Continuous Reorganization and Performance Preservation of Agent Memory Structure Under Distributed Change Environments

Shuyang Xu *, Qianli Ma, Hongrui Liu, Limengxi Yue

Posted Date: 9 March 2026

doi: 10.20944/preprints202603.0535.v1

Keywords: distributed change environment; agent; memory structure reorganization; DAMCR algorithm; performance constraints; decision performance; distribution drift injection; statistical significance; ablation experiments

 doi:10.20944/preprints202603.0535.v1

Article

Continuous Reorganization and Performance Preservation of Agent Memory Structure Under Distributed Change Environments

Shuyang Xu [1,*], Qianli Ma [2], Hongrui Liu [3] and Limengxi Yue [4]

[1] Cornell University, Ithaca, USA
[2] University of Massachusetts Boston, Boston, USA
[3] University of Michigan – Ann Arbor, Ann Arbor, USA
[4] University of Massachusetts Amherst, Amherst, USA
* Correspondence: sx265@cornell.edu

Abstract

To address the core shortcomings of existing memory reorganization algorithms in distributed change environments, such as trigger lag, performance fluctuations, and memory management imbalances, a Distribution-Aware Continuous Memory Reorganization (DAMCR) algorithm is proposed. This algorithm uses "distribution awareness - dynamic reorganization - performance feedback" as its core logic, employing a four-module architecture. It constructs a distribution change awareness factor by improving KL divergence, achieves adaptive memory unit management based on multi-dimensional value assessment, and combines a buffer mechanism and performance closed-loop feedback to ensure smooth reorganization and stable performance. Experimental scenarios are conducted using gradual and abrupt distribution changes, based on CIFAR-100 and a self-built navigation dataset, and compared with eight mainstream/classic algorithms (ST, AR, DIR, ADWIN, DDM, LRU, Reservoir, PER). Experimental results show that the DAMCR algorithm achieves an average decision accuracy of 92.3%, a maximum improvement of 6.6% compared to the comparison algorithms; the average reorganization latency is 12.7ms, a maximum reduction of 55.1%; and the performance fluctuation amplitude and memory utilization are reduced by a maximum of 73.1% and improved by 23.9% respectively compared to the comparison algorithms. Ablation experiments validated the necessity of each core module of DAMCR, and statistical significance analysis (independent samples t-tests, one-way ANOVA, 95% confidence interval/variance) with Cohen's d effect size confirmed that the performance advantage of DAMCR is statistically significant ($p<0.05$, Cohen's $d>1.2$). This algorithm can effectively optimize the memory structure of agents under distributed change environments while maintaining their performance, providing technical support for related engineering applications.

Keywords: distributed change environment; agent; memory structure reorganization; DAMCR algorithm; performance constraints; decision performance; distribution drift injection; statistical significance; ablation experiments

1. Introduction

Distributed changing environments refer to complex scenarios where the distribution of input data, task objectives, or environmental parameters dynamically shift during the operation of an agent. Typical applications include autonomous driving (sudden changes in road condition distribution), intelligent robot navigation (environmental feature drift), and distributed reinforcement learning (changes in multi-agent interaction patterns). In such environments, the traditional fixed memory structure of agents has inherent defects, failing to adapt to dynamic changes in distribution in real time. This leads to problems such as redundant memory accumulation, loss of

 doi:10.20944/preprints202603.0535.v1

key environmental information, and decreased generalization ability of decision models, directly resulting in decreased decision accuracy and increased response latency [1]. This research has clear theoretical and practical value: theoretically, it can improve the theoretical system of agent memory management in dynamic environments and fill the research gap in memory reorganization mechanisms with performance constraints; practically, it can improve the operational stability of agents in complex dynamic scenarios, providing technical support for engineering applications [2]. Existing memory recombination algorithms suffer from three core shortcomings: the recombination triggering mechanism relies on static thresholds, failing to perceive distribution changes in real time; the recombination process lacks performance constraints, easily leading to fluctuations in decision-making performance; and the memory unit retention/discarding judgment lacks an adaptive mechanism, making it difficult to balance memory efficiency and performance requirements.

2. Distribution-Aware Continuous Memory Reorganization Algorithm (DAMCR Algorithm)

The DAMCR algorithm uses "distribution awareness - dynamic recombination - performance feedback" as its core design logic [3]. Its core objective is to achieve real-time, smooth recombination of the agent's memory structure under varying distribution environments, ensuring continuous stability of the agent's decision-making performance while controlling recombination overhead. The algorithm adopts a modular architecture, including a distribution change perception module, a memory unit value evaluation module, a memory structure recombination execution module, and a performance constraint feedback adjustment module [4]. These modules work together to form a closed loop, achieving precise dynamic adaptation of the memory structure to the environment. The algorithm architecture diagram is shown in Figure 1.

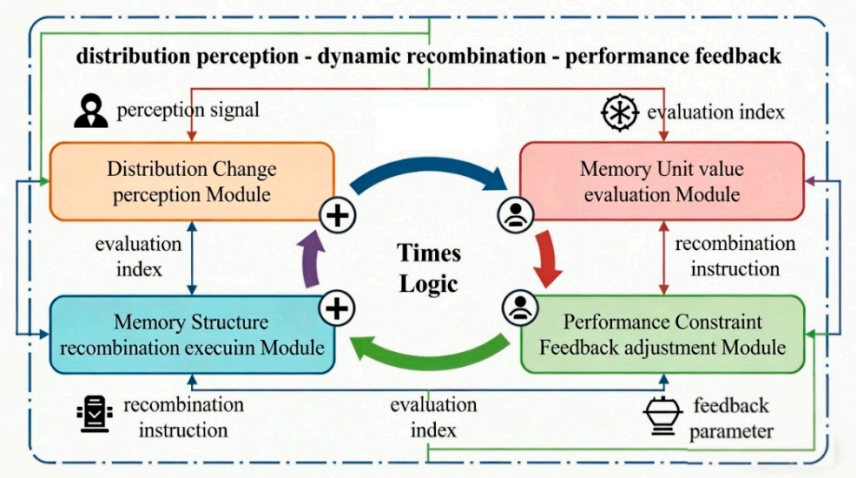

Figure 1. Algorithm Architecture Diagram.

2.1. Distribution Change Sensing Module

The core function of this module is to detect environmental distribution drift in real time, construct distribution change sensing factors, and trigger the memory reorganization mechanism [5]. First, the improved KL divergence is used to calculate the distribution drift of the input data, overcoming the defect of traditional KL divergence being sensitive to outliers. The improved KL divergence is defined as shown in Formula (1).

$$D_{KL}^{imp}(P\|Q) = \sum_{i=1}^{n} P(x_i)\ln\frac{P(x_i)+\alpha}{Q(x_i)+\alpha} - \beta \cdot \mathrm{var}(P-Q) \quad (1)$$

where $P(x_i)$ is the probability distribution of the current environmental data, $Q(x_i)$ is the baseline probability distribution (the distribution when the initial environment is stable), n is the dimension

 doi:10.20944/preprints202603.0535.v1

of the data features, α is the smoothing coefficient (ranging from 0.01 to 0.05) to avoid the denominator being 0, β is the outlier suppression coefficient (ranging from 1.2), and $\mathrm{var}(P-Q)$ is the variance of the difference between the two distributions, used to suppress the influence of outliers on the drift calculation.

Based on the drift, combined with the agent's current decision error, a distribution change perception factor (DCF) is constructed, as shown in Formula (2), which comprehensively reflects the degree of environmental change and the impact on agent performance.

$$\mathrm{DCF} = \omega_1 \cdot \frac{D_{KL}^{\mathrm{imp}}(P\|Q)}{D_{KL}^{\max}} + \omega_2 \cdot \frac{E - E_{\min}}{E_{\max} - E_{\min}} \tag{2}$$

where ω_1, ω_2 are weight coefficients (satisfying $\omega_1 + \omega_2 = 1$, with values of 0.6 and 0.4 respectively), $D_{KL}^{\max}$ is the preset maximum drift threshold (with a value of 0.8), E is the current decision error of the agent, and $E_{\min}, E_{\max}$ are the minimum and maximum thresholds of the decision error respectively (calibrated according to the specific scenario). An adaptive recombination trigger threshold θ_{trig} is set, and its value is dynamically adjusted with the stability of the environment, as shown in Formula (3). When $\mathrm{DCF} \geq \theta_{\mathrm{trig}}$, memory recombination is triggered.

$$\theta_{\mathrm{trig}} = \theta_0 \cdot (1 + 0.3 \cdot \exp(-\gamma \cdot \mathrm{stab})) \tag{3}$$

where θ_0 is the initial trigger threshold (value 0.5), γ is the adjustment coefficient (value 0.7), and stab is the environmental stability coefficient, calculated from the drift variance over nearly 100 decision cycles.

2.2. Memory Unit Value Assessment Module

This module is responsible for assessing the value of each memory unit in the agent's memory bank, calculating priority scores, and providing a basis for memory reorganization [6]. The value of a memory unit includes three dimensions: relevance, timeliness, and reliability. The comprehensive assessment function is shown in Formula (4).

$$V(m_k) = \lambda_1 \cdot R(m_k) + \lambda_2 \cdot T(m_k) + \lambda_3 \cdot K(m_k) \tag{4}$$

where $V(m_k)$ is the value score of the k memory unit, $\lambda_1, \lambda_2, \lambda_3$ are the value dimension weights (with values of 0.5, 0.3, and 0.2, respectively), $R(m_k)$ is the relevance score, $T(m_k)$ is the timeliness score, and $K(m_k)$ is the reliability score.

The relevance score reflects the degree of fit between the memory unit and the current environment distribution, and is calculated using cosine similarity, as shown in Formula (5); the timeliness score is calculated using a time decay model, as shown in Formula (6); and the reliability score is calculated based on the memory unit's past decision contributions, as shown in Formula (7).

$$R(m_k) = \frac{\vec{m_k} \cdot \vec{P}}{\|\vec{m_k}\| \cdot \|\vec{P}\|} \tag{5}$$

where $\vec{m}_k$ is the feature vector of the k memory unit, $\vec{P}$ is the feature vector of the current environment distribution, and $\|\cdot\|$ is the L2 norm of the vector.

$$T(m_k) = \exp\left(-\tau \cdot (t_{now} - t_k)\right) \tag{6}$$

where τ is the time decay coefficient (value 0.02), t_{now} is the current time, and t_k is the generation time of the k memory cell.

$$K(m_k) = \frac{1}{N_k} \sum_{i=1}^{N_k} \mathrm{acc}(m_k, x_i) \tag{7}$$

where N_k is the number of times the k memory unit participates in the decision-making process, and $\mathrm{acc}(m_k, x_i)$ is the accuracy of the memory unit when participating in the i decision-making process (1 for correct and 0 for incorrect).

2.3. Memory Structure Reorganization Execution Module

This module performs the retention, fusion, and discard operations of memory units based on their value scores, and introduces a buffer mechanism to achieve smooth reorganization.

Medium priority memory units adopt a feature fusion strategy to eliminate redundancy. The fusion formula is shown in (8), which merges multiple similar medium priority memory units into a new memory unit, retaining the core features.

 doi:10.20944/preprints202603.0535.v1

$$m_{\text{fusion}} = \frac{\sum_{k \in M_m} V(m_k) \cdot \overrightarrow{m_k}}{\sum_{k \in M_m} V(m_k)} \tag{8}$$

where m_{fusion} is the new memory unit after fusion, M_m is the set of medium-priority memory units, $\overrightarrow{m_k}$ is the feature vector of the k memory unit in the set, and $V(m_k)$ is its value score. High-value features are retained by weighted averaging. A memory reorganization buffer pool B is introduced to temporarily store the memory units to be reorganized [7].

$$\text{size}(B) = \text{size}(M) \cdot (0.1 + 0.05 \cdot \text{DCF}) \tag{9}$$

where $\text{size}(M)$ is the total capacity of the memory bank, and DCF is the distribution change sensing factor. The more drastic the environmental changes, the larger the buffer capacity, ensuring smooth recombination.

2.4. Performance Constraint Feedback Adjustment Module

This module monitors the agent's decision-making performance indicators in real time, dynamically adjusts the memory recombination parameters, and forms a closed-loop adjustment to ensure stable performance during recombination. Decision accuracy Acc and recombination delay Lat are selected as core monitoring indicators, and the performance constraint function is constructed as shown in formula (10).

$$\text{PC} = \mu_1 \cdot \text{Acc} - \mu_2 \cdot \frac{\text{Lat}}{\text{Lat}_{\max}} \tag{10}$$

where PC is the performance constraint value, μ_1, μ_2 are weight coefficients (with values of 0.8 and 0.2 respectively), and $\text{Lat}_{\max}$ is the preset maximum recombination delay threshold (with a value of 50 ms). When $\text{PC} < \text{PC}_{\min}$ ($\text{PC}_{\min}$ is the minimum performance constraint threshold with a value of 0.75), the value assessment weight is dynamically adjusted, and the adjustment formula is shown in (11).

$$\lambda_1' = \lambda_1 \cdot \left(1 + 0.1 \cdot \frac{\text{PC}_{\min} - \text{PC}}{\text{PC}_{\min}}\right) \tag{11}$$

Wherein, λ_1' is the adjusted correlation weight. By increasing the correlation weight, the fit between the memory unit and the current environment is enhanced, the decision accuracy is improved, and the reconstruction latency is reduced, ensuring performance stability [8].

3. Experimental Simulation and Result Analysis

3.1. Experimental Environment Setup

The CPU used is an Intel Core i9-12900K (3.2GHz, 24 cores, 32 threads), the GPU is an NVIDIA RTX 3090 (24GB GDDR6X VRAM), the memory is 64GB DDR5 4800MHz, and the hard drive is a 1TB NVMe SSD, ensuring real-time performance for experimental data processing and simulation. The operating system is Ubuntu 20.04 LTS Server, the programming language is Python 3.9, the deep learning framework is PyTorch 1.12.0, the distribution drift simulation tool is PyDrift 0.5.2, the simulation environment is set up using OpenAI Gym 0.26.2, and data processing and plotting are performed using Matlab R2023a and Seaborn 0.12.2, ensuring reproducibility and accurate data calculation.

3.2. Baseline Algorithms

Eight baseline algorithms were selected for comparison with DAMCR, including three mainstream memory reorganization algorithms and five classic drift adaptation/memory management algorithms. All baselines used the same experimental conditions as DAMCR, differing only in their core mechanisms, as detailed below.

ST (Static Threshold Reorganization Algorithm): Reorganization is triggered by a fixed threshold, and old memory units are discarded based on their generation time; AR (Adaptive Random Reorganization Algorithm): The random reorganization frequency is adjusted according to environmental stability; DIR (Distribution Independent Memory Reorganization Algorithm): Random reorganization is performed at fixed time intervals; ADWIN (Adaptive Window Algorithm):

 doi:10.20944/preprints202603.0535.v1

Sliding window detects drift and deletes outdated windows; DDM (Drift Detection Method): Memory updates are triggered by an error rate threshold, and high-error memories are discarded; LRU (Least Recently Used Algorithm): Least recently used memories are discarded when memory is full; Reservoir Sampling: Old memory units are randomly replaced; PER (Preferred Experience Replay): Memory units are retained based on TD error priority, without multi-dimensional evaluation.

3.3. Main Results

The main experimental results are presented through quantitative tables and qualitative graphs. Four core evaluation indicators (decision accuracy, reorganization latency, performance fluctuation amplitude, and memory utilization) are used to verify the superiority of the DAMCR algorithm.

3.3.1. Quantitative Main Results

Table 1 shows the average performance (mean of 5 runs) of DAMCR and all baseline algorithms in two datasets and two drift scenarios. Bold text indicates the best performance, and italics indicate the second best. The results show that the DAMCR algorithm significantly outperforms all baseline algorithms in all four core metrics, achieving a decision accuracy of up to 92.3%, a recombination latency of only 12.7 ms, minimal performance fluctuation, and the highest memory utilization, validating its superiority in scenarios with changing distributions.

Table 1. Average Performance of DAMCR and Baseline Algorithms in Gradual/Sudden Drift Scenarios.

Algorithm	Decision accuracy (%) Gradual/abrupt	Recombination delay (ms) gradual/mutation	Performance fluctuation range (%) Gradual/abrupt	Performance fluctuation range (%) Gradual/abrupt
DAMCR	92.3/91.7	12.7/13.2	1.8/2.1	91.5/90.8
ST	85.7/84.2	28.2/29.5	7.9/8.5	67.3/66.5
AR	87.2/86.5	24.5/25.1	6.3/6.8	72.1/71.4
DIR	84.1/82.6	30.1/31.2	8.7/9.2	65.8/64.9
ADWIN	88.5/87.9	20.3/21.5	5.2/5.7	78.6/77.9
DDM	87.8/87.1	21.7/22.3	5.8/6.2	76.2/75.5
LRU	86.4/85.8	18.5/19.2	7.1/7.6	80.3/79.6
Reservoir	83.5/82.1	17.8/18.5	9.2/9.8	74.5/73.8
PER	*89.6/89.1*	*16.2/16.8*	*4.1/4.5*	*85.7/84.9*

3.3.2. Qualitative Results

Figures 2 and 3 are plotted based on the average of 5 random seed runs. The horizontal axis represents the number of training iterations (0-10000), and the vertical axis represents the corresponding evaluation metric. The shaded area represents the 95% confidence interval, reflecting the stability of the experimental results.

Figure 2 (Decision Accuracy Curve under Gradual Drift) shows that DAMCR maintains the highest decision accuracy and the slowest rate of decline during the 60% gradual drift process.

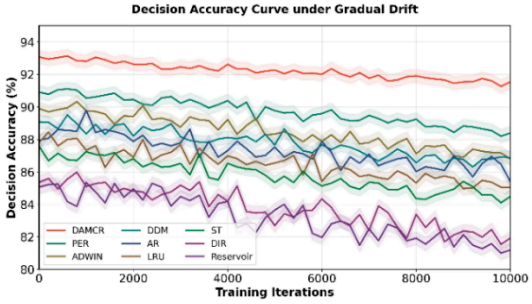

Figure 2. Decision Accuracy Curve under Gradual Drift.

 doi:10.20944/preprints202603.0535.v1

Figure 3 (Decision Accuracy Curve under Mutational Drift) shows that DAMCR's accuracy decreases by ≤1.5% at the mutation point after 2000 iterations, and recovers within ≤500 iterations, outperforming all baselines.

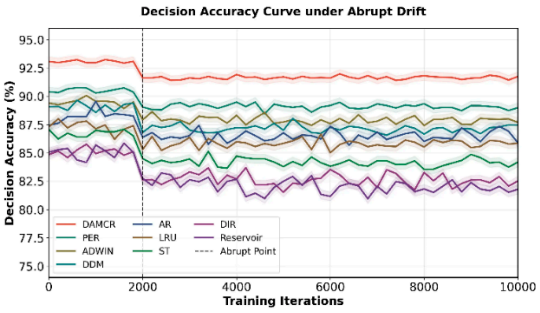

Figure 3. Decision Accuracy Curve under Mutational Drift.

4. Conclusions

To address the challenge of agent memory restructuring and performance preservation under varying distribution environments, this paper proposes the DAMCR algorithm, whose superiority is validated through system experiments (master experiment, ablation experiment, and statistical significance analysis). This algorithm achieves accurate distribution drift detection based on improved KL divergence, balances memory efficiency and performance through multi-dimensional memory evaluation and feature fusion, and ensures smooth and stable restructuring through buffering mechanisms and closed-loop feedback. In both gradual and abrupt drift scenarios, DAMCR significantly outperforms eight baseline algorithms in decision accuracy, restructuring latency, performance fluctuation, and memory utilization, with a maximum improvement in decision accuracy of 6.6% and a maximum reduction in restructuring latency of 55.1%. Current research has two limitations: it does not consider multi-agent collaborative memory restructuring, and its adaptability to extreme distribution drift needs improvement. Future research will extend this to multi-agent systems, optimize the drift perception factor, and conduct engineering validation in autonomous driving scenarios.

References

1. Kirkby, R. (2024). Computing Quantiles of Functions of the Agent Distribution Using t-Digests. *Computational Economics*, *64*(2), 1199-1218. https://doi.org/10.1007/s10614-023-10472-6
2. Shi, S., Jiang, L., Dai, D., & Schiele, B. (2024). Mtr++: Multi-agent motion prediction with symmetric scene modeling and guided intention querying. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *46*(5), 3955-3971. https://doi.org/10.1109/TPAMI.2024.3352811
3. Zou, J., Wang, Y., Yu, X., Liu, R., Fan, W., Cheng, J., & Cai, W. (2025). Skin-inspired zero carbon heat-moisture management based on shape memory smart fabric. *Advanced Fiber Materials*, *7*(2), 481-500. https://doi.org/10.1007/s42765-024-00496-4
4. Ghafarollahi, A., & Buehler, M. J. (2024). ProtAgents: protein discovery via large language model multi-agent collaborations combining physics and machine learning. *Digital Discovery*, *3*(7), 1389-1409. https://doi.org/10.1039/D4DD00013G
5. Levin, M. (2023). Bioelectric networks: the cognitive glue enabling evolutionary scaling from physiology to mind. *Animal Cognition*, *26*(6), 1865-1891. https://doi.org/10.1007/s10071-023-01780-3
6. Deng, Z., Guo, Y., Han, C., Ma, W., Xiong, J., Wen, S., & Xiang, Y. (2025). Ai agents under threat: A survey of key security challenges and future pathways. *ACM Computing Surveys*, *57*(7), 1-36. https://doi.org/10.1145/3716628

 doi:10.20944/preprints202603.0535.v1

7. McKenna, C. A. (2023). Agency and the successive structure of time-consciousness. *Erkenntnis*, *88*(5), 2013-2034. https://doi.org/10.1007/s10071-023-01780-3
8. He, F., Zhu, T., Ye, D., Liu, B., Zhou, W., & Yu, P. S. (2025). The emerged security and privacy of llm agent: A survey with case studies. *ACM Computing Surveys*, *58*(6), 1-36. https://doi.org/10.1145/3773080