Computers, Materials & Continua

Tech Science Press

https://doi.org/10.32604/cmc.2026.086261

ARTICLE

# SD-KRE: A Method for Structural Decoupling and Knowledge Reuse Evolution of Reinforcement Learning Reward Functions Assisted by Large Language Models

**Yuqing Cao, Xiliang Chen*, Legui Zhang*, Jun Lai, Haoyang Dong, Xuefei Sun and Xiaoyan Wang**

College of Command and Control Engineering, Army Engineering University of PLA, Nanjing, China

*Corresponding Authors: Xiliang Chen. Email: lgd_chenxiliang@aeu.edu; Legui Zhang. Email: zlg2026@aeu.edu

Received: 27 May 2026; Accepted: 16 July 2026; Published: 13 August 2026

**ABSTRACT:** The design of reward functions is crucial to the success of reinforcement learning, yet the process often relies on expert experience and is difficult to debug. Although large language models (LLMs) offer new opportunities for automated reward design, existing methods still face challenges such as poor interpretability, inability to reuse knowledge, and optimization blindness. To address these issues, this paper proposes a method for structural decoupling and knowledge reuse evolution, referred to as SD-KRE. Its core lies in treating the reward function as a composition of multiple structured units with clear semantics and functionally decoupled components, and in measuring the importance of each unit's utility through a multi-dimensional quantitative evaluation method; The quantified structural units are stored in a structured reward prior knowledge base and serve as prior knowledge to guide the LLM in targeted reuse and evolutionary optimization during subsequent iterations. Experiments on six continuous control and dexterous manipulation benchmarks show SD-KRE significantly outperforms Eureka and Text2Reward in final performance, convergence speed, and stability, while matching or surpassing manually designed rewards on most tasks. Its stronger alignment with human prior knowledge further validates the semantic validity of the approach. By enabling structured reuse and directed evolution, SD-KRE overcomes the blindness and forgetfulness of conventional LLM-based reward design. It delivers expert-level rewards with higher efficiency, providing a reliable plug-and-play solution for complex reinforcement learning tasks.

**KEYWORDS:** Reinforcement learning; large language models; reward design; structural decoupling; knowledge reuse; evolutionary optimization

## 1 Introduction

Reinforcement learning (RL) has achieved remarkable success in sequential decision-making tasks such as robotic control and autonomous driving [1]. As the core signal that defines task objectives and guides agent behaviors, the reward function directly determines the upper bound of policy performance and learning efficiency [2]. However, designing effective, stable and interpretable reward functions remains a critical bottleneck in complex scenarios, especially for high-dimensional continuous control and dexterous manipulation tasks.

Traditional reward design relies heavily on manual construction and potential-based reward shaping [2,3], which demands strong expert experience, leads to high design costs and produces structurally opaque and hard-to-debug rewards [4]. Data-driven methods including inverse reinforcement learning [5],

imitation learning [6] and preference learning [7] reduce manual intervention, but suffer from high data consumption, noisy signals and poor interpretability.

Benefiting from strong semantic understanding and code generation capabilities, large language models (LLMs) have brought a new paradigm for automated reward design. Existing LLM-based methods can be divided into reward initializers [8–13], human-in-the-loop iterators [14–16] and human-out-of-the-loop iterators [11–13,17–21]. Despite notable progress, they still suffer from three critical limitations: (1) Lack of Interpretability: Generated rewards are often treated as indivisible black-box code, making it impossible to debug or analyze the contribution of specific components [22]. (2) Knowledge Forgetting: Optimization strategies are typically memoryless; effective sub-routines or heuristic rules discovered in previous iterations are discarded and not retained for future reuse [11]. (3) Blind Optimization: The search process relies solely on global performance scores, lacking fine-grained guidance on which specific parts of the reward function need modification, leading to inefficient exploration.

To address these issues, this paper proposes SD-KRE, a structural decoupling and knowledge reuse evolution method for LLM-assisted RL reward function design. The core idea is to transform reward design from holistic black-box exploration into structured, interpretable and reusable white-box evolution. Unlike existing approaches that treat the reward as a monolithic script, SD-KRE decomposes the reward function into semantically distinct, functionally decoupled Reward Structural Units (RSUs). We introduce a multi-dimensional quantitative evaluator to score each unit based on performance correlation, stability, and temporal trends. These quantified units are stored in a Structured Reward Prior Knowledge Base. In subsequent iterations, high-scoring units are retrieved as prior knowledge to guide the LLM in targeted reuse and evolutionary optimization, effectively transforming the process from random mutation to directed evolution.

We validate SD-KRE on six benchmark tasks involving continuous control and dexterous manipulation. Results show SD-KRE outperforms Eureka in final performance, convergence speed, and stability, while matching or exceeding human-designed rewards. Ablation studies confirm the necessity of structural decoupling, multi-dimensional quantification, and knowledge reuse.

The main contributions are summarized as follows:

- We propose the SD-KRE framework that integrates structural decoupling and knowledge reuse evolution for LLM-assisted reward design.
- We design a lightweight structural unit extraction strategy and a multi-dimensional quantitative evaluation method for unit importance.
- We conduct extensive experiments on six benchmark tasks, verifying that SD-KRE outperforms state-of-the-art methods and achieves comparable or better performance than human-designed rewards.

The rest of this paper is organized as follows: Section 2 reviews related work. Section 3 elaborates the SD-KRE method. Section 4 presents experimental results and analysis. Section 5 concludes this paper.

## 2 Related Work

### *2.1 Reward Design in Reinforcement Learning*

In a standard Markov Decision Process (MDP), the reward function $R(s, a, s')$ provides scalar feedback that guides agents toward desired behaviors [1]. Reward design has evolved from explicit manual engineering to implicit data-driven learning. Traditional reward design is dominated by manual construction and potential-based reward shaping (PBRS) [2]. Such methods rely on expert experience to encode task objectives and can ensure policy invariance in theory. However, in high-dimensional continuous control and dexterous manipulation, manual design suffers from high cost, sparse rewards and ambiguity in target expression.

To reduce human dependence, data-driven reward learning methods have been proposed, including inverse reinforcement learning (IRL) [5], imitation learning [6], preference learning [7], and intrinsic motivation-based learning [23,24]. These methods derive reward signals from data and alleviate the sparse reward problem, but the challenge of efficiently extracting reward structures that are semantically clear, interpretable and optimizable from limited, noisy or subjective signals remains a pressing issue. This also presents a key opportunity for large models.

### *2.2 Reinforcement Learning Reward Design Methods Assisted by Large Language Models*

Benefiting from strong code generation and semantic understanding, large language models have driven reward design into an automated paradigm. Existing methods fall into three categories: reward initializers, human-in-the-loop iterators, and human-out-of-the-loop iterators.

Reward initializers generate complete reward codes in a zero-shot manner according to task descriptions. Representative works include Text2Reward [8], Auto MC-reward [9], Eureka [11], DrEureka [12], ERFSL [13]. These methods quickly realize zero-to-one reward construction but lack iterative optimization and are susceptible to model hallucinations, especially in complex manipulation tasks.

Human-in-the-loop iterators refine rewards through real-time human natural language or score feedback, such as REvolve [15], LLM-HFBF [16] and EUREKA-HF [11]. Although these methods improve alignment with human intent, they introduce heavy labor cost and long feedback latency, making them unsuitable for large-scale automated training.

Human-out-of-the-loop iterators, represented by Eureka [11], use evolutionary algorithms and trajectory feedback to achieve fully autonomous reward iteration. Other works such as ERFSL [13], CARD [17], RLAIF vs. RLHF [18], LIV [19], ULTRA [20], further improve efficiency by analyzing training logs or interaction data. They achieve impressive performance on robotic tasks, but still suffer from issues such as poor interpretability, high dependence on the quality of trajectory data, and a tendency to get stuck in local optima in complex scenarios.

Existing LLM-assisted reward design methods suffer from poor interpretability, knowledge waste, and blind optimization. Specifically, they treat reward functions as indivisible entities, fail to retain effective units across iterations, and lack fine-grained feedback for targeted updates. The SD-KRE method is proposed specifically to address these issues.

### *2.3 Research on the Interpretability and Structured Nature of Reward Functions*

To improve reward interpretability, recent research has explored structural decomposition and modular design. Works like [25,26] validate that breaking down rewards into sub-objectives improves traceability, while the R* framework [21] combines evolutionary algorithms with LLMs for module mutation and parameter optimization. However, current LLM-assisted methods still face three key bottlenecks that hinder efficient evolution:

Firstly, a dominance of static structures and a lack of dynamic quantitative evaluation. Existing methods largely rely on manually predefined decompositions and fixed weight configurations, making it difficult to adaptively measure the actual marginal contribution of individual structural units to policy learning from training data. Due to the absence of a data-driven dynamic evaluation mechanism, the system is unable to effectively distinguish between core effective units and noise-introducing units, resulting in a lack of fine-grained guidance for reward optimization.

Secondly, structures are isolated to single instances, lacking cross-iteration knowledge accumulation. Most approaches treat structured representations merely as intermediate forms generated for a single reward, failing to systematically manage and reuse effective structures as inheritable knowledge. During iterative optimization, validated high-value sub-structures are easily discarded as the overall code mutates, making it difficult to form cumulative evolutionary prior knowledge. This, in turn, triggers significant knowledge forgetting issues, resulting in each iteration remaining in a state of inefficient, repetitive exploration.

Thirdly, the decoupling granularity is coarse and mismatched with evolutionary operators. Existing structured representations typically remain at a relatively coarse target level, lacking fine-grained, semantically clear decoupling at the atomic level. Such coarse-grained representations are difficult to reconcile with evolutionary algorithms and cannot support efficient genetic operations such as directed mutation, recombination and selection, ultimately reducing search efficiency and undermining the controllability of the optimization direction.

To fill these gaps, this paper proposes SD-KRE, which unifies the modelling of reward structuring, interpretability and knowledge reuse evolution. Through autonomous decoupling, multi-dimensional evaluation and a structured prior knowledge base, it enables the automatic design of interpretable, reusable and evolvable reward functions driven by large language models, thereby providing a more robust and transparent reward optimization mechanism for complex reinforcement learning tasks.

## 3 SD-KRE

To address core issues in the design of reward functions for large language models-assisted reinforcement learning—such as the black-box nature of the process, the lack of optimization guidance, and the loss of valuable design expertise—this paper proposes an evolutionary method for the structural decoupling and knowledge reuse of reward functions. By structurally decomposing the reward functions generated by large language models, quantifying their importance across multiple dimensions, and systematically consolidating knowledge, this method provides clear reference criteria and optimization directions for the iterative generation by large language models, thereby facilitating a transition from blind exploration to targeted optimization of reward functions. The method firstly guides the LLMs to generate reward functions composed of semantically explicit, structured units; subsequently, using a lightweight yet robust analyzer, it quantifies the marginal contribution of each structured unit to task performance from training logs; all high-value units are stored in a reward prior knowledge base and injected into LLMs prompts as prior knowledge during subsequent iterations, thereby enabling the targeted reuse and improvement of high-quality patterns.

Fig. 1 shows the architectural flowchart of the SD-KRE method. The following sections will discuss three key technical aspects in turn: the structural decoupling of the reward function, the multi-dimensional quantitative assessment of the importance of reward structural units, and the construction and reuse of a structured reward prior knowledge base.

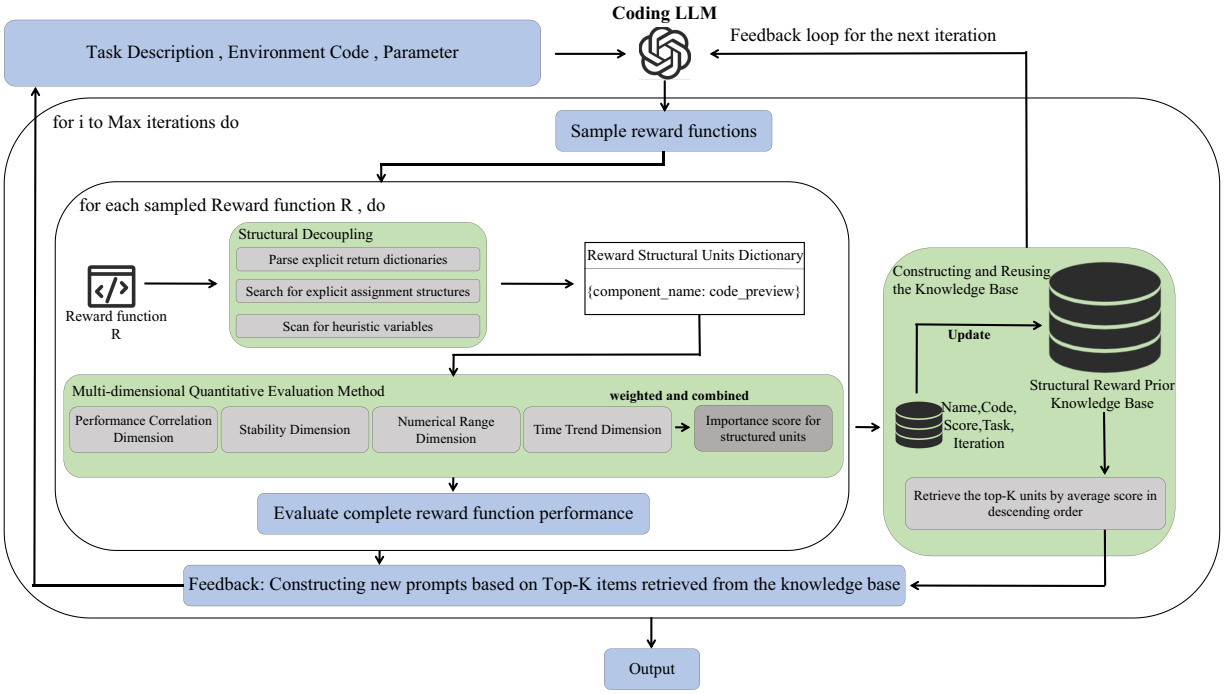


**Figure 1:** The overall framework of SD-KRE.

The pseudocode for the SD-KRE method is shown in Algorithm 1:

**Algorithm 1:** SD-KRE

**Require:** Task description $l$, environment code $M$, coding LLM $\mathcal{LLM}$, fitness function $F$, initial prompt prompt, knowledge base $KB = \varnothing$, Reward Structural Unit Set $U$, max iterations $N$, iteration batch size $K$, Top-K retrieval size $K_{\text{unit}}$

**for** $n = 1$ **to** $N$ **do**
  $R_1, \ldots, R_3 \leftarrow \text{LLM}(l, M, \text{prompt})$
  **for all** $R_i \in \{R_1, \ldots, R_k\}$ **do**
    $U_i \leftarrow \text{Structural Decoupling}(R_i)$
    **for all** $u \in U_i$ **do**
      $S_1(u) \leftarrow \text{Corr}(u, F.\text{performance})$
      $S_2(u) \leftarrow \text{Stability}(u, CV_{\text{target}} = 0.4)$
      $S_3(u) \leftarrow \text{Range}(u, \text{target norm} = 1)$
      $S_4(u) \leftarrow \text{Trend}(u)$
      $S(u) = \omega_1 \cdot S_1(u) + \omega_2 \cdot S_2(u) + \omega_3 \cdot S_3(u) + \omega_4 \cdot S_4(u)$
      $\text{KB.update}(u.\text{name}, u.\text{code}, S(u), \text{task} = l, \text{iteration} = n)$
    **end for**
    $s_i \leftarrow F(R_i)$
  **end for**
  $\text{best idx} \leftarrow \arg\max_{i \in \{1,k\}} s_i$
  $R_{\text{best}} \leftarrow R_{\text{best idx}}, s_{\text{best}} \leftarrow s_{\text{best idx}}$
  **if** $s_{\text{best}} >$ best score **then**
    best reward $\leftarrow R_{\text{best}}$, best score $\leftarrow s_{\text{best}}$

(Continued)

**Algorithm 1 (continued)**

19: **end if**
20: $U_{\text{top-K}} \leftarrow$ KB.retrieve top-K($K_{\text{unit}}$)
21: prompt $\leftarrow$ Construct Prompt($l, U_{\text{top-K}}$)
22: **end for**
**Ensure:** best reward

### *3.1 Methods for Structural Decoupling of the Reward Function*

This paper defines a Reward Structural Unit (RSU) as the smallest sub-reward unit within a reward function that is functionally independent, semantically complete and indivisible. Each structural unit is syntactically self-contained and logically decoupled, and can be directly reused, combined or iteratively optimised as a basic unit.

The structural decomposition of reward functions forms the cornerstone of the SD-KRE framework, with the aim of automatically identifying and isolating all independent structured units that constitute the total reward from the reward function code generated by the LLMs. To this end, we impose explicit structural constraints on the LLMs' output during the prompt engineering stage: when generating the `compute_reward` function, it must return two results simultaneously:

- A scalar total reward used for policy optimization;
- A dictionary of reward structural units named `rew_dict` where the keys are units' names with clear semantic meaning, such as `distance_reward` and `velocity_penalty`, and the values are the reward calculation expressions for the corresponding units.

To accommodate the diversity in format and style of code generated by LLMs, this paper proposes a robust three-stage progressive extraction strategy. The strategy takes a reward function string as input and outputs a structural units dictionary {`component_name: code_preview`}. The specific process is as follows:

Firstly, parse explicit return dictionaries. Prioritize matching the `return {...}` statement at the end of a function using regular expressions to directly extract the key-value pair structure.

Secondly, search for explicit assignment structures. If step 1 fails, fall back to searching for assignment statements of the form `rew_dict = {...}` to parse units' definition from them.

Thirdly, scan for heuristic variables. For code lacking explicit dictionary structures, further employ heuristic rules guided by naming conventions—identifying variable assignments prefixed with `reward_` or `penalty_` and reconstructing complete expressions in accordance with Python indentation rules.

Different from AST-based traditional parsing, SD-KRE uses lightweight regular expression and string matching with linear time complexity $O(L)$ relative to reward code length $L$. In practice, single reward decomposition only consumes milliseconds, introducing negligible constant overhead compared to RL step costs and barely affecting overall training efficiency. Fig. 2 provides complete Python code and extracted structural units to demonstrate the full parsing workflow for reproduction.

### *3.2 A Multi-Dimensional Quantitative Evaluation Method for the Importance of Reward Structural Units*

To accurately quantify the marginal contribution of each reward structural unit within the reward function generated in the previous iteration to the effectiveness of reinforcement learning training, this paper proposes a multi-dimensional, interpretable method for quantifying importance. This method evaluates the utility of each unit across four complementary dimensions—performance correlation, signal stability,

reasonableness of numerical range, and temporal evolution trends—and derives a normalized importance score $S(u) \in [0,1]$ through weighted fusion. This score is utilized for the subsequent storage, ranking, and retrieval of the structured reward prior knowledge base. The definitions and calculation methods for each dimension are as follows.

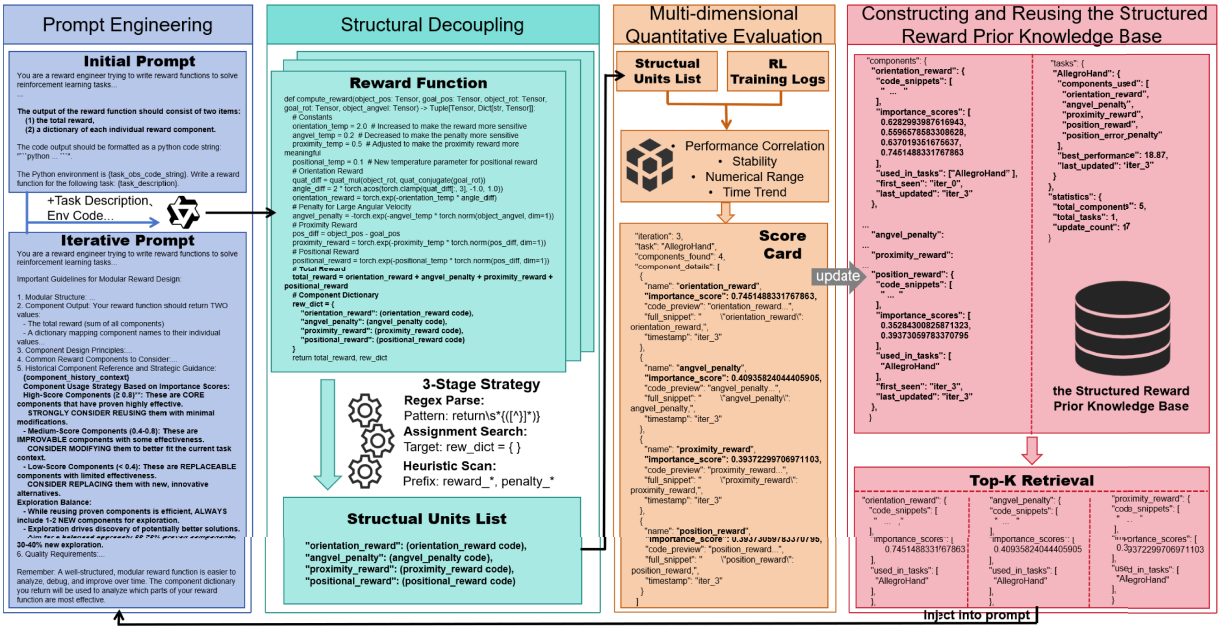


**Figure 2:** The complete technical pipeline of SD-KRE (exemplified by the 3rd iteration of the AllegroHand task). This figure illustrates the full-link implementation from prompt engineering and structural decoupling to multi-dimensional quantitative evaluation, knowledge base storage, and Top-K retrieval. The code snippets, component scores, and knowledge base records displayed in the figure are all derived from real experimental data.

### *3.2.1 Performance Correlation Dimension Score $S_1$*

This dimension is used to measure the degree of alignment between a unit's values and the overall performance of the task. A higher correlation suggests a stronger linear association with task performance and serves as an indicative proxy for the unit's potential importance, though correlation alone does not imply causal contribution. Core performance metrics for the task, such as average cumulative return, task success rate and number of consecutive completions, are selected to construct a performance sequence $P = [p_1, p_2, \ldots, p_T]$ (where $T$ represents the number of training steps). The corresponding unit values sequence $U = [u_1, u_2, \ldots, u_T]$ is then extracted, and the correlation is calculated using the Pearson correlation coefficient, as shown in Eq. (1):

$$S_1(u) = |\mathrm{corr}(P,U)| = \frac{\mathrm{Cov}(P,U)}{\sqrt{\mathrm{Var}(P)\cdot\mathrm{Var}(U)}} \tag{1}$$

where $\mathrm{Cov}(P,U)$ represents the covariance, and $\mathrm{Var}(P)$ and $\mathrm{Var}(U)$ represent the variances of the performance and unit values, respectively. This score directly reflects the extent to which the unit contributes to the overall task and serves as the core basis for ranking by importance.

*3.2.2 Stability Dimension Score $S_2$*

An effective reward unit should output a steady and distinguishable continuous signal; excessive fluctuation introduces noise, whilst insufficient fluctuation fails to provide an effective gradient. First, the variance Var(U) of the structured unit value sequence $U$ is calculated to characterize the absolute degree of fluctuation in the structured unit values. Next, the absolute value of the mean $\mu_u = \left|\overline{U}\right|$ of the unit value sequence is calculated, and the coefficient of variation is used to eliminate evaluation bias caused by differences in the magnitude of values across different units, as shown in Eq. (2):

$$CV(U) = \frac{\sqrt{\mathrm{Var}(U)}}{\mu_u + \varepsilon} \tag{2}$$

where $\varepsilon$ is a very small positive number to avoid a denominator of zero. Based on the continuous nature of the coefficient of variation, a smooth stability scoring function is constructed, as shown in Eq. (3):

$$S_2(u) = \left(\frac{CV(U)}{CV_{\text{target}}}\right) \cdot e^{\left(1-\frac{CV(U)}{CV_{\text{target}}}\right)} \tag{3}$$

where $CV_{\text{target}} = 0.4$ is the optimal target coefficient of variation. This function constrains the score to $[0,1]$ and attains a peak at $CV(U) = 0.4$, thereby effectively screening for stable and effective reward signals.

*3.2.3 Numerical Range Dimension Score $S_3$*

The values for each unit should fall within a reasonable range; if the range is too narrow, there will be insufficient discrimination, whilst if it is too wide, it may lead to an imbalance in rewards. For this dimension, the range of unit values $R(U) = \max(U) - \min(U)$ is first calculated and then normalized using the absolute value of the mean of the total rewards $\mu_{\text{total}} = \left|\overline{R}_{\text{total}}\right|$, as shown in Eq. (4):

$$R_{\text{norm}}(U) = \frac{R(U)}{\mu_{\text{total}} + \varepsilon} \tag{4}$$

where $\varepsilon$ is a very small positive number to avoid a denominator of zero. A Gaussian-type continuous scoring function is constructed based on the normalized range, such that $S_3(u) \in [0,1]$, as shown in Eq. (5):

$$S_3(u) = \exp\left(-\frac{2 \times (R_{\text{norm}}(U) - 1)^2}{9}\right) \tag{5}$$

This function has $R_{\text{norm}}(U) = 1$ as its optimal discrimination peak, automatically suppressing invalid units with excessively small or large values.

*3.2.4 Time Trend Dimension Score $S_4$*

The values of high-quality reward units should exhibit a steady upward trend throughout the training process, gradually converging towards the optimal policy. For this dimension, the sequence of unit values $U$ is divided into $K = 5$ equally-sized sub-sequences based on the number of training steps. The mean of each sub-sequence $\mu_1, \mu_2, \ldots, \mu_K$ is calculated to form a trend sequence $\mu$; a linear fit is then performed to obtain a slope $k$ that characterizes the direction of evolution, which is mapped via a sigmoid function to yield the score $[0,1]$, as shown in Eq. (6):

$$S_4(u) = \frac{1}{1 + \exp\left(-\frac{k}{\sigma_\mu + \varepsilon}\right)} \tag{6}$$

where $\sigma_\mu$ is the standard deviation of the trend sequence, and $\varepsilon$ is a very small positive number to avoid a zero denominator. When a unit exhibits a clear upward trend, the score approaches 1; for a downward trend, it approaches 0; and when there is no clear trend, it is approximately 0.5.

Finally, the scores from the four dimensions are weighted and combined according to the predefined weights to obtain the final importance score for the structured unit, as shown in Eq. (7):

$$S(u) = \omega_1 S_1(u) + \omega_2 S_2(u) + \omega_3 S_3(u) + \omega_4 S_4(u) \tag{7}$$

Based on these scores, the units are categorized into three levels for the purposes of knowledge reuse and iterative evolution:

- Core units: $S(u) \in [0.8, 1]$, which make a significant contribution to performance and should be prioritized for inclusion in the knowledge base and reused;
- Auxiliary units: $S(u) \in (0.4, 0.8)$, which make a certain contribution but have room for improvement; fine-tuning of parameters is possible;
- Invalid units: $S(u) \in [0, 0.4]$, which make no substantive contribution or cause negative interference, shall be excluded or restructured.

Note that each individual metric captures only one aspect of a unit's behavior; in particular, Pearson correlation measures linear association rather than causation. Therefore, the final importance score $S(u)$ relies on the integration of multiple complementary dimensions to mitigate the risk of misinterpreting spurious correlations as causal effects.

This multi-dimensional quantification scheme differentiates valid, noisy and invalid structural units to support knowledge storage, retrieval and evolutionary optimization in SD-KRE. The evaluation relies on linear statistical calculations with per-iteration complexity $O(n \bullet M \bullet T)$ per iteration, where $n$ denotes the number of sampled rewards, $M$ is the number of units, and $T$ represents the timesteps. Since unit count is far lower than state-action dimensions and all computations run offline on CPU, the process only takes milliseconds and creates no training bottleneck. Fig. 2 presents a full scoring case of four units on the AllegroHand task to demonstrate the evaluation module's standard output format.

### *3.3 Methods for Constructing and Reusing the Structured Reward Prior Knowledge Base*

All Reward Structured Units (RSUs) that have undergone multi-dimensional importance assessment will be uniformly stored in the Structural Reward Prior Knowledge Base, thereby enabling the structured consolidation, long-term preservation and traceable management of high-quality reward design expertise. The knowledge base utilizes the JSON format for standardized storage, featuring a concise structure, efficient querying and ease of expansion. Each knowledge record contains complete unit attributes: a structured unit name, an independently executable code snippet, multi-dimensional importance scores, the associated task identifier, the iteration discovery round, and the historical reuse count.

Before proceeding to the next round of reward function generation, the system injects high-value knowledge from the prior knowledge base into the prompts for the LLMs, thereby enabling prior-based directed evolution and reuse. To ensure optimization efficiency and focus on core effective patterns, this paper adopts a Top-K retrieval strategy based on importance scores: all structured units in the base are sorted in descending order by average importance score, and the top K high-value units with the highest scores are selected. Their names, code snippets and quantified scores are incorporated into the prompt context

as reference templates. The LLMs are guided to adaptively fine-tune their threshold parameters or weight coefficients based on the specific state space characteristics of the current task, whilst retaining their core semantic logic, thereby achieving a unification of structured reuse and contextual adaptation.

The structured reward knowledge base imposes trivial maintenance overhead. Each record merely takes hundreds of bytes, and long-running storage stays under 1 MB, with insertion and retrieval complexities of $O(1)$ and $O(NlogN)$, where $N$ is the total stored units. As iterations proceed, the knowledge base size converges and retrieval cost becomes negligible. Many structural units are reusable across tasks without duplicate storage, so total capacity rises sub-linearly with task quantity. Built-in semantic deduplication and low-value unit pruning remove redundant entries to avoid unbounded expansion. Even with hundreds of distinct tasks, its storage and runtime memory usage stay below 1 MB and create no resource bottlenecks.

Through the aforementioned mechanism, large language models can directly inherit and reuse reward structures validated in previous iterations while retaining a degree of exploratory freedom, thereby avoiding redundant and inefficient exploration. This process transforms traditional goal-less black-box search into a knowledge-driven evolution that is prior-informed, interpretable and optimized in a targeted manner, significantly enhancing the quality of reward function generation, convergence speed and optimization stability. The complete pipeline of our reward prior knowledge base—including storage, incremental updates, Top-K retrieval, and prompt injection—is fully illustrated in the rightmost panel of Fig. 2, accompanied by a complete JSON storage example of the structured units.

### *3.4 Comparative Analysis of Methods*

Table 1 provides a systematic comparison of existing representative methods leveraging large language models (LLMs) to design reward functions. TEXT2REWARD [8] and EUREKA [11] primarily treat reward functions as monolithic code blocks. The former focuses on zero-shot/few-shot generation coupled with human-feedback refinement, while the latter introduces evolutionary search and reward reflection mechanisms. However, both paradigms lack fine-grained decoupling of the internal reward structure. Consequently, their optimization processes heavily rely on the black-box inference or stochastic mutations of LLMs, hindering precise diagnosis and reuse of specific reward components.

**Table 1:** Comparison with existing LLM-based reward design methods.

| Comparison Dimension | Text2Reward [8] | Eureka [11] | R* [21] | SD-KRE |
|---|---|---|---|---|
| Core Strategy | Zero/Few-shot generation of dense reward code based on task descriptions, supporting iterative refinement with human feedback. | Leveraging LLM zero-shot generation and in-context improvement capabilities, performing iterative optimization via evolutionary search and reward reflection. | Decoupling reward design into structural evolution (module-level crossover + LLM reflection) and parameter alignment optimization (critic voting + preference learning). | Structurally decoupling reward functions into semantically independent Reward Structural Units (RSUs), establishing a structured reward prior knowledge base through multi-dimensional quantitative evaluation to enable knowledge-reusable evolution. |
| Structural Decoupling | ✗ | ✗ | ✓ | ✓ |

(Continued)

**Table 1 (continued)**

| Comparison Dimension | Text2Reward [8] | Eureka [11] | R* [21] | SD-KRE |
|---|---|---|---|---|
| Unit-level Quantitative Evaluation | ✗ | PC | ✗ | ✓ |
| Cross-round Knowledge Reuse | ✗ | ✗ | PC | ✓ |
| Overcomes Blind Optimization | PC | PC | PC | ✓ |
| Overcomes Catastrophic Forgetting | ✗ | ✗ | PC | ✓ |

Against this backdrop, R* [21] represents a notable advancement by pioneering the introduction of a modular reward structure. R* decomposes the reward function into independent modules, facilitates structural recombination across populations via module-level crossover, and leverages a critic population for parameter alignment. This design effectively enhances the efficiency of reward search and mitigates the greedy search problem caused by a single best candidate.

Despite R*'s architectural breakthrough, SD-KRE fundamentally differs from it in three key aspects:

- Ordinal Feedback vs. Multi-dimensional Diagnosis: R* only acquires coarse trajectory rankings via critic binary voting, whereas SD-KRE adopts four-dimensional scoring to quantify each RSU's contribution and screen defective units accurately;
- Elite Archive vs. Structured Knowledge Base: R* archives complete reward functions only for crossover selection, while SD-KRE builds a structured knowledge base to save reusable RSUs with task and iteration records, realizing cross-task knowledge transfer and mitigating catastrophic forgetting;
- Local Optimization vs. Global Memory: R* cannot locate faulty components merely relying on global performance, yet SD-KRE eliminates low-value units via multi-dimensional metrics and retains cross-epoch structural knowledge to fully reduce blind search and knowledge loss.

In summary, SD-KRE bridges the gaps in fine-grained interpretability, targeted optimization, and knowledge persistence among existing methods, marking a paradigm shift from black-box evolution to white-box knowledge evolution.

Notably, modern alignment frameworks like RLHF and RLAIF adopt fixed reward architectures and only tune parameters with preference data to match human preferences. This parameter-heavy, structure-light design carries intrinsic defects: fundamental structural defects including missing physical constraints and misleading reward signals cannot be fixed via pure parameter optimization, which may further trigger reward hacking.

Although R* adopts modularization to ease structural rigidity, its decomposition is syntax-limited and only conducts coarse binary voting without unit-wise contribution assessment. Differently, SD-KRE defines independent RSUs paired with four-dimensional scoring. Relying on the structured knowledge base, it realizes both parameter tuning and dynamic structural evolution, delivering physically reasonable, semantically intact reward functions with stronger robustness and interpretability than conventional alignment methods.

## 4 Experiment

### 4.1 Experimental Design

To evaluate the performance of the proposed SD-KRE method in automated reward function design, we devised an experimental framework encompassing four aspects: experimental environment and benchmarks, large language model configurations, comparison with baseline methods, and ablation experiments. This framework aims to validate the overall effectiveness of SD-KRE in enhancing final policy performance, accelerating convergence efficiency, and improving interpretability.

Experiments were conducted on the Isaac Gym [27] platform, selecting six reinforcement learning simulation environments with varying action/state space dimensions, control complexities, and task objectives as test benchmarks to evaluate the generalization ability of SD-KRE, as shown in Table 2. These tasks span challenges from basic continuous control to high-dimensional dexterous manipulation, effectively testing the robustness and generality of the automated reward function design method. The specific simulation environment is shown in Fig. 3.

**Table 2:** Basic information on experimental simulation environment.

| Environment | Task Description | Core Challenges | Action Space | State Space |
|---|---|---|---|---|
| Ball Balance | Control a tray to keep the ball at its centre. | Continuous control with high stability requirements. | 3 | 24 |
| Quadcopter | Control a quadcopter to fly from the starting point to the target point. | 12-Dimensional continuous control, high dynamics, underactuated system. | 12 | 21 |
| Ant | Control a quadruped robot to move forward at maximum speed. | 8-Joint high-dimensional continuous control, coordinated motion. | 8 | 60 |
| Humanoid | Control a humanoid bipedal robot to walk. | 21-Joint high-dimensional system, unstable and prone to falling. | 21 | 108 |
| Shadow Hand | Controlling a dexterous five-fingered hand to perform fine-motor tasks. | 20-Dimensional ultra-high-dimensional control, contact force sensing. | 20 | 42 |
| Allegro Hand | Control a four-fingered Allegro dexterous hand to perform tasks. | 16-Dimensional continuous control, multi-finger coordination, complex contact dynamics. | 16 | 88 |

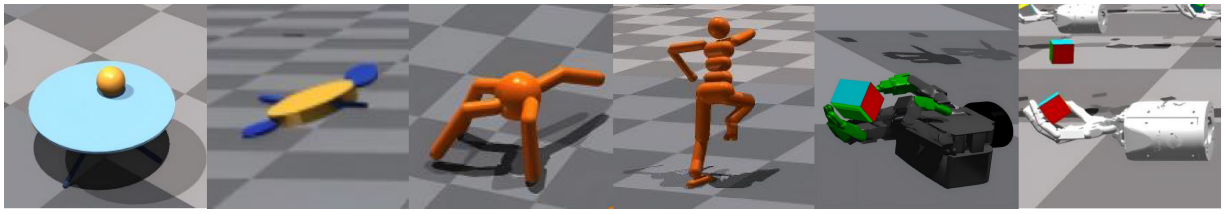

**Figure 3:** Simulation environment scenes.

We use Task Performance Score (TPS)—a task-normalized composite metric with environment-specific calculations—as the core performance indicator during training. It jointly evaluates final performance, convergence speed, and robustness, with higher values denoting better results. The specific definitions of TPS for each environment are as follows:

- Ball Balance: Measures the average number of steps during a training cycle during which the machine successfully prevents the ball from falling; a higher value indicates greater stability in balance control.
- Quadcopter: Expressed as the negative average Euclidean distance between the aircraft and the target position; a smaller absolute value indicates greater proximity to the target.
- Ant/Humanoid: Measures the cumulative progress reward of the robot moving from its initial position towards the target; a higher value indicates greater forward movement efficiency.
- Shadow Hand/Allegro Hand: The number of consecutive successes in specific tasks (such as grasping or rotation); a higher value indicates greater operational stability and success rates for the dexterous hand.

Furthermore, to quantify the alignment between generated reward signals and human prior knowledge, we introduced a Correlation metric, calculated as the Pearson correlation coefficient between the reward time series of the best-performing candidate reward and the manually designed benchmark reward in each iteration. A high correlation not only validated the validity of the generated reward function but also provided an intrinsic explanation for its outstanding performance, thereby enhancing the credibility and interpretability of the SD-KRE method in automatic reward design. All experiments were independently repeated five times using five different random seeds, and the mean results along with standard deviations are reported. All convergence curves in this section are plotted with 95% confidence intervals over five independent runs to ensure result reliability.

For fairness in comparison, both SD-KRE and the baseline method EUREKA used the Qwen-max API as the core LLM for reward generation and optimization, which excels in code generation, structural understanding, and prompt adherence, and supports parallel output. The underlying policy optimizer was uniformly set to the Proximal Policy Optimization (PPO) algorithm with fixed hyperparameters, as shown in Table 3. Both the policy network and the value function network adopted a two-layer MLP structure, with their specific architectures consistent with the environment.

**Table 3:** Hyperparameter settings.

| Hyperparameters | Values |
|---|---|
| Gamma | 0.99 |
| Learning rate | 5e−4 |
| Batch size | 4096 |
| Number of epochs | 10 |
| Clip range | 0.2 |
| Entropy coefficient | 0.01 |
| Maximum gradient norm | 0.5 |
| $\omega_1$ | 0.4 |
| $\omega_2$ | 0.2 |
| $\omega_3$ | 0.2 |
| $\omega_4$ | 0.2 |
| $K_{unit}$ | 3 |

To fully demonstrate the superiority of SD-KRE, three representative methods were chosen as baselines.

First, we consider EUREKA—a representative approach for LLM-based automatic reward function design. It builds an iterative framework via global code mutation and evolutionary selection. For fair comparison, our implementation strictly follows the original paper [11], using the same evolutionary pipeline. Key parameters (e.g., $K = 16$ samples per round, $N = 5$ total iterations) are consistent with SD-KRE to ensure equivalent computational cost. A notable limitation is that its in-context learning relies only on the top-performing reward function and overall score from the previous iteration, lacking unit-level structured analysis and knowledge reuse.

Second, we employ manually designed reward functions (Human) as a baseline. To evaluate whether our automated approach can match or exceed expert-level performance, we adopt widely used, finely tuned manual reward functions from representative papers and benchmarks for each task. These rewards typically take the form of weighted linear combinations of interpretable sub-objectives (e.g., distance, velocity, and attitude penalties), representing the performance upper bound from domain expertise. This baseline quantifies the performance gap between our automated method and the human-designed upper limit.

Third, Text2Reward. This method focuses on directly generating structured dense reward code from large language models in a zero-shot or few-shot manner, without involving an iterative optimization process. In this paper, we adopt its official prompt design and reward generation protocol, while keeping the LLM and RL hyperparameters strictly consistent with those of SD-KRE to ensure a fair comparison.

To deconstruct the SD-KRE framework and evaluate the contributions of its key components, we designed a series of ablation experiments. Each experiment investigated the impact on the final performance by removing or replacing specific parts of the framework. The specific variants are as follows:

- SD-KRE w/o SD: Structural decoupling is entirely removed; the LLM generates a monolithic, non-decomposable reward function. This variant validates the necessity of structural decomposition.
- SD-KRE w/o $S_1/S_2/S_3/S_4$: Four ablation variants are used to verify the contribution of each quantitative criterion. These variants respectively exclude the performance correlation, stability, numerical range, or temporal trend dimension from multi-dimensional scoring, and the remaining weights are renormalized.
- SD-KRE w/o KRE: The structured reward prior knowledge base is removed, disabling cross-iteration knowledge reuse. This variant tests the value of knowledge accumulation and directed optimization.

All experiments were run on a server with two NVIDIA GeForce RTX 3080 Ti GPUs, 128 GB RAM, and a 512 GB SSD, ensuring consistent computing resources.

### *4.2 Analysis of Comparative Experimental Results*

Fig. 4 illustrates the evolution of TPS (vertical axis) against RL training iterations (horizontal axis) across six benchmarks. Solid lines denote the mean performance over five independent runs, and the shaded regions represent the variability intervals computed based on multiple seeds. To verify the robustness of the improvements, we conducted independent two-sample t-tests on the results of five independent runs. The tests confirm that the performance improvement of SD-KRE over the baselines is statistically significant ($p < 0.05$).

As can be clearly seen from Fig. 4, SD-KRE significantly outperforms the Eureka and Text2Reward method across all tasks. Although Text2Reward can generate interpretable dense rewards in a zero-shot manner, it exhibits notable performance limitations on moderately complex tasks due to the absence of an iterative optimization mechanism. Compared to Eureka, SD-KRE not only demonstrates superior final convergence performance across the board, but also converges significantly faster. For example, in the

Humanoid and Allegro Hand tasks, as shown in Fig. 4d,f, the learning curves of the Eureka method exhibit a slow rise or even stagnation, whereas SD-KRE is able to rapidly identify effective reward structures and guide the policy towards efficient learning. This result strongly demonstrates the superiority of the reward function structural decoupling and knowledge reuse evolutionary method: by performing decoupling analysis and importance quantification on the reward function, SD-KRE is able to provide large language models with optimisation signals that are far more precise and instructive than those derived from global code mutation, thereby effectively overcoming the issues of blind exploration and knowledge waste inherent in traditional evolutionary methods.

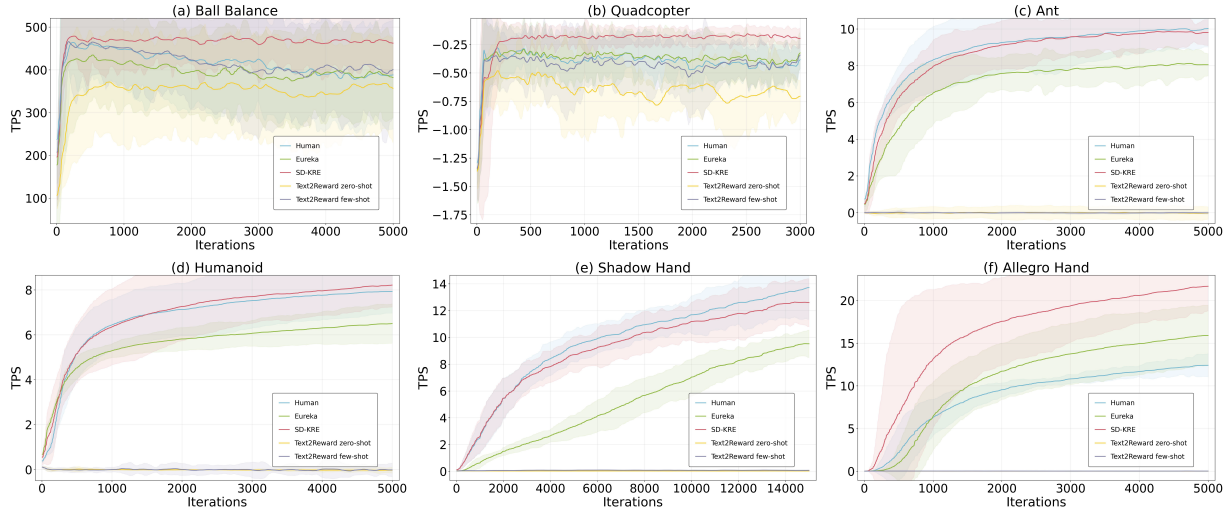


**Figure 4:** Experimental results comparing TPS metric.

Furthermore, SD-KRE demonstrates competitive performance against human expertise. It matches or exceeds manually designed rewards on BallBalance, Quadcopter, and Ant, and achieves near-expert or superior performance on challenging dexterous manipulation tasks (Shadow Hand, Allegro Hand). These results confirm that SD-KRE can automatically discover high-quality, interpretable reward structures, significantly lowering the barrier to applying RL in complex robotics.

Fig. 5 depicts the performance of the Correlation metric for the SD-KRE and EUREKA methods across six scenarios. The horizontal axis denotes the number of evolutionary iterations, and the vertical axis represents the correlation coefficient. Across all six benchmark tasks, SD-KRE demonstrates a distinct advantage over EUREKA: its correlation curves ascend more rapidly and converge to significantly higher terminal values, indicating a closer alignment with human expert intuition.

In relatively straightforward tasks such as Ball Balance and Quadcopter, as illustrated in Fig. 5a,b, SD-KRE rapidly achieved a high correlation approaching 0.9 after the first iteration and remained stable in subsequent iterations. This suggests that SD-KRE is capable of swiftly capturing core physical principles, such as position error and velocity penalties, and generating reward signals that are highly congruent with those designed by humans.

In complex tasks such as Ant and Humanoid, as shown in Fig. 5c,d, SD-KRE also demonstrates remarkable alignment capabilities. Its correlation values reach 1.0 or near 1.0 levels after the early iterations, indicating that the reward structure it generates is highly consistent with that designed by human experts.

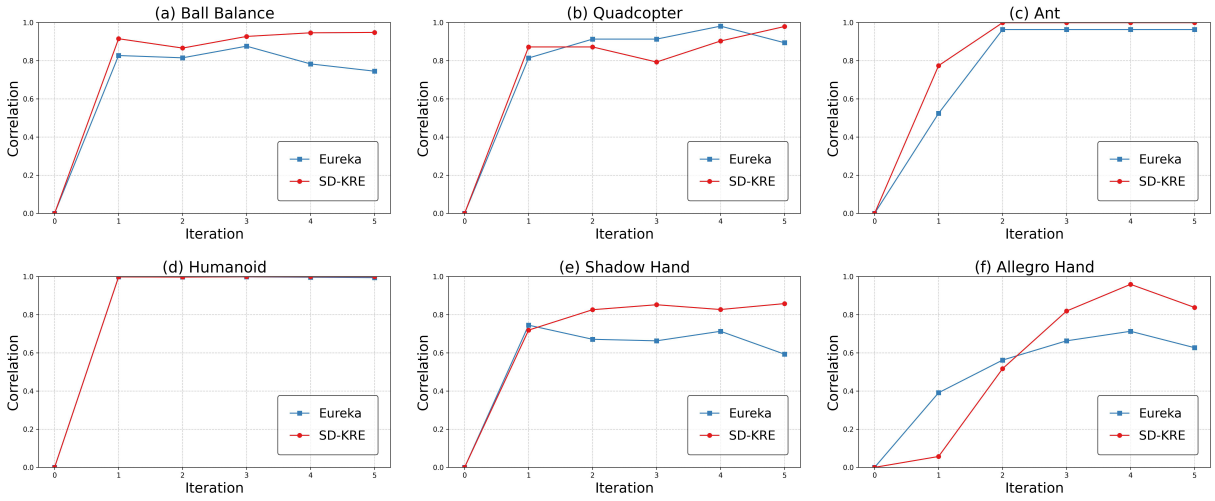


**Figure 5:** Experimental results comparing correlation metric.

In the most arduous dexterous hand tasks, Shadow Hand and Allegro Hand, as depicted in Fig. 5e,f, although the correlation curve of SD-KRE had a slow start, its correlation steadily improved as the iterations advanced and significantly outperformed that of Eureka in the later stages. This implies that, through its reward function structural decoupling and knowledge reuse evolution methods, SD-KRE is able to gradually uncover complex, human-intuitive reward patterns, such as finger coordination and contact force control. In contrast, Eureka is less efficient at exploring such fine-grained structures, resulting in reward signals that consistently fail to align adequately with those designed by experts.

The correlation results not only validate the superior quality of SD-KRE's reward signals but also provide a mechanistic explanation for its performance edge: the structured approach enables the LLM to evolve reward functions that are more consistent with human priors and task objectives, effectively overcoming the inefficiencies of blind exploration. Note that Pearson correlation only reflects the similarity of reward sequence trends instead of absolute value matching under linear assumptions, and high correlation cannot guarantee superior policy performance. Hence we adopt it merely as an auxiliary metric complementary to the core TPS for comprehensive reward assessment.

We further measure the computational overhead of SD-KRE for practical efficiency verification, with averaged runtime over six tasks listed in Table 4. SD-KRE consumes comparable total training time to Eureka. Its extra costs brought by decoupling, multi-dimensional scoring and knowledge base management are trivial relative to RL policy training, where log loading dominates decoupling latency. This matches our complexity analysis, proving SD-KRE yields prominent performance improvements with marginal extra computation.

In summary, the comparative experimental results fully validate the effectiveness and superiority of the SD-KRE method. Not only does it outperform the current state-of-the-art EUREKA method in terms of performance, but it also demonstrates strong potential to rival designs by human experts, successfully achieving a paradigm shift from blind exploration to directed optimization.

### *4.3 Analysis of Ablation Experiment Results*

Ablation experiments (as shown in Table 5) quantitatively verify the contribution and indispensability of each core module in the SD-KRE framework.

**Table 4:** Comparison of running times.

| Environment | Eureka | SD-KRE | | | | |
|---|---|---|---|---|---|---|
| | | Total Time | Structural Decoupling | Quantitative Assessment | Knowledge Base Maintenance | Extra Expenditure Ratio |
| Ball-Balance | 4 h 33 m 42 s | 4 h 19 m 50 s | 10.84 s | 12.8 ms | 32.1 ms | 0.0698% |
| Quadcopter | 4 h 50 m 49 s | 4 h 41 m 21 s | 12.64 s | 11.2 ms | 38.0 ms | 0.0752% |
| Ant | 3 h 6 m 36 s | 2 h 52 m 14 s | 11.53 s | 11.0 ms | 33.3 ms | 0.112% |
| Humanoid | 7 h 49 m 38 s | 8 h 56 m 4 s | 12.73 s | 11.7 ms | 23.5 ms | 0.0397% |
| Shadow Hand | 7 h 3 m 29 s | 10 h 9 m 27 s | 32.50 s | 24.2 ms | 36.4 ms | 0.0891% |
| Allegro Hand | 8 h 26 m 15 s | 10 h 31 m 16 s | 11.91 s | 10.6 ms | 33.8 ms | 0.0316% |

**Table 5:** Ablation experiment results.

| Task | Ball Balance | Quadcopter | Ant | Humanoid | Shadow Hand | Allegro Hand |
|---|---|---|---|---|---|---|
| SD-KRE | 466.90 ± 2.25 | −0.180 ± 0.0137 | 9.795 ± 0.383 | 8.096 ± 0.523 | 12.25 ± 0.731 | 21.18 ± 1.60 |
| SD-KRE w/o SD | 387.86 ± 3.02 | −0.399 ± 0.0235 | 8.290 ± 0.422 | 6.560 ± 0.393 | 7.83 ± 2.298 | 16.79 ± 1.51 |
| SD-KRE w/o $S_1$ | 400.18 ± 2.35 | −0.262 ± 0.0116 | 9.026 ± 0.110 | 7.164 ± 0.113 | 9.72 ± 0.811 | 18.12 ± 2.18 |
| SD-KRE w/o $S_2$ | 416.26 ± 6.07 | −0.253 ± 0.0183 | 9.061 ± 0.060 | 7.075 ± 0.084 | 10.54 ± 0.839 | 16.79 ± 1.51 |
| SD-KRE w/o $S_3$ | 426.26 ± 8.32 | −0.235 ± 0.0181 | 9.539 ± 0.233 | 7.755 ± 0.312 | 11.30 ± 0.297 | 18.83 ± 1.87 |
| SD-KRE w/o $S_4$ | 432.28 ± 8.92 | −0.225 ± 0.0505 | 9.793 ± 0.561 | 7.407 ± 0.289 | 11.46 ± 0.278 | 20.76 ± 2.40 |
| SD-KRE w/o KRE | 447.72 ± 8.27 | −0.199 ± 0.0374 | 9.499 ± 0.407 | 6.946 ± 0.193 | 11.86 ± 0.401 | 19.92 ± 1.73 |
| Eureka | 391.96 ± 12.35 | −0.385 ± 0.0456 | 8.039 ± 0.393 | 6.410 ± 0.410 | 8.90 ± 0.427 | 15.45 ± 1.66 |

First, structural decoupling is the foundation for performance improvement. The variant SD-KRE w/o SD shows a significant performance drop across all tasks, and its final performance are almost identical to EUREKA in complex dexterous tasks. Without decomposing the reward into independent, reusable structural units, the LLM cannot identify internal reward components, and the optimization degenerates into inefficient black-box search.

Second, multi-dimensional quantitative evaluation is critical for efficient optimization. Among all quantification dimensions, removing the performance correlation dimension (SD-KRE w/o $S_1$) leads to the most severe performance degradation, indicating that performance feedback is the core basis for screening effective reward units. Removing stability ($S_2$), numerical range ($S_3$), or temporal trend ($S_4$) also reduces performance but to a lesser extent, showing that these dimensions further enhance training stability and signal quality.

Third, the structured reward prior knowledge base is essential for knowledge accumulation and reuse. SD-KRE w/o KRE performs worse than the full model, especially in the middle and late training stages. Without cross-iteration knowledge reuse, the method cannot accumulate effective experience, resulting in low sample efficiency and slow convergence. The knowledge base enables directed knowledge transfer and avoids redundant exploration.

In summary, this ablation experiment has quantitatively validated the indispensability of each component within the SD-KRE framework. The decoupling of the reward function structure provides the foundational architecture for optimization, the multi-dimensional quantitative evaluation offers a scientific basis for selection, whilst the structured reward prior knowledge base ensures the accumulation and reuse of

knowledge. The synergistic interaction of these three elements collectively constitutes the SD-KRE's efficient and stable automated reward design capability.

To examine the rationale behind our hyperparameter selections, we adopted a controlled variable approach, systematically altering the key scoring function hyperparameters—the weight vector $\omega$, knowledge base retrieval count $K_{\text{unit}}$, number of trend segments $K$, and target coefficient of variation $CV_{\text{target}}$—to assess their influence on final policy performance (TPS). As shown in Fig. 6, across both environments, the default SD-KRE configuration attained peak performance with a relatively small standard deviation, confirming that the original setup not only maximizes performance but also ensures robust training stability, substantiating the soundness of the initial hyperparameter design.

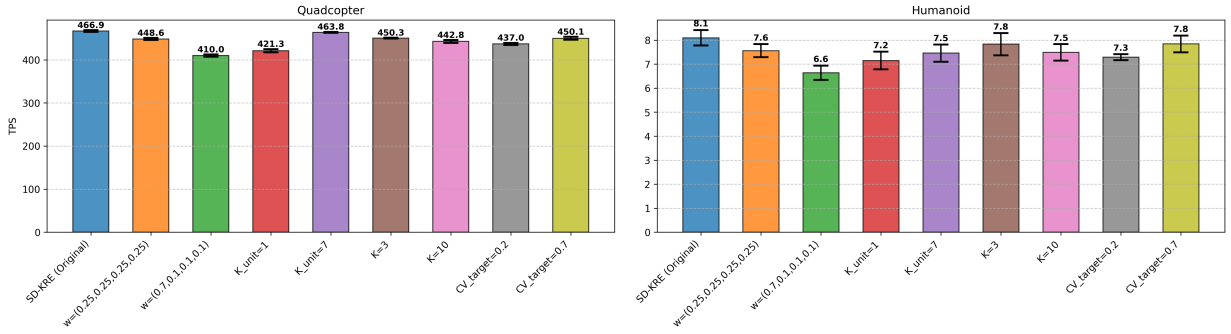


**Figure 6:** Hyperparameter sensitivity analysis of SD-KRE in the quadcopter and humanoid environments.

Assigning uniform weights (0.25, 0.25, 0.25, 0.25) resulted in a marginal performance dip below the original, with the drop staying within acceptable bounds and revealing an inherent fault tolerance in the multi-dimensional scoring mechanism regarding weight distribution. Conversely, extreme skew towards the performance correlation dimension (0.7, 0.1, 0.1, 0.1) triggered a pronounced performance decline: over-amplifying $S_1$ suppresses the discriminative power of stability ($S_2$), value range ($S_3$), and temporal trend ($S_4$), causing erratic noise units to be erroneously promoted into the knowledge base and consequently derailing the LLM's evolutionary trajectory. This outcome strongly supports the original scheme of a dominant weight of 0.4 counterbalanced by equal 0.2 contributions from the remaining dimensions.

In terms of the number of retrieved units, $K_{\text{unit}} = 5$ slightly outperformed $K_{\text{unit}} = 1$ in the Humanoid task, suggesting that complex tasks may benefit from a higher diversity of prior knowledge. However, neither surpassed the original configuration of $K_{\text{unit}} = 3$, indicating that injecting too many low-scoring units dilutes the signal strength of core high-quality patterns. For the number of trend segments, both $K = 3$ and $K = 10$ did not significantly deviate from the original configuration ($K = 5$) in either environment, with performance fluctuations within ±3%, demonstrating the strong robustness of the $S_4$ score to the choice of segmentation granularity.

Lowering $CV_{\text{target}}$ from 0.4 to 0.2 induced a steep performance drop, proving that over-prioritizing stability critically throttles the diversity of viable knowledge base signals. Elevating it to 0.7 offered partial rebound, but the gains remained capped and both alternatives still trailed the 0.4 baseline, establishing 0.4 as the sweet spot harmonizing exploratory width and stability: undue restriction inflicts tangible penalties, while over-loosening confers no material advantage.

In conclusion, SD-KRE demonstrates resilience against moderate hyperparameter drift, whereas extreme or lopsided settings incur substantial performance penalties. The original configuration strikes the most favorable equilibrium between stability and efficacy, affirming the judiciousness of its design.

### *4.4 Analysis of Ablation Experiment Results*

SD-KRE excels at complex continuous control and dexterous manipulation, yet its performance degrades on low-dimensional tasks with tightly coupled reward signals such as CartPole. In CartPole, the single balancing objective yields highly correlated positional, angular and velocity penalties that cannot be split into independent RSUs. Unlike Eureka, which optimizes monolithic reward formulas via global weight tuning, SD-KRE decouples rewards and scores each unit individually without accounting for inter-component synergy. Retrieved high-scoring units often conflict (e.g., prioritizing pole angle over cart position), triggering training oscillations. Additionally, accumulated contradictory units add noise to LLM prompts, while structural parsing and multi-dimensional evaluation bring extra computational overhead and lower sample efficiency. This reveals our method's application boundary: modular knowledge reuse delivers substantial gains for high-dimensional complex tasks, yet its structural overhead may outweigh benefits on simple environments with fully coupled reward objectives.

## 5 Conclusion

This paper addresses three major issues in the current design of reward functions for large language models—poor interpretability, knowledge wastage, and blind optimization—by proposing the SD-KRE method, which involves the structural decoupling of reward functions and the evolutionary reuse of knowledge. This method establishes a comprehensive framework comprising structural decoupling of reward functions, multi-dimensional quantification, and knowledge-reusing evolution, transforming reward design from a holistic, memoryless black-box exploration into a structured, directed white-box optimization. Through systematic experimental validation across six benchmark tasks, SD-KRE comprehensively outperforms the representative baseline method EUREKA and Text2Reward in terms of final performance, convergence speed and training stability, and achieves or exceeds the performance levels of human-designed rewards in multiple tasks. The introduced Correlation metric also confirms the high consistency between the generated rewards and human prior knowledge. Ablation experiments quantitatively validate the indispensability of each core component.

Future work could be pursued in three directions: firstly, expanding the mechanisms for extracting and quantifying structural units to support more complex logical structures, such as introducing control flow elements like conditional branches and loops, thereby enabling structured unit designs to accommodate a wider range of reward function forms; secondly, exploring the cross-task transfer and reuse of structured reward prior knowledge bases to construct a generalized reward prior knowledge base, thereby further enhancing the method's generalization capability and knowledge reuse efficiency by learning common reward patterns across different tasks; Thirdly, integrating lightweight offline evaluation or model prediction to establish a more efficient cycle for rapid reward function evaluation and optimization; for instance, utilizing surrogate models to predict reward function performance, thereby reducing reliance on interactions with the actual environment, which in turn further lowers computational costs and enhances optimization efficiency.

**Acknowledgement:** Not applicable.

**Funding Statement:** This research was funded by the National Natural Science Fund Projects, grant number 62273356.

**Author Contributions:** The authors confirm contribution to the paper as follows: Conceptualization, Yuqing Cao; methodology, Yuqing Cao; software, Yuqing Cao and Jun Lai; validation, Yuqing Cao; formal analysis, Yuqing Cao and Xiliang Chen; data curation, Yuqing Cao and Haoyang Dong; writing—original draft preparation, Yuqing Cao; writing—review and editing, Xiliang Chen, Legui Zhang and Xuefei Sun; visualization, Xiaoyan Wang; supervision, Xiliang Chen and Jun Lai; funding acquisition, Legui Zhang. All authors reviewed and approved the final version of the manuscript.

**Availability of Data and Materials:** The code implementing the methods described in this study is openly available in the GitHub repository: https://github.com/Candacexx/SD-KRE.git. Additional data supporting the findings are available from the corresponding author upon reasonable request.

**Ethics Approval:** Not applicable.

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

## References

1. Sutton R, Barto A. Reinforcement learning: an introduction. 1st ed. Cambridge, MA, USA: MIT Press; 1998.
2. Ng AY, Harada D, Russell S. Policy invariance under reward transformations: theory and application to reward shaping. In: Proceedings of the Sixteenth International Conference on Machine Learning; 1999 Jun 27–30; Bled, Slovenia. p. 278–87.
3. Devlin SM, Kudenko D. Dynamic potential-based reward shaping. In: Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2012); 2012 Jun 4–8; Valencia, Spain. p. 433–40.
4. Everitt T, Krakovna V, Orseau L, Legg S. Reinforcement learning with a corrupted reward channel. In: Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI 2017); 2017 Aug 19–25; Melbourne, Australia. p. 4705–13.
5. Ng AY, Russell S. Algorithms for inverse reinforcement learning. In: Proceedings of the 17th International Conference on Machine Learning (ICML 2000); 2000 Jun 29–Jul 2; Standord, CA, USA. p. 663–70.
6. Ho J, Ermon S. Generative adversarial imitation learning. In: Proceedings of the 30th International Conference on Neural Information Processing System; 2016 Dec 5–10; Barcelona, Spain. p. 4572–80.
7. Christiano P, Leike J, Brown TB, Martic M, Legg S, Amodei D. Deep reinforcement learning from human preferences. In: Proceedings of the 31st International Conference on Neural Information Processing Systems; 2017 Dec 4–9; Long Beach, CA, USA. p. 4302–10.
8. Xie T, Zhao S, Wu CH, Bisk Y, Farhadi A. Text2Reward: reward shaping with language models for reinforcement learning. In: Proceedings of the Twelfth International Conference on Learning Representations (ICLR 2024); 2024 May 7–11; Vienna, Austria.
9. Li H, Yang X, Wang Z, Zhu X, Zhou J, Qiao Y, et al. Auto MC-reward: automated dense reward design with large language models for minecraft. In: Proceedings of the 2014 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR); 2024 Jun 16–20; Seattle, WA, USA. p. 16426–35.
10. Wenhao Y, Gileadi N, Chuyuan F, Kirmani S, Kuang-Huei L, Montse GA, et al. Language to rewards for robotic skill synthesis. In: Proceedings of the 7th Conference on Robot Learning; 2023 Nov 6–9; Atlanta, GA, USA. p. 374–404.
11. Ma YJ, Liang W, Wang G, Huang DA, Bastani O, Jayaraman D, et al. Eureka: human-level reward design via coding large language models. In: Proceedings of the 12th International Conference on Learning Representations (ICLR 2024); 2023 Nov 6–9; Vienna, Austria.
12. Ma YJ, Liang W, Wang HJ, Zhu YK, Fan LX, Bastani O, et al. DrEureka: language model guided sim-to-real transfer. In: Proceedings of the Robotics: Science and Systems; 2024 Jul 15–19; Delft, The Netherlands.
13. Xie G, Xu J, Yang Y, Ding Y, Zhang S. Large language models as efficient reward function searchers for custom-environment multi-objective reinforcement learning. arXiv:2409.02428. 2024.
14. Minae K, Sang MX, Kalesha B, Dorsa S. Reward design with language models. In: Proceedings of the 11th International Conference on Learning Representations (ICLR); 2023 May 1–5: Kigali, Rwanda.
15. Hazra R, Sygkounas A, Persson A, Loutfi A, Zuidberg Dos Martires P. REvolve: reward evolution with large language models using human feedback. In: Proceedings of the 13th International Conference on Learning Representations (ICLR 2025); 2025 May 12–16; Singapore.
16. Nazir MS, Banerjee C. Zero-shot LLMs in human-in-the-loop RL: replacing human feedback for reward shaping. arXiv:2503.22723. 2025.

17. Sun S, Liu R, Lyu J, Yang JW, Zhang L, Li X, et al. A large language model-driven reward design framework via dynamic feedback for reinforcement learning. Knowl-Based Syst. 2025;317:111865. doi:10.1016/j.knosys.2025.111865.
18. Lee H, Phatale S, Mansoor H, Mesnard T, Ferret J, Lu K, et al. RLAIF vs. RLHF: scaling reinforcement learning from human feedback with AI feedback. In: Proceedings of the 41st International Conference on Machine Learning; 2024 Jul 21–27; Vienna, Austria. p. 26874–901.
19. Jason YM, William L, Vaidehi S, Vikash K, Amy Z, Osbert B, et al. LIV: language-image representations and rewards for robotic control. In: Proceedings of the 40th International Conference on Machine Learning (ICML 2023); 2023 Jun 23–29; Honolulu, HI, USA. p. 23301–20.
20. Tan H, Yan H, Yang Y. LLM-guided reinforcement learning: addressing training bottlenecks through policy modulation. arXiv:2505.20671. 2025.
21. Li P, Hao J, Tang H, Yuan Y, Qiao J, Dong Z, et al. R*: efficient reward design via reward structure evolution and parameter alignment optimization with large language models. In: Proceedings of the 42nd International Conference on Machine Learning; 2026 Jan 7–2025 Jul 19; Vancouver, BC, Canada. p. 34509–27.
22. Verma A, Murali V, Singh R, Kohli P, Chaudhuri S. Programmatically interpretable reinforcement learning. In: Proceedings of the 35th International Conference on Machine Learning (ICML 2018); 2018 Jul 10–15; Stockholm, Sweden.
23. Pierre-Yves O. What is intrinsic motivation? A typology of computational approaches. Front Neurorobot. 2007;1(6):6. doi:10.3389/neuro.12.006.2007.
24. Pathak D, Agrawal P, Efros AA, Darrell T. Curiosity-driven exploration by self-supervised prediction. In: Proceedings of the 34th International Conference on Machine Learning (ICML 2017); 2017 Aug 6–11; Sydney, NSW, Australia. p. 2778–87.
25. Adamczyk J, Makarenko V, Arriojas A, Tiomkin S, Kulkarni RV. Bounding the optimal value function in compositional reinforcement learning. In: Proceedings of the 39th Conference on Uncertainty in Artificial Intelligence (UAI 2023); 2023 Jul 31–Aug 4; Pittsburgh, PA, USA. p. 22–32.
26. Gulhane R, Indurthi SR. Beyond monolithic rewards: a hybrid and multi-aspect reward optimization for MLLM alignment. arXiv:2510.05283. 2025.
27. Makoviychuk V, Wawrzyniak L, Guo Y, Lu M, Storey K, Macklin M, et al. Isaac gym: high performance GPU-based physics simulation for robot learning. arXiv:2108.10470. 2021.