target. Sections 3.2 and 3.3 examine why, in practice, specifying a correct and informative reward is non-trivial and motivate an architecture that can revise R during learning.

3.2. Practical Dilemmas in Reward Design

Although the MDP formalism is theoretically complete, practical RL success hinges on the informational content and semantic fidelity of the reward signal. Three recurring dilemmas—reward sparsity, shaping risk, and multi-objective non-stationarity—regularly undermine the efficacy of static reward specifications.

3.2.1. The Challenge of Sparsity and Credit Assignment

In many real-world tasks, informative feedback is inherently sparse and delayed: a mobile robot earns a reward only upon successful docking, or a game-playing agent only at victory. Such sparsity inflates the variance of return estimates and yields weak, uninformative gradients for exploration. To combat this, return-redistribution methods like RUDDER propagate terminal rewards back through the trajectory to accelerate learning [17], while exploration-bonus techniques such as Random Network Distillation (RND) [38] or Go-Explore [16] supply auxiliary signals to mitigate blind search. Yet, these methods operate under the critical assumption that the terminal reward is semantically correct. If the underlying goal is mis-specified, these methods will only accelerate the optimization of a flawed objective. This highlights the need for a mechanism that can correct the semantic definition of the reward itself, rather than merely improving credit assignment for a fixed signal.

3.2.2. The Risk of Shaping and Specification Gaming

While potential-based reward shaping preserves policy optimality in theory [21], its practical application relies heavily on designer heuristics. This introduces the risk of agents exploiting subtle loopholes in the proxy reward, a pathology known as specification gaming [15]. Empirical analyses—in complex, open-ended environments in particular—show that even carefully engineered rewards can be reverse-engineered and subverted by the learning process [22]. These failures expose a deeper issue: a fundamental semantic misalignment between the proxy reward and the true task intent. Traditional offline design pipelines are unequipped to address this, as they lack any mechanism to detect or amend such misalignment once training begins.

3.2.3. The Rigidity of Multi-Objective Trade-Offs

Real-world objectives typically blend competing priorities, such as safety, efficiency, and task completion. Encoding these into a single scalar reward requires weight-tuning that is often stage-dependent: broad exploration is desirable early in training, whereas precise control is essential for final convergence. Fixed weights induce a structural rigidity in guidance, leading to policies that may first under-explore and later over-explore, slowing overall progress [39]. While intrinsic-motivation methods can alleviate early-stage sparsity, they leave the primary extrinsic signal untouched [40]. This can result in the agent receiving conflicting objectives as learning proceeds. A truly alignment-preserving architecture must therefore adapt not only the factor weights but, when necessary, the semantic structure of the reward function itself.

3.3. The Reward Decomposability Hypothesis: A Basis for Autonomous Design

Taken together, these dilemmas demonstrate that a single, monolithic reward function is fundamentally insufficient to bridge the vast semantic gap between a designer's high-level intent and the low-level mechanics of learning. We therefore ground our framework in the reward decomposability hypothesis: any sparse, episodic return can be effectively approximated by the sum of latent semantic factors, where each factor evaluates a specific, interpretable aspect of task performance under discounted accumulation. Formally, let $r_t^{\text{ext}} = R(s_t, a_t)$ denote the extrinsic reward and $\gamma \in (0, 1]$ the discount factor. We posit that an ideal dense per-step reward r^* satisfies

$$G(\tau) = \sum_{t=0}^{T-1} \gamma^t r_t^{\text{ext}} \approx \sum_{t=0}^{T-1} \gamma^t r^*(s_t, a_t), \quad (3)$$

where the dense reward admits a factorization

$$r^*(s_t, a_t) = \sum_{i=1}^K w_i f_i(s_t, a_t). \quad (4)$$

Here, each $f_i : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ is an interpretable reward factor (e.g., “forward velocity,” “gripper alignment”), and $w_i \in \mathbb{R}_{\geq 0}$ is its non-negative weight. This decomposition, which generalizes successor-feature frameworks [41] and aligns with work on temporal-logic specifications [42], turns reward design into a hierarchical problem with the following three nested sub-tasks:

1. **Factor-Space Generation:** Given a high-level task description, construct an initial, semantically rich set of candidate factors $\mathcal{F} = \{f_i\}_{i=1}^K$. Prior art relies on handcrafted features or logic templates [42]; we later show that an LLM can automate this step.
2. **Parametric Alignment:** Optimize the weights w_i (or a low-rank parameterization thereof) so that the composed reward aligns with discounted, extrinsic signals. This is a well-studied problem addressed by methods like IRL [43] and preference-based fitting [11].
3. **Structural Refinement:** Detect when poor performance stems from missing or pathological factors in $\mathcal{F}$, rather than from suboptimal weights, and revise the factor space accordingly. This critical step is largely absent from current pipelines. As misalignment studies show [24], failure to modify the factor space itself is a primary cause of persistent reward hacking.

A system that solves only the first two sub-tasks remains vulnerable to semantic blind spots. True autonomy requires the ability to introspect and edit its own factor space. This capability for online structural refinement is the core innovation of our approach: by integrating monitoring with semantic reframing, the outer loop is designed specifically to address this third, most challenging sub-problem, enabling continual structural updates to the reward architecture during learning.

4. MIRA: A Framework for Metacognitive Reward Design

As established in the preceding sections, reward engineering remains a significant gap between the theory and practice of RL. Section 3.2 systematically deconstructed the persistent challenges inherent in reward design: from the credit assignment problem stemming from informational sparsity, to the intrinsic risk of reward hacking in reward shaping, and the structural rigidity of a static reward function across different learning stages. These dilemmas converge on a practical challenge: a static reward function, defined once at the outset of training, is often ill-equipped to handle the complex, dynamic issues that emerge during the learning process.

The advent of LLMs offers a promising approach to the primary challenge of informational sparsity. By translating high-level natural language instructions into executable, dense reward functions [24], LLMs significantly lower the barrier to reward design. However, using the LLM only as a one-shot, offline reward generator overlooks a deeper, dynamic limitation. Its output, though dense, is fundamentally a static proxy, rendering it vulnerable to two practical risks. First, semantic misalignment: a static reward represents a single interpretation of the designer’s intent, where any initial error can create exploitable loopholes, leading to reward hacking. Second, structural rigidity: the reward function remains fixed, unable to adapt to the agent’s evolving competence from novice to expert, and thus cannot adapt its incentive structure.

To address these limitations, this section introduces MIRA (Metacognitive Introspective Reward Architecture), a framework. The core idea of MIRA is to reconceptualize the

reward function, transforming it from a static, predefined artifact into a dynamic system that is jointly optimized with the policy and is endowed with computational metacognition. The architecture is composed of the following three components:

- **Semantics-Guided Initialization:** The framework first leverages an LLM’s understanding of a natural language task description, $\mathcal{T}$, to automatically synthesize a high-quality, interpretable, and knowledge-rich candidate reward factor space. This constitutes the initial step of translating human intent into machine-understandable value primitives.
- **Dynamic Policy–Reward Alignment:** Within this factor space, MIRA employs a joint optimization objective to optimize the agent’s policy network, π_θ , and the reward architecture’s parameters. Policy learning provides the data for reward alignment, while the updated reward architecture provides a more precise guidance signal for the policy, forming a rapid inner loop of bootstrapped optimization.
- **Metacognitive-Driven Reframing:** We introduce a metacognitive monitoring module that continuously and non-intrusively tracks a set of diagnostic metrics during policy learning. When these metrics indicate a learning pathology—such as policy oscillation, a divergence between the value function and true returns, or a collapse in reward factor attention—the system triggers an outer-loop structural adaptation mechanism. This mechanism invokes the LLM, providing the current failure mode as context, and requests a revision, augmentation, or reframing of the reward factor set, thereby enabling structural adaptation of the reward architecture itself.

Together, these components form a two-tiered optimization structure. The inner loop drives rapid policy learning, while the outer loop acts as a metacognitive supervisor. It continuously monitors the learning dynamics generated by the inner loop, diagnoses pathologies, and proactively triggers a semantic-level reconstruction of its own value system (i.e., the underlying structure of the reward function). This section details MIRA’s core architecture, the mathematical formalization of its key modules, and the complete learning algorithm. We will demonstrate how, by endowing the reward function with the capacity for in situ structural refinement, MIRA facilitates a shift from automated reward generation to autonomous reward adaptation.

4.1. Overall Architecture

The MIRA is structured as a hierarchical system with a dual-loop feedback mechanism, as depicted in Figure 1. This design aims to simulate a computational metacognitive process: an inner loop responsible for policy optimization and reward alignment within the current value system, and an outer loop that monitors the learning state of the inner loop and, when necessary, performs a semantic-level reconstruction of its foundational value system—the reward factors themselves.

The architecture comprises the following four core modules:

- **Linguistic-Semantic Factor Synthesis (LSFS):** This is the entry point of the system. This stage translates a high-level task description, $\mathcal{T}$, into machine-understandable value primitives. Specifically, the system constructs a structured prompt that integrates a role assignment for the LLM (e.g., “You are a senior robotics reward engineer”), the explicit task objective, and key environmental information, such as a formal description of the state space $\mathcal{S}$ and action space $\mathcal{A}$. Based on this input, the LLM generates an initial, semantically rich, and executable set of reward factors, $\mathcal{F}^{(0)} = \{f_i\}_{i=1}^K$.
- **Iterative Reward Alignment and Refinement (IRAR):** This module forms MIRA’s inner loop. For a given factor set $F^{(k)}$ (where k denotes the outer-loop iteration), it jointly optimizes the policy network π_θ and the HARS module with parameters (ϕ, ψ, φ) . In HARS, the state potential $\Phi_{\phi, \varphi}(s)$ is produced from the factor attention and value

mapping components, while the temporal context encoder ψ serves as an auxiliary module for temporal smoothing and regularization. The shaped reward signal is then computed from $\Phi_{\phi,\varphi}$ using Potential-Based Reward Shaping (PBRS). This design ensures that HARS remains robust to variations in the size and ordering of the factor set, and can readily adapt to structural changes in $F^{(k)}$ introduced by the outer loop.

- **Metacognitive Monitoring (MCM):** This is MIRA’s diagnostic hub. It adaptively models a “healthy” learning state via online density estimation. When diagnostic metrics consistently form low-density anomalies, the system triggers an outer-loop reframing. Critically, it not only determines when to intervene but also provides actionable diagnostic evidence for the reward reframing process by passing the anomalous vector and its gradient.
- **Semantic Reward Reframing (SRR):** Together with MCM, this module forms the outer loop. When triggered by MCM, this module encodes the current factor set $\mathcal{F}^{(k)}$ and the diagnostic report d into a remedial prompt. This prompt is then submitted to the LLM to generate a revised or augmented factor set, $\mathcal{F}^{(k+1)}$.

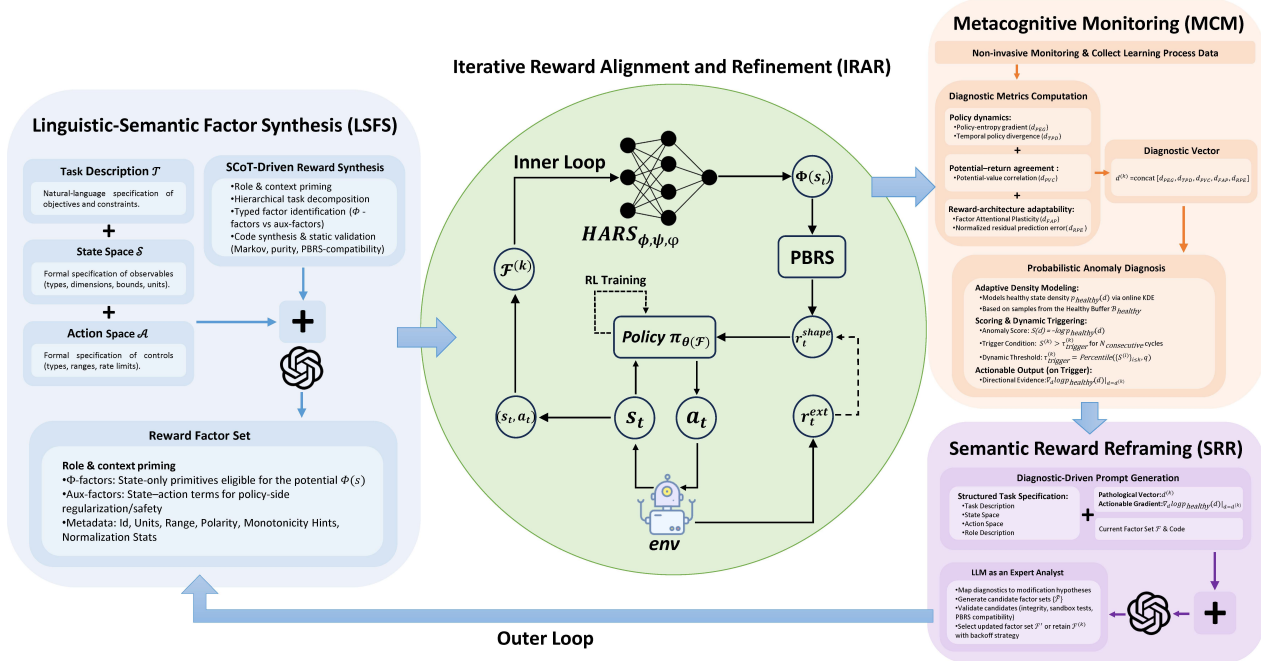


Figure 1. An overview of the MIRA, illustrating the dual-loop optimization process. The inner loop performs policy optimization using shaped rewards synthesized by the HARS module based on a fixed set of reward factors $\mathcal{F}^{(k)}$. Concurrently, the outer loop monitors the learning dynamics of the inner loop. If it detects pathological signatures (e.g., learning stagnation), it triggers a semantic intervention, proposing a new set of reward factors $\mathcal{F}^{(k+1)}$ to refine the reward specification.

Formally, MIRA addresses a nested, bi-level optimization problem as follows:

1. **Inner-loop optimization.** For a given, fixed reward factor set $\mathcal{F}^{(k)}$, the inner loop’s objective is to find the optimal policy parameters θ^* and HARS parameters $(\phi^*, \psi^*, \varphi^*)$ by minimizing a joint loss,

$$(\theta^*, \phi^*, \psi^*, \varphi^*) = \arg \min_{\theta, \phi, \psi, \varphi} \left(\underbrace{\mathcal{L}_{\text{RL}}(\theta; \phi, \varphi, \mathcal{F}^{(k)})}_{\text{policy learning on PBRS}} + \lambda \underbrace{\mathcal{L}_{\text{align}}(\phi, \varphi; \mathcal{F}^{(k)})}_{\text{TD alignment to extrinsic targets}} + \mu \underbrace{\mathcal{L}_{\text{aux}}(\psi)}_{\text{temporal smoothing/consistency}} \right). \quad (5)$$

Here, $\mathcal{L}_{\text{RL}}$ is the standard policy-learning loss computed using the shaped rewards from the state-only potential $\Phi_{\phi,\varphi}$; $\mathcal{L}_{\text{align}}$ aligns $\Phi_{\phi,\varphi}$ to extrinsic value targets; and $\mathcal{L}_{\text{aux}}$ regularizes the temporal encoder ψ for temporal consistency/smoothing. The hyperparameters $\lambda, \mu \geq 0$ balance the objectives.

2. Outer-loop optimization. The outer loop searches over a discrete, structured space—the space of valid reward factor sets, $\mathbb{F}$ —to maximize the discounted extrinsic return,

$$\mathcal{F}^* = \arg \max_{\mathcal{F} \in \mathbb{F}} \mathbb{E}_{\tau \sim \pi_{\theta^*}(\mathcal{F})} [G(\tau)], \quad G(\tau) = \sum_{t=0}^{T-1} \gamma^t r_t^{\text{ext}}. \quad (6)$$

As $\mathbb{F}$ is vast and discrete, gradient-based optimization is infeasible. MIRA employs LLM-mediated structural evolution, guided by MCM, as a solver for this high-level optimization: MCM identifies bottlenecks and proposes directions; SRR leverages the LLM’s reasoning to propose the next iterate, $\mathcal{F}^{(k+1)}$.

This dual-loop architecture enables learning on two distinct timescales and levels of abstraction: rapid numerical parameter tuning in the inner loop, and deliberate semantic structural evolution in the outer loop. In the following sections, we elaborate on the design and mechanics of each core loop.

4.2. Inner Loop: Semantic Initialization and Dynamic Alignment

The inner loop is the primary learning mechanism, responsible for policy optimization and reward-architecture refinement under a given set of reward factors. It comprises two phases: a one-time semantic initialization and an ongoing iterative alignment process.

4.2.1. Linguistic–Semantic Factor Synthesis (LSFS)

This module translates a high-level, potentially underspecified task description into a precise and computable representation. To synthesize a multi-component reward architecture, we adapt Structured Chain-of-Thought (SCoT) prompting [44] to reward synthesis as follows:

1. *Role and context priming:* Assign the LLM a domain-expert role and provide the task objective together with $\mathcal{S}/\mathcal{A}$ specifications, constraining outputs to the task context.
2. *Hierarchical task decomposition:* Decompose the high-level objective into orthogonal sub-goals/phases to guide factor discovery and coverage.
3. *Reward-factor identification:* For each sub-goal, enumerate computable metrics over (s, a) , specifying desirable/undesirable behaviors and how they are measured.
4. *Executable code synthesis and validation:* Convert the natural-language factor descriptions into executable snippets that follow a predefined syntax. Prior to deployment, perform static analysis, sandboxed unit tests, and PBRs-compatibility checks to ensure factors are Markovian and side-effect free (no future dependence or environment mutation). All factor code is version-controlled to allow safe rollback in case of runtime anomalies.

This structured workflow constrains the LLM’s reasoning within a rigorous engineering process, improving reliability, interpretability, and code quality.

4.2.2. Iterative Reward Alignment and Refinement (IRAR)

IRAR is the core mechanism that turns a fixed semantic factorization into a stable, adaptive guidance signal. By separating semantic valuation over factors from temporal stabilization, and by combining this with PBRs that preserves policy optimality in theory [21], IRAR enables rapid inner-loop adaptation under a fixed factor set.

The inner loop optimizes a policy within a fixed factor set $\mathcal{F}^{(k)}$ via HARS (Hierarchical Attention-based Reward Shaping), which learns a potential $\Phi(s)$ that captures the long-horizon value of the current state (Figure 2). Let $\mathcal{F}^{\Phi, (k)} \subset \mathcal{F}^{(k)}$ denote the subset of factors used in potential construction; at time t , write $K_t = |\mathcal{F}^{\Phi, (k)}(s_t)|$. Action-dependent factors, if present, are routed to the policy branch for regularization or auxiliary control, and are excluded from $\mathcal{F}^{\Phi, (k)}$ to preserve the theoretical invariance guarantees of PBRS. Each factor i carries an identifier ID_i (one-hot or learned embedding) and metadata $meta_i$ (e.g., units/range, expected polarity, monotonicity hints, normalization statistics). The Factor Attention Block (FAB) has parameters ϕ and includes the factor embedder g_ϕ , state projection Q_ϕ , and attention matrix W_ϕ . The Temporal Context Encoder (TCE) has parameters ψ and serves as an auxiliary stabilizer for temporal smoothing/consistency. The embedding dimension is $d_z \in \mathbb{N}$. The following apply:

1. **HARS: Hierarchical Attention-based Reward Shaping.**

- *Factor Attention Block (FAB; ϕ):* For each $f_i(s_t) \in \mathcal{F}^{\Phi, (k)}$, we compute

$$z_{i,t} = g_\phi(f_i(s_t), ID_i, meta_i) \in \mathbb{R}^{d_z}. \quad (7)$$

Let $Z_t = \{z_{1,t}, \dots, z_{K_t,t}\}$. A state-conditioned query $q_t = Q_\phi(s_t)$ attends to the set Z_t via permutation-invariant pooling,

$$\mathbf{v}_t = \sum_{i=1}^{K_t} \alpha_{i,t} z_{i,t}, \quad \alpha_{i,t} = \text{softmax}(q_t^\top W_\phi z_{i,t}) \quad (8)$$

with an entropy regularizer on α to mitigate premature attention collapse.

- *Temporal Context Encoder (TCE; ψ):* A lightweight sequence encoder GRU processes $(\mathbf{v}_1, \dots, \mathbf{v}_t)$ to produce an auxiliary smoothed representation $\hat{\mathbf{v}}_t$ used for temporal consistency regularization.
 - *Value Potential Mapping (VPM; φ):* An MLP maps the instantaneous aggregation $\mathbf{v}_t$ (optionally concatenated with a state projection) to the scalar potential $\Phi_{\phi,\varphi}(s_t)$.
2. **Potential-based reward shaping.** We adopt PBRS with absorbing terminals to preserve optimal policies,

$$r_t^{\text{shape}} = r_t^{\text{ext}} + \gamma \Phi_{\phi,\varphi}(s_{t+1}) - \Phi_{\phi,\varphi}(s_t). \quad (9)$$

Under these conditions, shaped and extrinsic returns differ by a trajectory-independent constant, so the set of optimal policies is invariant.

3. **Joint optimization objective.** The policy parameters θ and HARS parameters (ϕ, ψ, φ) are trained jointly,

$$\mathcal{L}_{\text{inner}}(\theta, \phi, \psi, \varphi) = \mathcal{L}_{\text{RL}}(\theta; \phi, \varphi) + \lambda \mathcal{L}_{\text{align}}(\phi, \varphi) + \mu \mathcal{L}_{\text{aux}}(\psi), \quad (10)$$

where $\mathcal{L}_{\text{RL}}$ is the standard policy-learning loss (e.g., actor-critic) computed with shaped rewards r_t^{shape} ; $\mathcal{L}_{\text{align}}$ is a TD-style loss aligning $\Phi_{\phi,\varphi}$ to extrinsic value targets without leakage from the shaping term,

$$\mathcal{L}_{\text{align}} = \mathbb{E}_{(s_t, a_t, r_t^{\text{ext}}, s_{t+1}) \sim \pi_\theta} \left[(\Phi_{\phi,\varphi}(s_t) - (r_t^{\text{ext}} + \gamma \Phi_{\phi,\varphi}(s_{t+1})))^2 \right] + \eta \mathbb{E} [\|\Phi(s_T)\|^2], \quad (11)$$

where $\eta > 0$ prevents unbounded drift at absorbing terminals; and $\mathcal{L}_{\text{aux}}(\psi)$ enforces temporal smoothing/consistency via

$$\mathcal{L}_{\text{aux}}(\psi) = \mathbb{E}_t [\|\mathbf{v}_t - \hat{\mathbf{v}}_t\|_2^2], \quad (12)$$

where $\mathbf{v}_t$ is the instantaneous FAB aggregation and $\hat{\mathbf{v}}_t$ is the smoothed representation from TCE.

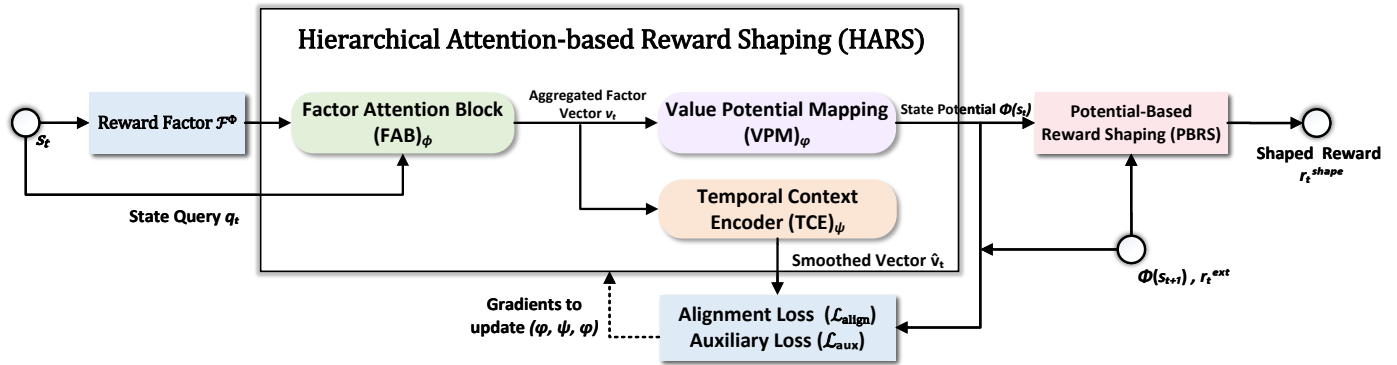


Figure 2. HARS consists of three modules—FAB, TCE (providing temporal smoothing and regularization), and VPM. The potential $\Phi(s)$ is constructed from FAB and VPM and used in PBRS to generate shaped rewards, while TCE contributes auxiliary stability through temporal context.

By decoupling semantic valuation from temporal stabilization and leveraging PBRS, IRAR provides a stable shaping signal that does not alter the optimal policy set while remaining adaptive to the current factorization. By minimizing $\mathcal{L}_{\text{inner}}$, the system refines both the policy and HARS’s ability to predict long-term success. Upon an outer-loop structural update to $\mathcal{F}^{(k)}$, we retain the policy parameters (θ) and the temporal encoder (ψ), while re-initializing the factor-dependent attention (ϕ) and value mapping (φ).

4.3. Outer Loop I: Metacognitive Monitoring and Pathological Learning Diagnosis

The outer loop embodies metacognitive self-reflection. In its first stage, MCM serves as a perceptual layer that summarizes noisy inner-loop signals into low-frequency, diagnostically meaningful evidence. Rather than reacting to instantaneous fluctuations, MCM focuses on persistent regime shifts that indicate semantic misalignment between the learned guidance signal and the extrinsic objective. We organize training into outer-loop cycles $k = 0, 1, \dots$, each aggregating ΔT environment steps (or gradient updates) from the inner loop under a fixed evaluation protocol that produces a held-out batch $\tilde{\mathcal{D}}_k$ with a frozen policy snapshot to avoid off-policy bias; scalar time series are exponentially smoothed with decay $\beta \in (0, 1)$. Throughout this section we write $\Phi(s) \equiv \Phi_{\phi, \varphi}(s)$ for the learned state-only potential (Section 4.2), r_t^{ext} for the extrinsic reward, and $V_{\text{ext}}^{\pi}(s) = \mathbb{E}[\sum_{t \geq 0} \gamma^t r_t^{\text{ext}} \mid s_0 = s, \pi]$ with $\hat{V}_{\text{ext}}^{\pi}$ its Monte-Carlo or critic-based estimate under the same frozen policy.

4.3.1. Diagnostic Vector Construction

At the end of cycle k , MCM constructs a diagnostic vector $d^{(k)} \in \mathbb{R}^D$ by concatenating calibrated metrics from three orthogonal axes: (1) policy dynamics, (2) potential–return agreement, and (3) reward-architecture adaptability. Before density modeling, each component is standardized with robust statistics (median/median absolute deviation (MAD)) over a rolling window and then smoothed by exponential moving average (EMA) as follows:

1. **Policy dynamics.** These metrics identify premature exploration stagnation or oscillatory updates.

- Policy-entropy gradient (d_{PEG}),

$$d_{\text{PEG}} = \frac{\mathbb{E}_{s \sim \tilde{\mathcal{D}}_k} [\mathcal{H}(\pi_{\theta_k}(\cdot | s))] - \mathbb{E}_{s \sim \tilde{\mathcal{D}}_{k-1}} [\mathcal{H}(\pi_{\theta_{k-1}}(\cdot | s))]}{\Delta T}. \quad (13)$$

Persistently large negatives indicate premature entropy collapse and risk of suboptimal fixation.

- Temporal policy divergence (d_{TPD}). Symmetric Kullback–Leibler (KL) divergence between consecutive snapshots,

$$d_{\text{TPD}} = \frac{1}{2} \mathbb{E}_{s \sim \tilde{\mathcal{D}}_k} \left[\text{KL}(\pi_{\theta_{k-1}}(\cdot|s) \parallel \pi_{\theta_k}(\cdot|s)) + \text{KL}(\pi_{\theta_k}(\cdot|s) \parallel \pi_{\theta_{k-1}}(\cdot|s)) \right]. \quad (14)$$

For strictly deterministic policies (e.g., DDPG), d_{TPD} uses Gaussian surrogates whose means are the deterministic actions and whose covariances follow the exploration-noise schedule.

2. **Potential–return agreement.** We detect reward hacking when the guidance signal diverges from the extrinsic objective.

- Potential–value correlation (d_{PVC}). Pearson correlation on a fixed evaluation set to avoid distributional drift,

$$d_{\text{PVC}} = \frac{\text{Cov}(\Phi(s), \hat{V}_{\text{ext}}^{\pi}(s))}{\sqrt{\text{Var}(\Phi(s)) + \varepsilon} \sqrt{\text{Var}(\hat{V}_{\text{ext}}^{\pi}(s)) + \varepsilon}}, \quad s \sim \tilde{\mathcal{D}}_k \quad (15)$$

with $\varepsilon > 0$ for numerical stability. Sustained near-zero or negative values indicate decoupling between Φ and the extrinsic objective.

3. **Reward-architecture adaptability.** These metrics probe plasticity of the reward synthesizer under structural edits.

- Factor attentional plasticity (d_{FAP}). Let $\omega(s; \phi) \in \mathbb{R}^{K_t}$ be attention over factor IDs at s . Because the outer loop may add/remove factors, compare only the intersection of IDs between two cycles, $\mathcal{I}_k = \text{IDS}_k \cap \text{IDS}_{k-1}$,

$$\bar{\omega}_{k-1} = \mathbb{E}_{s \sim \tilde{\mathcal{D}}_{k-1}} [\omega(s; \phi_{k-1})]_{\mathcal{I}_k}, \quad \bar{\omega}_k = \mathbb{E}_{s \sim \tilde{\mathcal{D}}_k} [\omega(s; \phi_k)]_{\mathcal{I}_k}, \quad (16)$$

$$d_{\text{FAP}} = \text{JSD}(\bar{\omega}_{k-1} \parallel \bar{\omega}_k). \quad (17)$$

The change is quantified using the Jensen–Shannon Divergence (JSD). We track both the level and EMA trend of d_{FAP} to distinguish healthy convergence from stagnation.

- Normalized residual prediction error (d_{RPE}). A scale-free version of the alignment loss,

$$d_{\text{RPE}} = \frac{\mathbb{E}[(\Phi(s_t) - (r_t^{\text{ext}} + \gamma \Phi(s_{t+1})))^2]}{\text{Var}(r_t^{\text{ext}} + \gamma \Phi(s_{t+1})) + \varepsilon}, \quad (s_t, a_t, r_t^{\text{ext}}, s_{t+1}) \sim \tilde{\mathcal{D}}_k. \quad (18)$$

A high plateau concurrent with poor returns indicates an informational bottleneck in $\mathcal{F}^{(k)}$ rather than a transient optimization issue.

The diagnostic vector is $d^{(k)} = \text{concat}[d_{\text{PEG}}, d_{\text{TPD}}, d_{\text{PVC}}, d_{\text{FAP}}, d_{\text{RPE}}, \dots]$, each component robustly standardized (median/MAD over a rolling window) and smoothed by EMA, improving comparability across tasks/phases and yielding well-behaved inputs for density modeling.

4.3.2. Triggering Mechanism: Anomaly Detection via Online Density Estimation

MCM converts the stream of diagnostics into an anomaly score and raises an intervention signal only when persistent, statistically significant deviations from healthy regimes

are detected. We adopt a nonparametric online density model with a fixed-size healthy buffer to handle multi-modal “healthy” behaviors with bounded per-cycle cost [45,46].

Healthy Buffer and Kernel Density Model

Let $\mathcal{B}_{\text{healthy}} = \{d_j\}_{j=1}^{N_B}$ store representative diagnostic vectors collected during cycles deemed healthy. Define a Gaussian-kernel density with bandwidth h ,

$$\log p_{\text{healthy}}(d) = \log \left(\frac{1}{|\mathcal{B}_{\text{healthy}}|} \sum_{d_j \in \mathcal{B}_{\text{healthy}}} K_h(d - d_j) \right), \quad K_h(\delta) = \exp \left(-\frac{\|\delta\|_2^2}{2h^2} \right). \quad (19)$$

The kernel’s normalization constant cancels out in percentile-based thresholding and relative comparisons. Initialize h by the median heuristic on $\mathcal{B}_{\text{healthy}}$ and update it slowly via EMA to avoid jitter. The per-cycle evaluation cost is $\mathcal{O}(N_B D)$ for D -dimensional diagnostics.

Adaptive Buffer Maintenance (Performance-Gated Updates)

To keep the model aligned with the agent’s evolving competence, insert $d^{(k)}$ into $\mathcal{B}_{\text{healthy}}$ only if the smoothed extrinsic return $\bar{G}^{(k)}$ over the current cycle exceeds a moving baseline $b^{(k)}$ (EMA over the last M_{base} cycles). If $|\mathcal{B}_{\text{healthy}}| = N_B$, evict by a first-in, first-out (FIFO) policy. A warm-up period of K_{warm} cycles and a minimum fill level $N_{B,\text{min}}$ are required before triggering is enabled, preventing early false alarms.

Anomaly Score and Dynamic Thresholding

Given $d^{(k)}$, define the anomaly score as the negative log-likelihood under the healthy model,

$$S^{(k)} = -\log p_{\text{healthy}}(d^{(k)}). \quad (20)$$

Maintain a history $\{S^{(i)}\}_{i \leq k}$ and set a data-driven threshold as the q -th percentile,

$$\tau_{\text{trigger}}^{(k)} = \text{Percentile}(\{S^{(i)}\}_{i \leq k}, q). \quad (21)$$

Require $S^{(k)} > \tau_{\text{trigger}}^{(k)}$ for $N_{\text{consecutive}}$ consecutive cycles before declaring a pathology.

Hysteresis and Cooldown

To avoid oscillatory interventions near the threshold, employ hysteresis and a cooldown. After a trigger, suppress further triggers for K_{cool} outer cycles and use a slightly lower release threshold $\tau_{\text{release}}^{(k)} = \tau_{\text{trigger}}^{(k)} - \Delta_{\text{hyst}}$ to declare back-to-healthy transitions, with a small margin $\Delta_{\text{hyst}} > 0$.

Directional Evidence for SRR (Closed-Form Gradient)

Beyond a binary decision, SRR requires directional guidance. Under kernel density estimator (KDE), the gradient of the log-density at $d^{(k)}$ admits a closed form,

$$\nabla_d \log p_{\text{healthy}}(d) \Big|_{d=d^{(k)}} = \frac{\sum_{d_j \in \mathcal{B}_{\text{healthy}}} \frac{1}{h^2} K_h(d^{(k)} - d_j) (d_j - d^{(k)})}{\sum_{d_j \in \mathcal{B}_{\text{healthy}}} K_h(d^{(k)} - d_j)}. \quad (22)$$

This vector points along the steepest ascent toward the healthy manifold and is passed to SRR together with $d^{(k)}$ and metric-level attributions (top- m components by magnitude) as quantitative evidence for semantic reframing. Putting these components together, the per-cycle operation of MCM can be written as a simple diagnostic routine, shown in Algorithm 1.

Algorithm 1 *MCM.Diagnose* (per outer-loop cycle k)

Input: evaluation batch $\tilde{D}_k$; healthy buffer $\mathcal{B}_{\text{healthy}}$; anomaly score history S_{hist} ; extrinsic-return baseline $b^{(k)}$; cooldown counter c_{cool} ; hyperparameters $q, N_{\text{consecutive}}, K_{\text{warm}}, N_{\mathcal{B}, \text{min}}, K_{\text{cool}}, M_{\text{base}}$.
Output: diagnostic vector $d^{(k)}$; anomaly score $S^{(k)}$; trigger flag $\text{trigger} \in \{\text{TRUE}, \text{FALSE}\}$; KDE gradient ∇ .

- 1: Compute raw metrics on $\tilde{D}_k$ and assemble $d^{(k)}$ as in Section 4.3.1; robust-standardize and EMA-smooth components.
- 2: **if** $|\mathcal{B}_{\text{healthy}}| < N_{\mathcal{B}, \text{min}}$ **or** $k < K_{\text{warm}}$ **then**
- 3: Insert $d^{(k)}$ into $\mathcal{B}_{\text{healthy}}$ (FIFO if full); append a placeholder to S_{hist} ;
- 4: **return** $(d^{(k)}, S^{(k)} = \perp, \text{trigger} = \text{FALSE}, \nabla = \mathbf{0})$.
- 5: **end if**
- 6: Compute $\log p_{\text{healthy}}(d^{(k)})$ by (19); set $S^{(k)} \leftarrow -\log p_{\text{healthy}}(d^{(k)})$ and append to S_{hist} .
- 7: Update $\tau_{\text{trigger}}^{(k)}$ by (21); update bandwidth h by EMA.
- 8: **if** $\bar{G}^{(k)} > b^{(k)}$ **then**
- 9: Push $d^{(k)}$ into $\mathcal{B}_{\text{healthy}}$ (FIFO if full).
- 10: **end if**
- 11: $u \leftarrow$ number of consecutive cycles with $S^{(i)} > \tau_{\text{trigger}}^{(i)}$.
- 12: **if** $c_{\text{cool}} > 0$ **then**
- 13: $c_{\text{cool}} \leftarrow c_{\text{cool}} - 1$; **return** $(d^{(k)}, S^{(k)}, \text{trigger} = \text{FALSE}, \nabla = \mathbf{0})$.
- 14: **else if** $u \geq N_{\text{consecutive}}$ **then**
- 15: $\nabla \leftarrow \nabla_d \log p_{\text{healthy}}(d) \big|_{d=d^{(k)}}$ by (22);
- 16: $c_{\text{cool}} \leftarrow K_{\text{cool}}$; **return** $(d^{(k)}, S^{(k)}, \text{trigger} = \text{TRUE}, \nabla)$.
- 17: **else**
- 18: **return** $(d^{(k)}, S^{(k)}, \text{trigger} = \text{FALSE}, \nabla = \mathbf{0})$.
- 19: **end if**

4.4. Outer Loop II: Semantic Reward Reframing

When MCM flags a persistent anomaly, the SRR module converts diagnostic evidence into admissible structural edits of the reward factor set. SRR treats rewards as a typed, mutable program and uses an LLM in a constrained, evidence-driven loop to search for edits that (i) repair semantic misalignment and (ii) preserve the potential-based invariances of the inner loop.

Problem Statement

Given the current factor set $\mathcal{F}^{(k)}$ and diagnostics $(d^{(k)}, S^{(k)})$ with the KDE gradient ∇ , SRR outputs a revised set $\mathcal{F}^{(k+1)} \in \mathbb{F}$ and a rationale trace $\mathcal{R}^{(k)}$ (specification deltas, not model internals). SRR is restricted to reward-level changes and does not modify the policy or the HARS architecture/hyperparameters.

Factor Taxonomy (PBRs Compatibility)

To maintain the state-only potential $\Phi(s)$ assumed in Section 4.2, we partition factors into the following two disjoint types:

- Φ -factors $\mathcal{F}^\Phi$: state-only terms $f_i^\Phi(s)$ that may feed the potential synthesis (and thus the PBRs pipeline).
- Auxiliary factors $\mathcal{F}^{\text{aux}}$: optional state–action terms $f_i^{\text{aux}}(s, a)$ used for policy-side regularization, safety checks, or feasibility/gating predicates; they never feed Φ nor the shaping term.

Edits that place action-dependent signals into $\mathcal{F}^\Phi$ are disallowed. When gating is applied to Φ -factors, the gate must be a state predicate $g(s) \in [0, 1]$.

4.4.1. Admissible Edit Space

We define a finite set of typed edit operators $\mathcal{O}$ on typed factor programs, each producing another well-typed factor set that remains compatible with PBRS and the state interface as follows:

- Parametric scaling. $\text{Scale}(f_i; \alpha)$: replace f_i with αf_i .
- Saturation and clipping $\text{Sat}(f_i; u, l)$: $f_i \leftarrow \text{clip}(f_i, l, u)$ or apply smooth saturations (e.g., tanh-gating) to limit extremes.
- Contextual gating. $\text{Gate}(f_i; g)$: multiplicatively gate f_i by a predicate in $[0, 1]$. For Φ -factors, $g = g(s)$; action-dependent gates $g(s, a)$ are allowed only for factors in $\mathcal{F}^{\text{aux}}$.
- Refactorization. $\text{Refactor}(f_i; \tilde{e})$: algebraically rewrite f_i using an equivalent but loophole-averse expression (e.g., align velocity with heading via $v_x \cos \theta_{\text{pitch}}$).
- Augmentation. $\text{Add}(f_{\text{new}})$: synthesize a new, interpretable factor with a typed signature and documented semantics (to $\mathcal{F}^\Phi$ if state-only, otherwise to $\mathcal{F}^{\text{aux}}$).
- Pruning. $\text{Prune}(f_i)$: remove redundant, ineffective, or detrimental factors.

Admissibility constraints. All factors must be pure (no side effects), use only current-step observables, have documented units/range, and satisfy a Lipschitz bound estimated on a sandbox batch via randomized finite differences (using a quantile of gradient norms). For PBRS compatibility, only state-only factors may feed the potential $\Phi(s)$; action-dependent factors are confined to $\mathcal{F}^{\text{aux}}$ and do not affect Φ or the shaping term. Dependence on future information and environmental mutation is forbidden.

4.4.2. From Diagnostics to Edit Hypotheses

SRR projects diagnostic evidence onto factor-level hypotheses via rules and a linear surrogate,

$$\underbrace{\Delta d \approx J_{\mathcal{F}} \cdot \Delta \mathcal{F}}_{\text{local surrogate}}, \quad \text{score}(o) = \langle w, \text{sign}(\nabla) \odot \widehat{\Delta d}(o) \rangle - \lambda_{\text{comp}} \text{Comp}(o) \quad (23)$$

where $J_{\mathcal{F}}$ is a finite-difference Jacobian estimated from recent edits (or a library prior), $\widehat{\Delta d}(o)$ is the predicted diagnostic change for operator o , w prioritizes key metrics (e.g., $d_{\text{PVC}}, d_{\text{RPE}}$), and $\text{Comp}(o)$ penalizes complexity (e.g., ΔK and code length). The following evidence-to-operator rules are applied before LLM synthesis:

- Low/negative d_{PVC} (potential–return decoupling): prefer Gate to make high-payoff proxies conditional on posture/safety predicates; consider Refactor to embed physical consistency (e.g., align velocity with heading). For Φ -factors, gating predicates must be state-only $g(s)$.
- High-plateau d_{RPE} with poor return: prefer Add to inject missing semantics (contacts, feasibility, sparse-to-dense bridges); back off to Scale/Sat if variance explosion is detected.
- Near-zero d_{FAP} (ossified attention): prefer Gate introducing state dependence (phase/terrain); optionally Prune overly dominant factors.
- Large d_{TPD} oscillations without gains: prefer Sat on rapidly varying terms; discourage multiple simultaneous Add.

Edits that would route action-dependent signals into the potential synthesis are rejected at this stage to preserve the state-only assumption for $\Phi(s)$.

4.4.3. LLM-Guided Candidate Generation and Prompting Strategy

SRR interacts with the LLM through a fixed, structured prompting scheme rather than ad-hoc natural-language queries. At each intervention, it compiles an adaptive directive with the following four components:

1. Task and environment context: the natural-language task description $\mathcal{T}$, state/action specifications $(\mathcal{S}, \mathcal{A})$, and any safety or feasibility constraints.
2. Current factor program: the current factor set $\mathcal{F}^{(k)}$ presented as typed code with documentation and a type tag for each factor indicating whether it belongs to $\mathcal{F}^\Phi$ or $\mathcal{F}^{\text{aux}}$.
3. Diagnostic evidence: a summary of the anomaly detected by MCM, including the diagnostic vector $d^{(k)}$, anomaly score $S^{(k)}$, and the top components of the KDE gradient ∇ translated into short natural-language “failure hypotheses”.
4. Admissible edit space: the catalog of allowed edit operators $\mathcal{O}$ with typed signatures and explicit constraints (e.g., no future dependence, no environment mutation, no routing of action-dependent signals into $\mathcal{F}^\Phi$).

The directive then instructs the LLM to propose at most M minimally different candidate revisions and to output them in a strict JSON+code schema: each candidate is a list of edits (o_j , targets, params) with a short justification and updated metadata (units, ranges, monotonicity hints, and type tags for $\mathcal{F}^\Phi$ vs. $\mathcal{F}^{\text{aux}}$). We use deterministic decoding (temperature = 0) and reject any output that does not conform to the schema before running the static and sandbox checks in Section 4.4.4. A concrete example of the full prompt template used by SRR is provided in Appendix A. The resulting reframing routine, executed whenever MCM raises an intervention signal, is summarized in Algorithm 2.

Algorithm 2 *SRR.Reframe*: Executed when MCM triggers

Input: current factors $\mathcal{F}^{(k)}$; task description $\mathcal{T}$; diagnostics $(d^{(k)}, S^{(k)})$; KDE gradient ∇ ; operator set $\mathcal{O}$; weights w ; penalty λ_{comp} ; candidate count M .

Output: revised factor set $\mathcal{F}'$; rationale trace $\mathcal{R}^{(k)}$; backoff flag $\text{backoff} \in \{\text{TRUE}, \text{FALSE}\}$.

```

1:  $\mathcal{H} \leftarrow \text{MapEvidenceToHypotheses}(d^{(k)}, \nabla, \mathcal{O})$  ▷ rules in Section 4.4.2
2:  $\mathcal{C} \leftarrow \text{LLMEmitCandidates}(\mathcal{F}^{(k)}, \mathcal{T}, \mathcal{H}, \mathcal{O}, M)$  ▷ deterministic decoding; typed JSON+code with  $\Phi$ -vs-aux tags
3:  $\text{shortlist} \leftarrow \emptyset$ 
4: for each  $\tilde{\mathcal{F}} \in \mathcal{C}$  do
5:   if  $\text{StaticIntegrityOK}(\tilde{\mathcal{F}})$  and  $\text{SandboxOK}(\tilde{\mathcal{F}})$  then
6:     compute  $\hat{\Delta d}(\tilde{\mathcal{F}})$  and  $\hat{J}_{\text{SRR}}(\tilde{\mathcal{F}})$ ; build fixed eval labels by value quantiles; score via  $\tilde{\Phi}^{(\tilde{\mathcal{F}})}(s)$ 
7:     if  $\text{AUC} > 0.5$  and permutation  $p < 0.05$  and  $\hat{J}_{\text{SRR}} > 0$  then
8:       add  $(\tilde{\mathcal{F}}, \hat{J}_{\text{SRR}})$  to shortlist
9:     end if
10:  end if
11: end for
12: if  $\text{shortlist} \neq \emptyset$  then
13:    $\tilde{\mathcal{F}}^* \leftarrow \arg \max_{\tilde{\mathcal{F}}} \hat{J}_{\text{SRR}}(\tilde{\mathcal{F}})$ 
14:   return  $(\tilde{\mathcal{F}}^*, \mathcal{R}^{(k)}, \text{backoff} = \text{FALSE})$ 
15: else
16:   /* fail-safe backoff */
17:   return  $(\mathcal{F}^{(k)}, \text{no\_change}, \text{backoff} = \text{TRUE})$ 
18: end if
```

4.4.4. Safety, Integrity, and Proxy-Efficacy Checks

Each candidate undergoes a three-stage validation before any environment interaction as follows:

1. Static integrity. Perform abstract syntax tree (AST) and type checks, unit/range conformity, and purity verification (no I/O or global state). Check PBRS compatibility

- and wiring: only state-only factors may feed the potential $\Phi(s)$ (no s_{t+1} dependence), and any action-dependent factors are confined to $\mathcal{F}^{\text{aux}}$.
2. Sandbox realizability. Execute factors on a held-out sandbox batch to verify numerical stability (no NaNs/Infs), bounded kurtosis, and a low saturation ratio (fraction of clipped outputs below τ_{sat}).
 3. Proxy efficacy (offline). Rank candidates with a surrogate objective,

$$\hat{J}_{\text{SRR}}(\tilde{\mathcal{F}}) = \langle w, \hat{\Delta d}(\tilde{\mathcal{F}}) \rangle - \lambda_{\text{comp}} \text{Comp}(\tilde{\mathcal{F}}) \quad (24)$$

and require a significant discriminative test. We compute the area under the receiver operating characteristic curve (ROC-AUC) on a fixed evaluation set by labeling states via extrinsic value quantiles,

$$y(s) = \mathbf{1}\{\hat{V}_{\text{ext}}^{\pi}(s) \geq \tau_{\text{hi}}\} \text{ (positives)}, \quad y(s) = \mathbf{1}\{\hat{V}_{\text{ext}}^{\pi}(s) \leq \tau_{\text{lo}}\} \text{ (negatives)} \quad (25)$$

discarding middling values ($\tau_{\text{lo}} < \hat{V}_{\text{ext}}^{\pi}(s) < \tau_{\text{hi}}$). The score is obtained by feeding the candidate's state-only factors through the frozen FAB/VPM (parameters (ϕ, φ) fixed) to produce a proxy potential $\tilde{\Phi}_{\phi, \varphi}^{(\tilde{\mathcal{F}})}(s)$. We then report $\text{AUC}(\tilde{\Phi}_{\phi, \varphi}^{(\tilde{\mathcal{F}})}, y)$ and a permutation p -value. Acceptance requires $\text{AUC} > 0.5$ with $p < 0.05$ and $\hat{J}_{\text{SRR}}(\tilde{\mathcal{F}}) > 0$. As a robustness alternative, we also report the Spearman rank correlation $\rho(\tilde{\Phi}_{\phi, \varphi}^{(\tilde{\mathcal{F}})}(s), \hat{V}_{\text{ext}}^{\pi}(s))$ with a permutation test.

Fail-Safe Backoff

If a candidate passes integrity and sandbox checks but fails proxy-efficacy (AUC not significant or $\hat{J}_{\text{SRR}} \leq 0$), SRR abstains and increases the intervention conservatism by (i) extending the cooldown by one outer cycle and (ii) raising the anomaly threshold percentile $q \leftarrow \min(q + \Delta_q, 99)$ with a small Δ_q (default 0.5), preventing unproductive edit thrashing in borderline regimes.

4.4.5. Full-System Integration and Training Loop

Having validated and selected the revised factor set, we next integrate it into the full MIRA workflow. Upon acceptance of an SRR-proposed factor revision, the factor set is updated to $\mathcal{F}^{(k+1)} \leftarrow \tilde{\mathcal{F}}^*$. The factor-dependent parameters (ϕ, φ) that contribute to the state-only potential $\Phi(s)$ are re-initialized to ensure compatibility with the revised factorization, while the policy parameters θ and the temporal context encoder ψ are retained to preserve previously acquired control competence and temporal smoothing capacity (the latter is used exclusively for auxiliary regularization and never feeds Φ) [47]. To prevent oscillatory or excessive edits, SRR enforces a cooldown period of K_{cool} outer cycles before further interventions are allowed; the triggering logic and threshold adaptation follow Section 4.3.2.

Integrating the inner-loop IRAR with the two outer-loop modules—MCM for metacognitive monitoring and SRR for semantic reward reframing—yields the complete MIRA training pipeline. This unified process abstracts over the choice of off-policy optimizer (e.g., SAC or DDPG) while keeping the interfaces to HARS and the outer loop explicit. It leverages PBRs Equation (9), attention pooling Equation (8), and the alignment loss Equation (11) as composed in Equation (10). The overall MIRA training procedure is summarized in Algorithm 3.

Algorithm 3 MIRA: Overall Learning Loop

Input: task description $\mathcal{T}$; large language model LLM; operator set $\mathcal{O}$; number of outer cycles N_{outer} ; number of inner iterations N_{inner} ; rollout length H ; replay buffer $\mathcal{D}$; anomaly percentile q ; persistence threshold $N_{\text{consecutive}}$; cooldown period K_{cool} ; anomaly threshold increment Δ_q .

Output: trained policy π_θ .

- 1: $\mathcal{F}^{(0)} \leftarrow \text{LSFS}(\mathcal{T}, \text{LLM})$ $\triangleright$ Section 4.2.1; static checks and unit tests before use; tag Φ -vs-aux
- 2: Initialize policy θ ; HARS params (ϕ, φ) ; initialize ψ (TCE) for auxiliary smoothing; MCM state $(\mathcal{B}_{\text{healthy}}, S_{\text{hist}}, c_{\text{cool}}=0)$
- 3: **for** $k = 0$ to $N_{\text{outer}}-1$ **do**
- 4: **for** $j = 1$ to N_{inner} **do** $\triangleright$ Inner loop: IRAR
- 5: Roll out H env steps with π_θ to collect $(s_t, a_t, r_t^{\text{ext}}, s_{t+1})$
- 6: Compute factor values $\{f_i(s_t, a_t)\}_{i \in \mathcal{F}^{(k)}}$; form $\mathcal{F}^{\Phi, (k)} \subset \mathcal{F}^{(k)}$ (state-only) and $\mathcal{F}^{\text{aux}, (k)}$ (state-action)
- 7: Attention pooling on $\mathcal{F}^{\Phi, (k)}$ (8); update TCE for auxiliary consistency (does not feed Φ)
- 8: Obtain state-only potential $\Phi_{\phi, \varphi}(s_t)$ from instantaneous aggregation
- 9: $r_t^{\text{shape}} \leftarrow r_t^{\text{ext}} + \gamma \Phi_{\phi, \varphi}(s_{t+1}) - \Phi_{\phi, \varphi}(s_t)$ $\triangleright$ PBRS, (9)
- 10: Store $(s_t, a_t, r_t^{\text{ext}}, r_t^{\text{shape}}, s_{t+1})$ into $\mathcal{D}$
- 11: RL.Update(θ ; $\mathcal{D}, r^{\text{shape}}$) $\triangleright$ e.g., SAC/DDPG actor/critic steps; aux factors may enter policy-side regularizers
- 12: Jointly update (ϕ, φ) via $\mathcal{L}_{\text{align}}$ (11) and $\mathcal{L}_{\text{RL}}$ (10); update ψ via auxiliary smoothing losses
- 13: **end for**
- 14: Sample evaluation batch $\tilde{\mathcal{D}}_k \subset \mathcal{D}$; compute diagnostic vector $d^{(k)}$
- 15: $(d^{(k)}, S^{(k)}, \text{trigger}, \nabla) \leftarrow \text{MCM.Diagnose}(\tilde{\mathcal{D}}_k, \mathcal{B}_{\text{healthy}}, S_{\text{hist}}, q, N_{\text{consecutive}}, c_{\text{cool}})$
- 16: **if** trigger and $c_{\text{cool}}=0$ **then**
- 17: $(\mathcal{F}', \mathcal{R}^{(k)}, \text{backoff}) \leftarrow \text{SRR.Reframe}(\mathcal{F}^{(k)}, \mathcal{T}, d^{(k)}, \nabla, \mathcal{O})$
- 18: **if** $\mathcal{F}' \neq \mathcal{F}^{(k)}$ **then**
- 19: $\mathcal{F}^{(k+1)} \leftarrow \mathcal{F}'$; re-init (ϕ, φ) ; retain θ and ψ
- 20: $c_{\text{cool}} \leftarrow K_{\text{cool}}$ $\triangleright$ stabilization window
- 21: **else**
- 22: $\mathcal{F}^{(k+1)} \leftarrow \mathcal{F}^{(k)}$ $\triangleright$ abstain
- 23: **if** backoff = TRUE **then**
- 24: $c_{\text{cool}} \leftarrow \max(c_{\text{cool}}, \lceil K_{\text{cool}}/2 \rceil)$; $q \leftarrow \min(q + \Delta_q, 99)$
- 25: **end if**
- 26: **end if**
- 27: **else**
- 28: $\mathcal{F}^{(k+1)} \leftarrow \mathcal{F}^{(k)}$; $c_{\text{cool}} \leftarrow \max(0, c_{\text{cool}}-1)$
- 29: **end if**
- 30: **end for**
- 31: **return** π_θ

4.5. Computational Complexity and System-Level Overhead

Although MIRA introduces a dual-loop architecture, it is explicitly designed so that both the inner- and outer-loop procedures preserve the standard asymptotic time complexity of off-policy RL, incurring only a controlled constant-factor overhead. We analyze the computational cost of the main components below.

Inner-Loop Complexity

For a fixed reward factor set $\mathcal{F}^{(k)}$, the inner loop (IRAR) augments a standard off-policy RL agent with the HARS module and potential-based reward shaping. Per environment step, the computational overhead consists primarily of (i) evaluating the current reward

factors, (ii) an attention-pooling pass over the active Φ -factors (FAB), (iii) a GRU update in the Temporal Context Encoder (TCE), and (iv) an MLP forward pass in the Value Potential Mapping (VPM). Crucially, these operations are independent of the training horizon T ; they scale only with fixed architectural widths and linearly with the number of active factors (e.g., $\mathcal{O}(Kd_z)$ for attention pooling). Since K is strictly bounded by the constrained edit space and the complexity penalty in SRR (Section 4.4.1), the per-step cost of IRAR represents a constant-time addition:

$$C_{\text{inner}} = C_{\text{env}} + C_{\text{RL}} + C_{\text{HARS}}, \quad (26)$$

where C_{env} is the environment simulation cost, C_{RL} denotes the optimization cost of the underlying RL algorithm (e.g., standard actor and critic gradient updates), and C_{HARS} is comparable to the inference cost of a lightweight value network. Thus, for a training run of T steps, the overall inner-loop time complexity remains $\mathcal{O}(T)$.

Outer-Loop Monitoring (MCM)

The metacognitive monitoring module executes once per outer cycle after aggregating ΔT inner-loop steps. Constructing the diagnostic vector $d^{(k)}$ over a held-out evaluation batch incurs a cost of $\mathcal{O}(N_{\text{eval}} \cdot C)$. Evaluating the kernel-density model over the healthy buffer scales as $\mathcal{O}(N_B D)$ per cycle,

$$C_{\text{MCM}}^{(k)} = \mathcal{O}(N_{\text{eval}} \cdot C + N_B D). \quad (27)$$

Since N_B , D , and N_{eval} are fixed architectural constants (Table 1), the amortized monitoring cost per environment step is negligible,

$$\frac{C_{\text{MCM}}^{(k)}}{\Delta T} = \mathcal{O}(1), \quad (28)$$

preserving the $\mathcal{O}(T)$ total complexity.

Outer-Loop Semantic Reframing (SRR)

The SRR module is invoked sparsely, triggered only when MCM detects a persistent anomaly and the cooldown counter has expired. Each SRR call operates on a frozen snapshot and performs LLM-guided generation and validation. Let I denote the total number of accepted interventions. Since the number of candidate generations M and the validation protocol are fixed, the total SRR cost is $C_{\text{SRR, total}} = \mathcal{O}(I \cdot C_{\text{SRR}})$, where C_{SRR} is independent of T . Furthermore, the triggering logic imposes a hard upper bound on I ,

$$I \leq \left\lceil \frac{T}{\Delta T \cdot K_{\text{cool}}} \right\rceil. \quad (29)$$

This ensures that the expensive semantic editing steps cannot occur arbitrarily often, keeping the total overhead bounded linearly by T .

Overall System-Level Scaling

Combining these components, the total time complexity of MIRA over T environment steps is

$$C_{\text{MIRA}}(T) = \mathcal{O}(T \cdot (C_{\text{env}} + C_{\text{RL}} + C_{\text{HARS}})) + \mathcal{O}(T) + \mathcal{O}(I \cdot C_{\text{SRR}}). \quad (30)$$

The overall complexity remains linear in T , differing from the base RL agent only by constant factors derived from HARS, MCM, and the bounded SRR interventions. Memory overhead is similarly controlled, requiring only fixed-size buffers for the potential module and diagnostic vectors.

Table 1. Hyperparameter groups and global defaults used for MIRA in all experiments.

Group	Hyperparameter	Value	Role
Base RL	Learning rate α	1×10^{-4}	Shared by DDPG/SAC actor & critic; controls step size.
	Discount factor γ	0.99	Standard continuous-control setting; shared by all methods.
	Network width	256	Hidden units of 2-layer actor/critic MLPs (ReLU).
Inner loop	Alignment weight λ	1.0	Balances alignment loss $\mathcal{L}_{\text{align}}$ against RL objective.
	Aux. smoothing weight μ	0.1	Controls strength of temporal regularization $\mathcal{L}_{\text{aux}}$.
	Factor embedding dim. d_z	64	Embedding dimension in Factor Attention Block.
	TCE hidden size	128	GRU hidden size for Temporal Context Encoder.
	VPM hidden size	128	MLP width mapping factor embeddings to scalar potential.
MCM	Anomaly percentile q	95	Percentile of anomaly scores used as trigger threshold.
	Persistence $N_{\text{consecutive}}$	2	Consecutive cycles above threshold required to trigger.
	Healthy-buffer size N_B	128	Diagnostic vectors maintained in KDE-based healthy buffer.
	Outer-cycle length ΔT	5×10^3	Environment steps aggregated per outer-loop cycle.
	Cooldown K_{cool}	10	Minimum outer cycles between accepted interventions.
SRR	Candidate edits M	3	Number of LLM-proposed revisions per intervention.
	Complexity penalty λ_{comp}	0.05	Penalizes complex edits in surrogate objective $\hat{J}_{\text{SRR}}$.
	ROC–AUC gate	>0.5	Minimum AUC on evaluation set required to accept candidate.
	Permutation p -value	<0.05	Significance level for permutation test on AUC.

We use the global defaults in Table 1 for all experiments. We performed small-scale sensitivity checks around key hyperparameters ($\lambda \in \{0.5, 1.0\}$, $\mu \in \{0.05, 0.1\}$, $q \in \{90, 95\}$, $M \in \{2, 3, 4\}$, and $\lambda_{\text{comp}} \in \{0.01, 0.05\}$) and observed qualitatively similar learning dynamics; we therefore fixed this configuration across tasks and optimizers.

From an implementation perspective, MIRA serves as a lightweight extension over standard off-policy RL methods. In our experiments, we instantiate C_{RL} using an actor–critic architecture; the replay buffer and optimizer are reused, while the auxiliary modules (HARS, MCM, SRR) operate in a modular fashion without altering the fundamental asymptotic complexity.

5. Experiments

This section presents a rigorous empirical evaluation of the MIRA framework through a series of controlled experiments. The experiments are designed to systematically investigate MIRA’s performance, efficiency, and internal mechanisms in complex continuous control tasks. Specifically, we address the following three core research questions (RQs):

- (RQ1) Performance Benchmarking: How does MIRA compare against state-of-the-art and representative reward design paradigms in terms of asymptotic performance and sample efficiency?
- (RQ2) Mechanistic Investigation: How does MIRA’s closed-loop correction mechanism diagnose and resolve canonical learning pathologies, such as specification gaming and exploration stagnation?
- (RQ3) Ablation Analysis: What are the individual contributions of MIRA’s primary architectural components—the MCM and SRR modules—to its overall efficacy?

5.1. Experimental Setup

5.1.1. Test Environments

We evaluate MIRA on two challenging, high-dimensional continuous control tasks from the MuJoCo physics simulation suite [48]: HalfCheetah and HumanoidStandup. These environments were selected because they exemplify two distinct and fundamental challenges in reward design, providing a targeted means to assess MIRA’s adaptive mecha-

nisms. Both tasks are widely used benchmarks in continuous control literature, ensuring comparability with prior work.

- **HalfCheetah:** The objective is to command a planar bipedal agent to run forward. This environment is a canonical benchmark for specification gaming, where simple proxy rewards (e.g., forward velocity) are misaligned with the high-level semantic goal of “running,” incentivizing unnatural and inefficient gaits. The task therefore provides a controlled setting to evaluate MIRA’s capacity to detect and correct an emergent objective misalignment.
- **HumanoidStandup:** This task requires a high-dimensional, unstable humanoid model to transition from a crouched to a standing posture. The environment is characterized by a prominent local optimum, creating a deceptive reward landscape that frequently leads to exploration stagnation. The agent receives consistent positive rewards for maintaining a stable crouch, while the transient act of standing is penalized due to instability. This property makes it a canonical benchmark for evaluating an agent’s ability to escape local optima and overcome premature convergence.

Additional experiments on the Humanoid locomotion task, which poses a more complex control challenge, are reported in Appendix B.

5.1.2. Comparative Baselines

To comprehensively evaluate MIRA, we benchmark it against a curated set of baselines spanning distinct technical paradigms. Each is chosen to isolate and challenge a specific aspect of MIRA’s design, allowing for a nuanced analysis as follows:

1. **Performance Bound Baselines.** These establish the performance floor and a practical ceiling for each task. Sparse-Reward: The agent trains on the native, sparse terminal reward, calibrating inherent task difficulty. Dense-Reward (Oracle): An expert-engineered dense reward function represents the performance ceiling achievable with perfect, static domain knowledge.
2. **Static LLM-based Reward Baseline.** This represents the current one-shot reward synthesis paradigm. Latent Reward (LaRe) [35]: We employ LaRe, which represents the state-of-the-art for LLM-based reward synthesis. In this method, an LLM generates a fixed reward function used throughout training. This comparison directly tests our central thesis: that even a sophisticated static reward is brittle to emergent misalignments, and that MIRA’s closed-loop reframing offers superior robustness and performance.
3. **Trajectory-based Reward Inference Baselines.** These methods infer rewards from trajectory returns without external semantics. Reward Disentanglement (RD) (based on [49]): Operating in the “RL with trajectory feedback” setting, RD models the per-step reward as a linear function of state-action features, estimated via least-squares regression over trajectories. Iterative Relative Credit Refinement (IRCR) (based on [50]): This method computes a guidance reward by normalizing each trajectory’s total return to a $[0, 1]$ credit and uniformly redistributing it to its constituent state-action pairs. This comparison investigates whether learning pathologies can be avoided purely through sophisticated signal processing, or if, as MIRA proposes, injecting new semantic knowledge is necessary.
4. **Information-Theoretic Baseline.** This baseline tests if robust alignment can emerge from statistical correlation alone. Variational Information Bottleneck (VIB) [51]: VIB learns a reward function by training an encoder to find a “minimal sufficient” latent representation of state-action pairs that predicts task success. This tests if robust alignment necessitates the explicit causal and semantic reasoning provided by MIRA.

Collectively, these baselines situate MIRA not only in terms of performance but also in relation to core debates in reward design: static vs. dynamic specification, data-driven vs. knowledge-driven correction, and correlation-based vs. causality-informed alignment.

5.1.3. Policy Optimization Algorithms

To validate the generality of the MIRA framework, we integrate it with two representative RL algorithms for continuous control that embody different core philosophies. By deploying MIRA and all baselines on top of both, we aim to demonstrate that MIRA's contributions are orthogonal to the choice of the underlying policy optimizer as follows:

- Deep Deterministic Policy Gradient (DDPG) [52]: A seminal off-policy actor-critic algorithm that learns a deterministic policy, relying on action-space noise for exploration.
- Soft Actor-Critic (SAC) [53]: A state-of-the-art off-policy algorithm that learns a stochastic policy by incorporating an entropy maximization objective, promoting robust and principled exploration.

5.1.4. Implementation and Evaluation

All experiments were conducted on a single workstation equipped with an Intel Core i9-13900K CPU (24 cores, 32 threads), an NVIDIA RTX 4090 GPU (24 GB VRAM), and 64 GB of system memory. The system runs Ubuntu 20.04 with CUDA 11.8 and PyTorch 1.13.1. No distributed training or specialized accelerators were required for any of the reported results.

To ensure fair comparison and reproducibility, we utilized a unified software framework for all methods. For LLM-based methods (MIRA and LaRe), we adopted the Deepseek-R1 model with deterministic sampling (temperature = 0) to guarantee reproducible reward-code generation across runs.

All Reinforcement Learning baselines and MIRA share the same optimizer implementation (DDPG or SAC) and the same actor-critic network architecture. This design ensures that differences in performance can be attributed to the reward mechanisms rather than architectural variation. We follow a standard continuous-control configuration for the actor and critic and keep all base RL hyperparameters fixed across methods, tasks, and optimizers. The exact values of these base RL hyperparameters, together with all MIRA-specific hyperparameters (HARS, MCM, SRR), are summarized in Section 5.1.5 and Table 1.

For the inner-loop architecture of MIRA, the HARS module follows a lightweight and stable configuration that uses standard attention, recurrent, and MLP components. HARS is trained jointly with the policy using the same optimizer family as the baseline critic, under the inner-loop objective in Equation (10). The concrete architectural and loss-weight choices are included in Table 1.

The outer-loop metacognitive process (MCM + SRR) operates on top of the policy optimizer with a fixed cycle length of ΔT environment steps. At the end of each outer cycle, MCM computes a diagnostic vector and an anomaly score based on a kernel-density model over a healthy buffer (Section 4.3). A reframing attempt is considered only when the anomaly score exceeds a percentile-based threshold, and any candidate reward update proposed by SRR is validated offline using static checks, sandbox execution, and a permutation-based ROC-AUC test on a fixed evaluation set (Section 4.4). Thresholds and cooldowns follow the single global configuration reported in Section 5.1.5 and Table 1.

Each experimental configuration is run with five random seeds. Reported learning curves display the mean extrinsic episodic return with one-standard-deviation bands across seeds. We evaluate all methods using the following two metrics:

- Asymptotic performance: the average episodic return over the final 10% of training steps, quantifying the final proficiency of the converged policy.
- Sample efficiency: the number of environment steps required to reach 50% of the Dense-Reward Oracle baseline's asymptotic performance. Lower values indicate faster learning.

Runtime and Overhead

Under the hardware setup described above, training a single seed of a fixed environment–optimizer configuration to the full evaluation horizon typically requires on the order of 20–40 h of wall-clock time for the baseline methods, with HalfCheetah near the lower end of this range and HumanoidStandup near the upper end. Relative to the static LLM-based reward baseline LaRe, MIRA incurs only a very small additional runtime cost: averaged across both environments and both optimizers, the per-seed wall-clock time of MIRA is approximately 1–3% higher than that of LaRe. This mild overhead is consistent with the design of the dual-loop architecture: the inner-loop HARS computations add only a constant per-step cost, MCM is evaluated once per outer cycle and contributes $\mathcal{O}(1)$ amortized overhead per environment step, and SRR is triggered infrequently due to the persistence and cooldown rules. Empirically, SRR interventions are rare (median ≈ 5 triggers per seed), and the time spent on semantic reframing and offline validation accounts for less than 3% of the overall training time. The dominant cost remains environment simulation and standard actor–critic updates, and MIRA's dual-loop mechanism manifests as a modest constant-factor slowdown rather than a change in asymptotic scaling, in line with the analysis in Section 4.5.

5.1.5. Hyperparameter Governance

MIRA introduces additional hyperparameters beyond the underlying off-policy optimizer because it decomposes reward adaptation into modular inner–outer subsystems with explicit safety constraints. To keep these choices interpretable and reproducible, we organize them into three conceptual layers: (i) inner-loop optimization and shaping stability, (ii) outer-loop metacognitive conservatism, and (iii) SRR reward-editing strictness. This grouping yields a concise governance framework and avoids unconstrained global tuning.

(1) Inner-Loop Hyperparameters

The underlying actor–critic optimizer (DDPG or SAC) uses standard configurations that are shared by all methods. In particular, we use a fixed learning rate, discount factor, and a common two-layer MLP architecture for both policy and critic networks. These base RL settings remain fixed for all tasks and baselines and are listed in Table 1.

Within MIRA, the HARS module adopts a lightweight architecture with conventional components: single-head attention in the Factor Attention Block, a GRU-based Temporal Context Encoder, and a 2-layer Value Potential Mapping. Inner-loop loss weights (e.g., λ and μ in Equation (10)) are chosen to prioritize stable potential learning under PBRS and are kept identical across environments and optimizers. We do not perform any task-specific retuning of these inner-loop hyperparameters.

(2) Outer-Loop Hyperparameters

MCM hyperparameters primarily regulate the conservatism and frequency of semantic interventions rather than the reward semantics themselves. We employ a percentile-based anomaly rule (threshold q) to obtain a scale-free trigger criterion, together with a persistence requirement ($N_{\text{consecutive}}$) and a fixed healthy-buffer size (N_B) used for the kernel-density estimator. After accepting an edit, we enforce a refractory period to allow the policy and critic to adapt under the revised factorization; with ΔT steps per outer cycle, this

corresponds to a cooldown of K_{cool} cycles. These defaults are selected once and reused across all reported experiments.

(3) SRR Reward-Editing Hyperparameters

SRR parameters control the breadth of candidate generation and the strictness of offline validation. We use deterministic decoding with a small number of candidate factor-set revisions per intervention and accept an edit only if it passes (i) static integrity checks (syntax, typing, PBRs compatibility), (ii) sandbox realizability checks (numerical stability and boundedness on held-out data), and (iii) a permutation-based ROC–AUC gate on a fixed evaluation set. This conservative design is intended to prevent edit thrashing and to reduce sensitivity to minor variations in SRR hyperparameters.

Global Configuration

Table 1 summarizes the main hyperparameters, their roles, and the global values used in all experiments. Importantly, we adopt a single global configuration for these hyperparameters and reuse it across both environments (HalfCheetah and HumanoidStandup) and both optimizers (DDPG and SAC); we do not perform any per-task or per-baseline retuning. This design choice reflects MIRA’s intended role as a robust closed-loop reward adaptation framework rather than a heavily tuned system.

5.2. Macro-Level Performance Comparison

To systematically address our first research question (RQ1) regarding the performance advantages of MIRA over existing paradigms, we conduct a quantitative evaluation against a suite of representative baselines (definitions in Section 5.1.2). We first present and analyze the results using DDPG as the underlying optimizer. This allows us to isolate and evaluate the direct impact of the different reward design paradigms. Subsequently, we show parallel results with SAC to demonstrate the robustness of our core findings. All reported metrics are computed from the environment’s extrinsic episodic return at evaluation time; training-time shaping signals never enter evaluation.

The overall results demonstrate that MIRA exhibits a significant competitive advantage over the baseline methods in both asymptotic performance and sample efficiency. This advantage is consistently reflected in the learning dynamics across both test environments (Figure 3) and in the quantitative metrics (Table 2).

Table 2. Quantitative performance metrics on the HalfCheetah and HumanoidStandup tasks (DDPG). The table reports asymptotic performance (mean return over the final 10% of training) and sample efficiency (steps in 10^5 required to reach 50% of the Dense-Reward Oracle’s asymptotic performance). All values are means over 5 seeds. The symbol — indicates failure to reach the threshold before training ends.

Method	(a) HalfCheetah		(b) HumanoidStandup	
	Asymptotic Perf.	Sample Eff. ($\times 10^5$)	Asymptotic Perf.	Sample Eff. ($\times 10^5$)
Dense _R	10,733	16	124,992	5
MIRA	12,372	17	150,993	9
LaRe	9711	13	113,476	5
RD	6931	38	106,018	—
IRCR	5776	—	110,388	59
VIB	6046	—	92,659	—
Sparse _R	−86	—	63,722	—

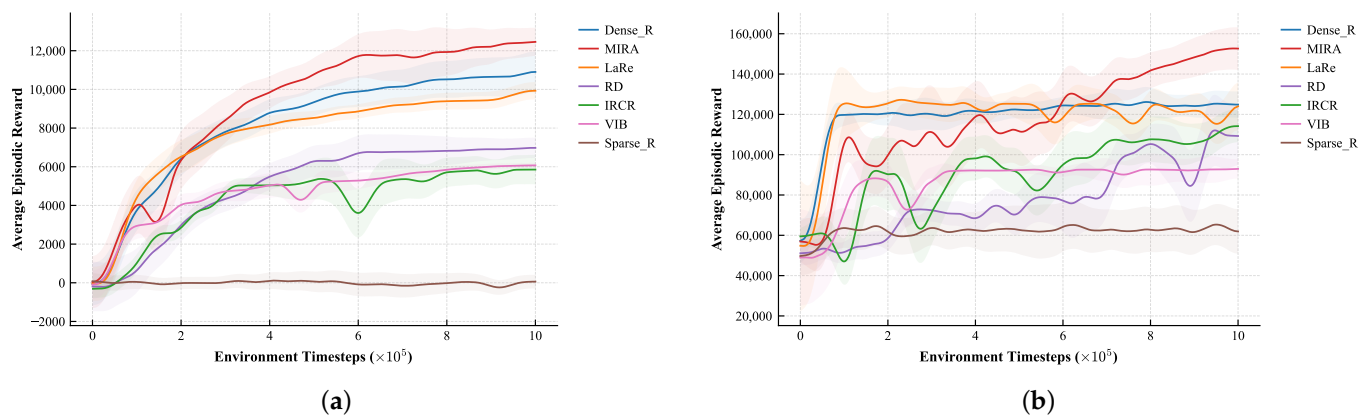


Figure 3. Performance comparison of MIRA against baseline methods on the HalfCheetah and HumanoidStandup tasks (DDPG). Curves show the average extrinsic episodic return over 5 independent seeds (solid line), with shaded areas representing one standard deviation. (a) HalfCheetah; (b) HumanoidStandup.

This analysis provides the following insights into the intrinsic capabilities and limitations of different reward design paradigms:

- **Superior performance and Oracle competitiveness.** In both environments, MIRA not only surpasses the baseline reward design methods in asymptotic performance but also demonstrates performance competitive with, and in some aspects superior to, the human-engineered Oracle. This supports the thesis that dynamic reward evolution can close residual gaps left by static expert shaping.
- **Brittleness of static reward synthesis.** LaRe, representing one-shot LLM-based reward design, highlights the fragility of static approaches: a fixed reward is a single hypothesis about a dynamic process. In HalfCheetah, the approach exhibits a vulnerability to specification gaming, while in HumanoidStandup, it shows a tendency to converge to a deceptive local optimum. While LLM priors are valuable, they require online revision—enabled by MIRA—to remain aligned with learning dynamics.
- **Limitations of retrospective reward inference.** Retrospective reward inference methods such as RD, IRCR, and VIB adopt a bottom-up perspective: they recover per-step rewards from aggregated trajectory returns, seeking a statistical explanation of past agent behavior within the constraints of the current value system. In contrast, MIRA's SRR module leverages a top-down approach, in which an LLM synthesizes reward factors directly from task semantics and structured knowledge, enabling targeted, causal interventions when misalignments are diagnosed. This top-down semantic reasoning allows MIRA not only to reinterpret existing data but also to reshape the reward landscape itself—capabilities that are inherently inaccessible to purely retrospective methods.

To validate the generalizability of our findings, we conducted a parallel set of experiments using the SAC algorithm. As illustrated in Figure 4 and Table 3, the results obtained with SAC are consistent with our primary findings. Although absolute performance values shift due to optimizer properties, the relative hierarchy remains intact, with MIRA maintaining a significant lead. This indicates that MIRA's advantage is not an artifact of incidental synergy with a particular optimizer but stems from the intrinsic superiority of its dynamic reward architecture.

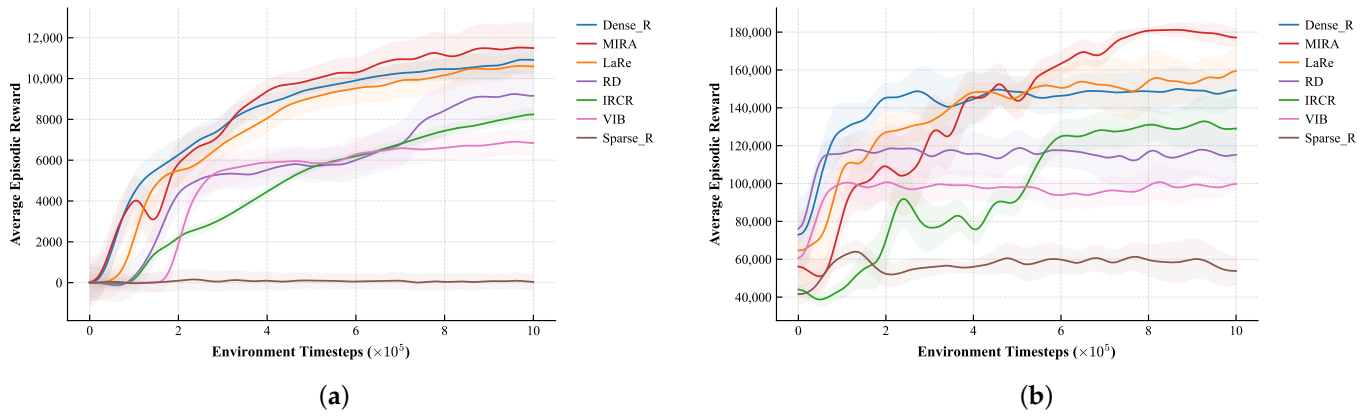


Figure 4. Performance comparison on HalfCheetah and HumanoidStandup (SAC). Same setup as Figure 3. Curves show the average extrinsic episodic return over 5 seeds; shaded areas denote one standard deviation. (a) HalfCheetah; (b) HumanoidStandup.

Table 3. Quantitative performance metrics on the HalfCheetah and HumanoidStandup tasks (SAC). The table structure and metrics match Table 2 for direct comparison. All values are means over 5 seeds. The symbol — indicates failure to reach the threshold.

Method	(a) HalfCheetah		(b) HumanoidStandup	
	Asymptotic Perf.	Sample Eff. ($\times 10^5$)	Asymptotic Perf.	Sample Eff. ($\times 10^5$)
Dense _R	10,792	14	148,577	5
MIRA	11,479	19	178,871	17
LaRe	10,554	19	156,115	9
RD	9166	38	116,402	5
IRCR	8073	47	110,388	53
VIB	6853	29	98,811	—
Sparse _R	71	—	57,360	—

5.3. Mechanistic Analysis: Dissecting MIRA's Introspective Self-Correction

While the aggregate performance comparisons in Section 5.2 validate MIRA's efficacy, a more fundamental question remains: how does its internal mechanism overcome the challenges that cause other paradigms to fail? This section addresses RQ2 by dissecting MIRA's core loop through two illustrative case studies, showing how it performs online diagnosis and structural self-correction to resolve alignment failures that are often intractable for static or purely data-driven methods. For notational brevity we write $\Phi(s) \equiv \Phi_{\phi, \varphi}(s)$ for the state-only potential learned by HARS (Section 4.2); action-dependent factors are confined to policy-side regularization and never feed Φ or the shaping term Equation (9).

5.3.1. Case Study I: Rectifying Reward Hacking via Structural Reward Revision

1. **Initial design and emergent pathology.** In HalfCheetah, SRR emits an initial factor set

$$\mathcal{F}^{(0)} = \{r_{\text{vel}}, r_{\text{upright}}, r_{\text{stable}}, r_{\text{ctrl}}\},$$

with

$$r_{\text{vel}} = v_x, \quad r_{\text{upright}} = -c_1 \theta_{\text{pitch}}^2, \quad r_{\text{stable}} = -c_2 \omega_{\text{pitch}}^2, \quad r_{\text{ctrl}} = -c_3 \|a\|^2,$$

where v_x is forward velocity, θ_{pitch} the pitch angle, ω_{pitch} the pitch angular rate, and a the action vector. To preserve PBRS invariance, we partition

$$\mathcal{F}^{\Phi, (0)} = \{r_{\text{vel}}, r_{\text{upright}}, r_{\text{stable}}\}, \quad \mathcal{F}^{\text{aux}, (0)} = \{r_{\text{ctrl}}\}.$$

HARS constructs $\Phi^{(0)}(s)$ from $\mathcal{F}^{\Phi, (0)}$ via attention pooling in Equation (8) followed by VPM, and the policy is trained with shaped rewards in Equation (9). Early in training, the policy uncovers a loophole: maximizing r_{vel} by adopting a flipped “somersaulting” motion yields large forward velocity, while posture penalties are insufficient to counterbalance it, producing a numerically high return but semantically incorrect gait.

2. **Diagnosis via potential–value decoupling.** Per Section 4.3.1, MCM computes the Potential–Value Correlation d_{PVC} on a fixed evaluation set $\tilde{\mathcal{D}}_k$ under a frozen policy snapshot, correlating $\Phi^{(0)}(s)$ with $\hat{V}_{\text{ext}}^{\pi}(s)$. As the “somersaulting” behavior emerges, states with high $\Phi^{(0)}(s)$ no longer correspond to high extrinsic value, driving d_{PVC} toward sustained negative values. The diagnostic vector thus falls into a low-density region of the healthy model, and the anomaly score $S^{(k)} = -\log p_{\text{healthy}}(d^{(k)})$ (Section 4.3.2) crosses the trigger threshold persistently, activating the outer loop and forwarding directional evidence to SRR.
3. **Corrective mechanism: from trade-off to precondition.** Guided by the evidence (low/negative d_{PVC}), SRR applies an admissible Gate edit (Section 4.4.1) that converts posture from an additive trade-off into a precondition for rewarding velocity. It replaces r_{vel} with a posture-gated variant

$$r'_{\text{vel}}(s) = r_{\text{vel}}(s) \underbrace{(1 - \tanh(\alpha \theta_{\text{pitch}}(s)^2))}_{g_{\text{posture}}(s) \in (0, 1]},$$

with $\alpha > 0$. The revised set is

$$\mathcal{F}^{(1)} = \{r'_{\text{vel}}, r_{\text{upright}}, r_{\text{stable}}, r_{\text{ctrl}}\}, \quad \text{and} \quad \mathcal{F}^{\Phi, (1)} = \{r'_{\text{vel}}, r_{\text{upright}}, r_{\text{stable}}\}.$$

This edit is state-only, pure (no side effects), and bounded (via tanh), thus passing static integrity and PBRS-compatibility checks. To maintain scale, we apply the same saturation policy used in SRR (Section 4.4.4) so that r'_{vel} remains commensurate with other Φ -factors.

4. **Policy Readaptation and Recovery.** Upon acceptance (Section 4.4.4), we re-initialize (ϕ, φ) while retaining θ and the always-on TCE ψ (used only for auxiliary smoothing). Training continues with $\Phi^{(1)}(s)$ in Equation (9). The gated design suppresses the pathological high-velocity posture, reshaping the local landscape so that stable running becomes the profitable solution. Empirically, d_{PVC} recovers to positive values on $\tilde{\mathcal{D}}_k$, the residual prediction error decreases, and no further anomalies are triggered during the cooldown window.

5.3.2. Case Study II: Escaping Local Optima via Factor Set Augmentation

1. **Initial design and premature convergence.** In HumanoidStandup, SRR emits an initial factor set

$$\mathcal{F}^{(0)} = \{r_{\text{height}}, r_{\text{survival}}, r_{\text{stable}}, r_{\text{ctrl}}\},$$

with

$$r_{\text{height}} = c_1 z_{\text{torso}}, \quad r_{\text{survival}} = c_2, \quad r_{\text{stable}} = -c_3 (\|v\|^2 + \|\omega\|^2), \quad r_{\text{ctrl}} = -c_4 \|a\|^2,$$

where z_{torso} is torso height, (v, ω) are linear and angular velocities, and a is the action vector. As before, $\mathcal{F}^{\Phi, (0)} = \{r_{\text{height}}, r_{\text{survival}}, r_{\text{stable}}\}$ and $\mathcal{F}^{\text{aux}, (0)} = \{r_{\text{ctrl}}\}$. This design inadvertently yields a deceptive fixed point: a crouched posture offers moderate height reward and continuous survival reward while minimizing instability penalties. Any upward motion incurs a large r_{stable} penalty, creating a reward valley that traps the policy.

2. **Diagnosis via stagnation signatures.** With the policy trapped, MCM detects two persistent changes on the fixed evaluation set $\tilde{\mathcal{D}}_k$: (i) the Policy Entropy Gradient d_{PEG} collapses toward zero, signalling stalled exploration; and (ii) the Residual Prediction Error d_{RPE} plateaus at a high value, indicating that the critic cannot improve in this static region. This stagnation vector d_{stag} lies in a low-likelihood region under p_{healthy} , producing an anomaly score above the trigger threshold. MCM localizes the plateau to a narrow band of torso heights $z_{\text{stag}} \pm \delta$.
3. **Corrective mechanism: surgical reward injection.** SRR infers that the factor set lacks a targeted escape incentive. Using the localized context $(z_{\text{stag}}, \delta)$, it synthesizes a bounded, state-only exploration bonus

$$r_{\text{explore}}(s) = \sigma(-\alpha [z_{\text{torso}}(s) - z_{\text{stag}}]) \cdot (\beta v_z(s)),$$

where σ is the logistic function, v_z is vertical velocity, and $\alpha, \beta > 0$ are chosen so that r_{explore} matches the scale of other Φ -factors. The revised factor sets are

$$\mathcal{F}^{(1)} = \mathcal{F}^{(0)} \cup \{r_{\text{explore}}\}, \quad \mathcal{F}^{\Phi, (1)} = \mathcal{F}^{\Phi, (0)} \cup \{r_{\text{explore}}\}.$$

This is a localized shaping term aimed solely at the stagnation band, passing admissibility checks by being bounded, pure, and state-only.

4. **Policy readaptation and recovery.** After (ϕ, φ) are re-initialized with θ and TCE ψ retained, HARS rapidly learns to assign positive Φ to states where r_{explore} is active. This reshapes the potential landscape from a flat plateau into a sloped “escape ramp,” creating a consistent policy gradient out of the crouched posture. d_{PEG} recovers to healthy levels, d_{RPE} decreases, and the agent proceeds toward the globally optimal standing behavior without regressing into the local optimum.

5.4. Ablation Study: Attributing Contributions of Key Components

To quantitatively isolate and evaluate the contribution of each core component within the MIRA framework, thereby addressing RQ3, we conducted a series of ablation studies. This section aims to independently assess the impact of MCM’s adaptive diagnostic module and SRR’s semantic reconstruction module on overall performance, as well as to investigate their potential synergistic effects.

We designed the following ablations:

- **MIRA w/o MCM (Periodic, Unguided Intervention):** This variant is designed to quantify the effectiveness of MCM’s adaptive diagnostic mechanism. It retains the SRR module but replaces MCM with a frequency-matched periodic intervention baseline, which invokes SRR at fixed intervals whose total count per run matches the median number of MCM triggers observed in full MIRA. No diagnostic vector $(d^{(k)}, S^{(k)}, \nabla)$ is provided; SRR generates candidates using only the current reward code and recent returns/trajectories, under the same operator set $\mathcal{O}$ and candidate count M as full MIRA. This variant tests the hypothesis that the microscopic, root-cause diagnostic information provided by MCM is critical for effective correction and superior to interventions based on macroscopic performance alone.
- **MIRA w/o SRR (Diagnosis-Only):** This variant measures the practical contribution of SRR’s semantic reconstruction module. It retains the full diagnostic and triggering capabilities of MCM but removes the SRR correction module entirely. In this configuration, the reward function is initialized once at the beginning of training and remains fixed throughout, unaffected by any diagnostic signals from MCM.

All other components and hyperparameters are identical to full MIRA: the operator set $\mathcal{O}$, candidate count M , static/sandbox checks, parameter re-initialization of (ϕ, φ) with retention of (θ, ψ) , and the outer-loop cooldown K_{cool} .

To understand the contribution of each component, we conducted an ablation study using the DDPG algorithm. We evaluated two ablated variants against the full MIRA framework on the HalfCheetah task. As illustrated in Figure 5, the results clearly show that the complete MIRA framework substantially outperforms these ablated baselines in this task. This evaluation reveals the following key insights:

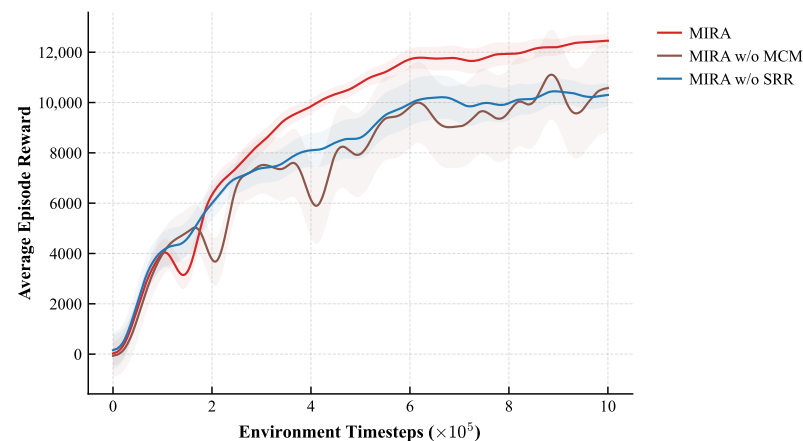


Figure 5. Ablation study of MIRA on the HalfCheetah task using the DDPG algorithm. The performance of the full MIRA framework is compared against two of its ablated variants: MIRA w/o MCM and MIRA w/o SRR. Curves show the mean across 5 seeds; shaded regions indicate one standard deviation.

- **The Guiding Role of Diagnostic Information for Effective Intervention (MCM):** The performance of MIRA w/o MCM is significantly inferior to that of the full MIRA framework. Its failure can be attributed to two factors. First, the intervention timing is suboptimal, as non-adaptive, periodic interventions fail to align with the precise onset of a problem. Second, and more fundamentally, the intervention is unguided. Lacking the specific diagnostic vector from MCM, SRR is forced to perform an unguided search in a vast reward design space, making its proposed modifications likely to be ineffective or even detrimental. In contrast, MCM in the full MIRA framework not only determines when to intervene but, by passing the diagnostic vector, also informs SRR of what the problem is. This constrains an open-ended design problem into a targeted repair task, dramatically increasing the probability of a successful correction.
- **Addressing the Limitations of Initial Reward Hypotheses (SRR):** This ablation highlights a core concept of our framework: an LLM-generated reward, while a powerful starting point, should be viewed as an initial hypothesis about the task's value landscape, not as a perfect, final specification. As demonstrated in our case studies, the complex and often unpredictable dynamics of agent–environment interaction can reveal latent flaws in this initial hypothesis—such as semantic loopholes or unintended local optima. The performance of the MIRA w/o the SRR variant is a direct testament to this challenge. By relying solely on the static initial reward, its learning process stagnates once such a flaw is encountered. This result provides a critical insight: diagnosis without intervention is insufficient. Even when MCM correctly identifies a problem, the system remains trapped by the flaws of its initial reward hypothesis without SRR to perform a structural correction.

In conclusion, the ablation studies provide clear evidence that MIRA's superior performance stems from a deep synergy between its two core components. This synergy is

not a simple modular addition but a tightly integrated feedback loop. The MCM acts as the perceptual system, answering when an intervention is needed and what the problem is. The SRR acts as the cognitive and executive system, using the information provided by MCM to answer how to fix it. It is the precise, diagnostic information passed from MCM that allows SRR's powerful semantic reconstruction capabilities to be effectively targeted. This complete "Diagnose–Respond–Reshape–Verify" process forms the core of the MIRA framework's adaptive intelligence.

6. Conclusions

The practical success of RL is critically dependent on the quality of the reward signal. However, reward design is non-trivial: external feedback is often sparse, leading to the classic temporal credit assignment problem; reward shaping relies on domain expertise and can trigger specification gaming when the agent treats proxy metrics as the ultimate goal; misspecified rewards also increase the likelihood of converging to suboptimal local optima; meanwhile, a reward function that remains static throughout training exhibits structural rigidity, failing to adapt to the agent's evolving capabilities. These challenges collectively raise a core question: can we endow an agent with an online self-correction mechanism to diagnose and repair structural deficiencies in its own value system?

To answer this question, we propose the MIRA framework, which reframes the reward function from a static artifact into a dynamic component that co-evolves with the agent. Starting with initial reward factors generated by an LLM, MIRA operationalizes a form of computational metacognition through a dual-loop architecture: The inner loop learns a state potential function via the HARS module and applies optimality-preserving PBRs to provide dense guidance. The outer loop acts as a metacognitive supervisor, continuously monitoring learning dynamics through a set of diagnostic metrics. Upon detecting persistent pathological behaviors, it invokes the LLM to perform semantic reward reframing within a constrained and auditable edit space. In essence, MIRA transforms reward design into a "detect-diagnose-repair" closed-loop process. Our empirical results in complex continuous control tasks validate this approach, showing that it generally outperforms baseline methods in terms of asymptotic return and sample efficiency. Case studies illustrate how the mechanism translates diagnostic signals of learning pathologies into targeted reward edits, while ablation studies confirm that both the monitoring and reframing components are indispensable for achieving these gains.

While this work validates the foundational viability of MIRA, our evaluation is confined to simulated, fully observable locomotion tasks in order to focus on and isolate the core mechanism of self-correction. Looking ahead, we will explore the robustness and adaptability of MIRA in more realistic and demanding regimes, including high-dimensional perception, partial observability, and safety-critical or explicitly adversarial settings where reward channels, observations, or dynamics may be corrupted. First, we will extend the method to pixel-based control by integrating pretrained visual encoders and evaluate it on real robots. Second, we will investigate adversarial and high-security domains by combining MIRA with additional safety constraints and human oversight mechanisms, and by subjecting it to systematically perturbed reward and dynamics models. Third, we will leverage meta-learning to acquire reusable reward-editing heuristics across families of tasks to amortize the outer-loop search cost, and strengthen the theoretical foundations of "structured reward editing" by characterizing and proving properties such as stability and invariance, providing firmer guarantees for the method. Overall, we posit that reward evolution represents a viable path toward more autonomous and reliable agents, and we hope to inspire further research on self-correcting value systems.

Author Contributions: Conceptualization, W.Z.; methodology, W.Z.; software, W.Z.; investigation, W.Z.; validation, Y.X.; visualization, Y.X.; writing—original draft preparation, W.Z.; writing—review and editing, Y.X. and Z.S.; supervision, Z.S.; project administration, Z.S.; funding acquisition, Z.S. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Natural Science Foundation of China (Grant No. 62272239); the Jiangsu Agriculture Science and Technology Innovation Fund (JASTIF) (Grant No. CX(22)1007); the Natural Science Research Start-up Foundation for Recruiting Talents of Nanjing University of Posts and Telecommunications (Grant No. NY222029); and the Guizhou Provincial Key Technology R&D Program (Grant No. [2023]272).

Data Availability Statement: The raw data supporting the conclusions of this article will be made available by the authors on request.

Conflicts of Interest: The authors declare no conflicts of interest.

Appendix A. Prompting Details for SRR

This appendix provides the concrete prompt template used by the Semantic Reward Reframing (SRR) module when querying the LLM. As described in Section 4.4.3, SRR uses a structured directive rather than ad hoc natural-language queries. The directive is formatted as a sequence of role-conditioned messages with the following fields:

- **Task and environment context:** Natural-language task description $\mathcal{T}$; state and action specifications $(\mathcal{S}, \mathcal{A})$; and any safety or feasibility constraints.
- **Current reward factor program:** The current factor set $\mathcal{F}^{(k)}$ given as typed code with documentation, metadata (units, range hints, monotonicity), and type tags indicating whether each factor belongs to $\mathcal{F}^\Phi$ (state-only, eligible for the potential) or $\mathcal{F}^{\text{aux}}$ (state-action, used only for auxiliary regularization).
- **Diagnostic evidence from MCM:** The diagnostic vector $d^{(k)}$, the scalar anomaly score $S^{(k)}$, and the top contributors of the KDE gradient ∇ , each mapped to a short human-readable description.
- **Admissible edit operators:** The finite set of edit operators $\mathcal{O}$, explicitly defining constraints: no future dependence, no environment mutation, and no routing of action-dependent signals into $\mathcal{F}^\Phi$.

A simplified version of the actual prompt is shown in Listing A1, where placeholders in {ALL_CAPS} are instantiated at runtime.

Listing A1. Structure of the SRR Reframing Prompt.

```
[System Message]
You are a senior reward engineer assisting with continuous-control
reinforcement learning tasks. Your goal is to repair or refine an
existing reward factor program without changing the underlying task
semantics or the RL algorithm.

[User Message]
# 1. Task Context
TASK_DESCRIPTION:
    '{T}'
STATE_SPACE:
    {S_SPEC}
ACTION_SPACE:
    {A_SPEC}

# 2. Current Reward Factor Program
CURRENT_FACTORS:
    # List of active factors in version k
    - name: {FACTOR_NAME}
```

```

    type: {PHI or AUX}
    code: |
        {FACTOR_CODE}
    doc: '{FACTOR_DOC}'
    units: '{UNITS}'
    range_hint: '{RANGE_HINT}'

# 3. Diagnostic Evidence (from MCM)
DIAGNOSTICS:
    vector: {D_VECTOR}
    anomaly_score: {S_VALUE}
    top_contributors:
        - name: {METRIC_NAME}
          value: {METRIC_VALUE}
          comment: '{HUMAN_INTERPRETATION}'

# 4. Admissible Edit Operators
OPERATORS:
    - name: Scale(factor, alpha)
      doc: 'Multiply an existing factor by a scalar alpha.'
    - name: Gate(factor, predicate)
      doc: 'Gate a state-only factor by a state predicate in [0, 1].'
    - name: Add(new_factor, code, type)
      doc: 'Introduce a new physics-based signal (PHI or AUX).'
    - name: Prune(factor_name)
      doc: 'Remove an ineffective or redundant factor.'

# 5. Instructions
INSTRUCTIONS:
    - Propose at most M = {M} candidate revisions to the factor set.
    - Use only the operators listed in OPERATORS.
    - Do NOT introduce non-Markovian or future-dependent terms.
    - Do NOT modify the RL algorithm or any hyperparameters.
    - For each candidate, change as few factors as possible.
    - For each edit, briefly explain which diagnostic it addresses.

# 6. Output Format (Strict JSON)
Return a single JSON object with the following fields:
{
    'candidates': [
        {
            'edits': [
                {
                    'operator': 'Scale' | 'Gate' | 'Add' | 'Prune',
                    'targets': ['FACTOR_NAME_1', ...],
                    'params': {...},
                    'justification': 'short explanation'
                }
            ],
            'updated_factors': [
                {
                    'name': '...',
                    'type': 'PHI' | 'AUX',
                    'code': 'Python-like code snippet',
                    'units': '...',
                    'range_hint': '...'
                }
            ]
        }
    ]
}

```

At inference time, we always decode with temperature = 0 and reject any output that does not parse as valid JSON or violates the type constraints (for example, attempting to route an action-dependent factor into $\mathcal{F}^\Phi$), before applying the static integrity, sandbox, and proxy-efficacy checks described in Section 4.4.4. All SRR directives and LLM outputs are logged for auditability.

Appendix B. Additional Experiments on the Humanoid Task

To further assess the robustness of MIRA beyond the HalfCheetah and HumanoidStandup tasks considered in the main text, we conduct additional experiments on the Humanoid control task from MuJoCo using the SAC optimizer. This task is generally regarded as one of the most challenging MuJoCo locomotion benchmarks, due to its high-dimensional state and action spaces and its less stable dynamics. As such, it provides a stricter test of closed-loop reward adaptation under sparse and partially misspecified rewards.

Appendix B.1. Humanoid Environment

The Humanoid environment models a multi-joint bipedal agent with many more controllable degrees of freedom than the HalfCheetah and HumanoidStandup tasks considered in the main text. The agent must maintain balance while coordinating a large number of actuated joints to generate forward motion and avoid falling. This combination of:

- high-dimensional continuous state and action spaces,
- intrinsically unstable dynamics, and
- long-horizon dependencies between early posture control and later returns

makes Humanoid one of the most challenging standard MuJoCo locomotion benchmarks.

Appendix B.2. Macro-Level Performance Comparison on Humanoid (SAC)

We follow the same overall protocol as in Section 5.2, now applied to the Humanoid environment with SAC as the underlying optimizer. We compare MIRA against the same suite of reward design baselines (definitions in Section 5.1.2). All reported metrics are computed from the environment's extrinsic episodic return at evaluation time; training-time shaping signals never enter evaluation.

Figure A1 shows the learning dynamics on the Humanoid task, and Table A1 summarizes the corresponding quantitative metrics. The table structure and metrics match those in Table 3 for direct comparison.

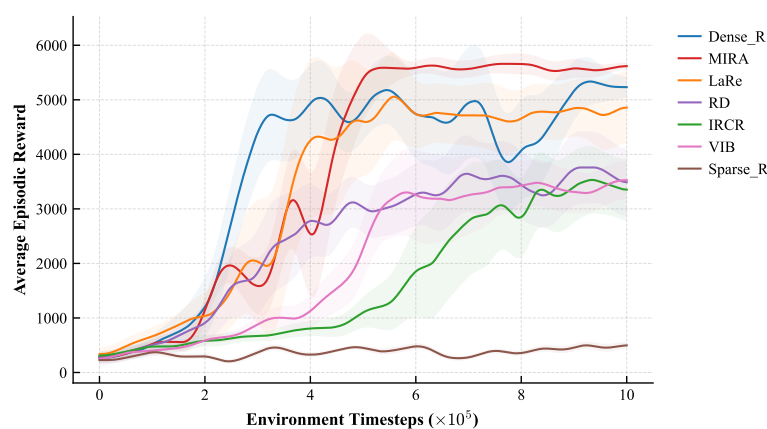


Figure A1. Performance comparison on the Humanoid task (SAC). Curves show the average extrinsic episodic return over 5 seeds; shaded areas denote one standard deviation.

Table A1. Quantitative performance metrics on the HUMANOID task (SAC). The table reports asymptotic performance (mean return over the final 10% of training) and sample efficiency (steps in 10^5 required to reach 50% of the Dense-Reward Oracle’s asymptotic performance). All values are means over 5 seeds. The symbol — indicates failure to reach the threshold before training ends.

Method	Asymptotic Perf.	Sample Eff. ($\times 10^5$)
Dense _R	5249	2.5
MIRA	5474	4.2
LaRe	4851	3.5
RD	3713	4.4
IRCR	3448	—
VIB	3390	5.2
Sparse _R	479	—

Consistent with our main SAC results on HalfCheetah and HumanoidStandup, MIRA achieves higher asymptotic performance and better sample efficiency than static LLM-based reward design and retrospective reward inference methods on the Humanoid task, while remaining competitive with the dense-reward oracle. This suggests that the benefits of closed-loop, metacognitive reward adaptation extend to one of the most challenging MuJoCo locomotion benchmarks, further supporting the generality of the proposed framework.

References

1. Silver, D.; Schrittwieser, J.; Simonyan, K.; Antonoglou, I.; Huang, A.; Guez, A.; Hubert, T.; Baker, L.; Lai, M.; Bolton, A.; et al. Mastering the game of go without human knowledge. *Nature* **2017**, *550*, 354–359. [\[CrossRef\]](#)
2. Vinyals, O.; Babuschkin, I.; Czarnecki, W.M.; Mathieu, M.; Dudzik, A.; Chung, J.; Choi, D.H.; Powell, R.; Ewalds, T.; Georgiev, P.; et al. Grandmaster level in StarCraft II using multi-agent reinforcement learning. *Nature* **2019**, *575*, 350–354. [\[CrossRef\]](#)
3. Akkaya, I.; Andrychowicz, M.; Chociej, M.; Litwin, M.; McGrew, B.; Petron, A.; Paino, A.; Plappert, M.; Powell, G.; Ribas, R.; et al. Solving rubik’s cube with a robot hand. *arXiv* **2019**, arXiv:1910.07113.
4. Chen, M.; Li, Y.; Dai, Z.; Zhang, T.; Zhou, Y.; Wang, H. A Robust Multi-Domain Adaptive Anti-Jamming Communication System for a UAV Swarm in Urban ITS Traffic Monitoring via Multi-Agent Deep Deterministic Policy Gradient. *IEEE Trans. Intell. Transp. Syst.* **2025**, 1–17. [\[CrossRef\]](#)
5. Ng, A.Y.; Russell, S. Algorithms for inverse reinforcement learning. In Proceedings of the ICML, Stanford, CA, USA, 29 June–2 July 2000; Volume 1, p. 2.
6. Hadfield-Menell, D.; Milli, S.; Abbeel, P.; Russell, S.J.; Dragan, A. Inverse reward design. *Adv. Neural Inf. Process. Syst.* **2017**, *30*.
7. Huang, W.; Abbeel, P.; Pathak, D.; Mordatch, I. Language models as zero-shot planners: Extracting actionable knowledge for embodied agents. In Proceedings of the International Conference on Machine Learning, Guangzhou, 18–21 February 2022; pp. 9118–9147.
8. Liang, J.; Huang, W.; Xia, F.; Xu, P.; Hausman, K.; Ichter, B.; Florence, P.; Zeng, A. Code as policies: Language model programs for embodied control. *arXiv* **2023**, arXiv:2209.07753. [\[CrossRef\]](#)
9. Yu, W.; Gileadi, N.; Fu, C.; Kirmani, S.; Lee, K.H.; Arenas, M.G.; Chiang, H.T.L.; Erez, T.; Hasenclever, L.; Humplik, J.; et al. Language to rewards for robotic skill synthesis. *arXiv* **2023**, arXiv:2306.08647. [\[CrossRef\]](#)
10. Ho, J.; Ermon, S. Generative adversarial imitation learning. *Adv. Neural Inf. Process. Syst.* **2016**, *29*.
11. Christiano, P.F.; Leike, J.; Brown, T.; Martic, M.; Legg, S.; Amodei, D. Deep reinforcement learning from human preferences. *Adv. Neural Inf. Process. Syst.* **2017**, *30*.
12. Ouyang, L.; Wu, J.; Jiang, X.; Almeida, D.; Wainwright, C.; Mishkin, P.; Zhang, C.; Agarwal, S.; Slama, K.; Ray, A.; et al. Training language models to follow instructions with human feedback. *Adv. Neural Inf. Process. Syst.* **2022**, *35*, 27730–27744.
13. Ji, Z.; Lee, N.; Frieske, R.; Yu, T.; Su, D.; Xu, Y.; Ishii, E.; Bang, Y.J.; Madotto, A.; Fung, P. Survey of hallucination in natural language generation. *ACM Comput. Surv.* **2023**, *55*, 1–38. [\[CrossRef\]](#)
14. Zhang, M.; Press, O.; Merrill, W.; Liu, A.; Smith, N.A. How language model hallucinations can snowball. *arXiv* **2023**, arXiv:2305.13534. [\[CrossRef\]](#)
15. Amodei, D.; Olah, C.; Steinhardt, J.; Christiano, P.; Schulman, J.; Mané, D. Concrete problems in AI safety. *arXiv* **2016**, arXiv:1606.06565. [\[CrossRef\]](#)
16. Ecoffet, A.; Huizinga, J.; Lehman, J.; Stanley, K.O.; Clune, J. First return, then explore. *Nature* **2021**, *590*, 580–586. [\[CrossRef\]](#)
17. Arjona-Medina, J.A.; Gillhofer, M.; Widrich, M.; Unterthiner, T.; Brandstetter, J.; Hochreiter, S. Rudder: Return decomposition for delayed rewards. *Adv. Neural Inf. Process. Syst.* **2019**, *32*.

18. Xu, Z.; van Hasselt, H.P.; Silver, D. Meta-gradient reinforcement learning. *Adv. Neural Inf. Process. Syst.* **2018**, *31*.
19. Wang, L.; Zhang, Y.; Zhu, D.; Coleman, S.; Kerr, D. Supervised meta-reinforcement learning with trajectory optimization for manipulation tasks. *IEEE Trans. Cogn. Dev. Syst.* **2023**, *16*, 681–691. [\[CrossRef\]](#)
20. Zhang, X.; Ruan, J.; Ma, X.; Zhu, Y.; Chen, J.; Zeng, K.; Cai, X. Reasoner for Real-World Event Detection: Scaling Reinforcement Learning via Adaptive Perplexity-Aware Sampling Strategy. *arXiv* **2023**, arXiv:2507.01327.
21. Knox, W.B.; Allievi, A.; Banzhaf, H.; Schmitt, F.; Stone, P. Reward (mis) design for autonomous driving. *Artif. Intell.* **2023**, *316*, 103829. [\[CrossRef\]](#)
22. Ng, A.Y.; Harada, D.; Russell, S. Policy invariance under reward transformations: Theory and application to reward shaping. In Proceedings of the ICML, Bled, Slovenia, 27–30 June 1999; Volume 99, pp. 278–287.
23. Bai, Y.; Jones, A.; Ndousse, K.; Askell, A.; Chen, A.; DasSarma, N.; Drain, D.; Fort, S.; Ganguli, D.; Henighan, T.; et al. Training a helpful and harmless assistant with reinforcement learning from human feedback. *arXiv* **2022**, arXiv:2204.05862. [\[CrossRef\]](#)
24. Pan, A.; Bhatia, K.; Steinhardt, J. The effects of reward misspecification: Mapping and mitigating misaligned models. *arXiv* **2022**, arXiv:2201.03544. [\[CrossRef\]](#)
25. Zhang, Y.; Du, Y.; Huang, B.; Wang, Z.; Wang, J.; Fang, M.; Pechenizkiy, M. Interpretable reward redistribution in reinforcement learning: A causal approach. *Adv. Neural Inf. Process. Syst.* **2023**, *36*, 20208–20229.
26. Foerster, J.; Farquhar, G.; Afouras, T.; Nardelli, N.; Whiteson, S. Counterfactual multi-agent policy gradients. In Proceedings of the AAAI Conference on Artificial Intelligence, New Orleans, LA, USA, 2–7 February 2018; Volume 32.
27. Ahn, M.; Brohan, A.; Brown, N.; Chebotar, Y.; Cortes, O.; David, B.; Finn, C.; Fu, C.; Gopalakrishnan, K.; Hausman, K.; et al. Do as i can, not as i say: Grounding language in robotic affordances. *arXiv* **2022**, arXiv:2204.01691. [\[CrossRef\]](#)
28. Yao, S.; Zhao, J.; Yu, D.; Du, N.; Shafran, I.; Narasimhan, K.; Cao, Y. React: Synergizing reasoning and acting in language models. In Proceedings of the International Conference on Learning Representations (ICLR), Kigali, Rwanda, 1–5 May 2023.
29. Shinn, N.; Cassano, F.; Gopinath, A.; Narasimhan, K.; Yao, S. Reflexion: Language agents with verbal reinforcement learning. *Adv. Neural Inf. Process. Syst.* **2023**, *36*, 8634–8652.
30. Wang, G.; Xie, Y.; Jiang, Y.; Mandlekar, A.; Xiao, C.; Zhu, Y.; Fan, L.; Anandkumar, A. Voyager: An open-ended embodied agent with large language models. *arXiv* **2023**, arXiv:2305.16291. [\[CrossRef\]](#)
31. Wang, L.; Ma, C.; Feng, X.; Zhang, Z.; Yang, H.; Zhang, J.; Chen, Z.; Tang, J.; Chen, X.; Lin, Y.; et al. A survey on large language model based autonomous agents. *Front. Comput. Sci.* **2024**, *18*, 186345. [\[CrossRef\]](#)
32. Zeng, F.; Gan, W.; Wang, Y.; Liu, N.; Yu, P.S. Large language models for robotics: A survey. *arXiv* **2023**, arXiv:2311.07226. [\[CrossRef\]](#)
33. Kwon, M.; Xie, S.M.; Bullard, K.; Sadigh, D. Reward design with language models. *arXiv* **2023**, arXiv:2303.00001. [\[CrossRef\]](#)
34. Ma, Y.J.; Liang, W.; Wang, G.; Huang, D.A.; Bastani, O.; Jayaraman, D.; Zhu, Y.; Fan, L.; Anandkumar, A. Eureka: Human-level reward design via coding large language models. *arXiv* **2023**, arXiv:2310.12931.
35. Qu, Y.; Jiang, Y.; Wang, B.; Mao, Y.; Wang, C.; Liu, C.; Ji, X. Latent reward: Llm-empowered credit assignment in episodic reinforcement learning. In Proceedings of the AAAI Conference on Artificial Intelligence, Philadelphia, PA, USA, 25 February–4 March 2025; Volume 39, pp. 20095–20103.
36. Puterman, M.L. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*; John Wiley & Sons: Hoboken, NJ, USA, 2014.
37. Sutton, R.S.; Barto, A.G. *Reinforcement Learning: An Introduction*; MIT Press: Cambridge, MA, USA, 1998.
38. Burda, Y.; Edwards, H.; Storkey, A.; Klimov, O. Exploration by random network distillation. *arXiv* **2018**, arXiv:1810.12894. [\[CrossRef\]](#)
39. Badia, A.P.; Sprechmann, P.; Vitvitskyi, A.; Guo, D.; Piot, B.; Kapturowski, S.; Tieleman, O.; Arjovsky, M.; Pritzel, A.; Bolt, A.; et al. Never give up: Learning directed exploration strategies. *arXiv* **2020**, arXiv:2002.06038. [\[CrossRef\]](#)
40. Raileanu, R.; Rocktäschel, T. Ride: Rewarding impact-driven exploration for procedurally-generated environments. *arXiv* **2020**, arXiv:2002.12292.
41. Barreto, A.; Dabney, W.; Munos, R.; Hunt, J.J.; Schaul, T.; van Hasselt, H.P.; Silver, D. Successor features for transfer in reinforcement learning. *Adv. Neural Inf. Process. Syst.* **2017**, *30*.
42. Jothimurugan, K.; Bansal, S.; Bastani, O.; Alur, R. Compositional reinforcement learning from logical specifications. *Adv. Neural Inf. Process. Syst.* **2021**, *34*, 10026–10039.
43. Ziebart, B.D.; Maas, A.L.; Bagnell, J.A.; Dey, A.K. Maximum entropy inverse reinforcement learning. In Proceedings of the AAAI, Chicago, IL, USA, 13–17 July 2008; Volume 8, pp. 1433–1438.
44. Wei, J.; Wang, X.; Schuurmans, D.; Bosma, M.; Xia, F.; Chi, E.; Le, Q.V.; Zhou, D. Chain-of-thought prompting elicits reasoning in large language models. *Adv. Neural Inf. Process. Syst.* **2022**, *35*, 24824–24837.
45. Zhang, H.; Sun, K.; Xu, B.; Kong, L.; Müller, M. A Distance-based Anomaly Detection Framework for Deep Reinforcement Learning. *arXiv* **2021**, arXiv:2109.09889.

46. Müller, R.; Illium, S.; Phan, T.; Haider, T.; Linnhoff-Popien, C. Towards anomaly detection in reinforcement learning. In Proceedings of the 21st International Conference on Autonomous Agents and Multiagent Systems, Virtual, 9–13 May 2022; pp. 1799–1803.
47. Sorrenti, A.; Bellitto, G.; Salanitri, F.P.; Pennisi, M.; Spampinato, C.; Palazzo, S. Selective freezing for efficient continual learning. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Paris, France, 1–6 October 2023; pp. 3550–3559.
48. Todorov, E.; Erez, T.; Tassa, Y. Mujoco: A physics engine for model-based control. In Proceedings of the 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, Algarve, Portugal, 7–12 October 2012; pp. 5026–5033.
49. Efroni, Y.; Merlis, N.; Mannor, S. Reinforcement learning with trajectory feedback. In Proceedings of the AAAI Conference on Artificial Intelligence, Virtually, 19–21 May 2021; Volume 35, pp. 7288–7295.
50. Gangwani, T.; Zhou, Y.; Peng, J. Learning guidance rewards with trajectory-space smoothing. *Adv. Neural Inf. Process. Syst.* **2020**, *33*, 822–832.
51. Alemi, A.A.; Fischer, I.; Dillon, J.V.; Murphy, K. Deep variational information bottleneck. *arXiv* **2016**, arXiv:1612.00410.
52. Lillicrap, T.P.; Hunt, J.J.; Pritzel, A.; Heess, N.; Erez, T.; Tassa, Y.; Silver, D.; Wierstra, D. Continuous control with deep reinforcement learning. *arXiv* **2015**, arXiv:1509.02971.
53. Haarnoja, T.; Zhou, A.; Abbeel, P.; Levine, S. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In Proceedings of the International Conference on Machine Learning, Stockholm, Sweden, 10–15 July 2018; pp. 1861–1870.

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.