IJICIS, Vol.25, No.3, 73-92
DOI: 10.21608/ijicis.2025.411803.1419

International Journal of Intelligent Computing and Information Sciences

https://ijicis.journals.ekb.eg/

RL-BASED FRAGMENT ALLOCATION AND REPLICATION FOR DISTRIBUTED HERITAGE MULTIMEDIA DATABASES

Shymaa H. Mahmoud*

Museums and Archaeology Sites Management Department
Faculty of Archeology, Ain Shams University,
Cairo , Egypt
shymaa.hosny@arch.asu.edu.eg

Ahmed E. Abdelraouf

Information System Department
Faculty of Computer and Information Science,Ain Shams University,
Cairo , Egypt
ahmed_ezzat@cis.asu.edu.eg

Nagwa L. Badr

Information System Department
Faculty of Computer and Information Science,Ain Shams University,
Cairo , Egypt
nagwabadr@cis.asu.edu.eg

Mohamed I.Ali

Museums and Archaeology Sites Management Department
Faculty of Archeology, Ain Shams University,
Cairo, Egypt
mohamed.ibrahim22s@arch.asu.edu.eg

Received 2025-08-07; Revised 2025-08-09; Accepted 2025-08-10

Abstract: *Optimizing fragment allocation and replication in distributed heritage multimedia databases is crucial for minimizing query execution costs in dynamic, resource-constrained environments. Existing heuristics, such as VFAR (Vertical Fragment Allocation and Replication), often neglect inter-fragment dependencies, join relationships, and site capacity limitations, which can result in inefficient allocations. This paper introduces RL-FAWM (Reinforcement Learning-based Fragment Allocation and Replication for Workload-aware Multimedia Systems), a Q-learning framework that models fragment placement as a Markov Decision Process. RL-FAWM incrementally learns effective allocation and replication strategies over multiple episodes using a Q-table, enabling the system to adapt to dynamic workload changes. The approach incorporates structured workload matrices, including read frequency (FRM), manipulation intensity (FMM), and co-access patterns (FCAM), along with inter-site communication costs and strict storage capacity constraints. A cost estimator provides real-time feedback to guide the learning process toward globally optimal and constraint-compliant configurations.Experimental results on a distributed multimedia case study simulating cultural heritage archives demonstrate that RL-FAWM significantly outperforms VFAR, reducing replication costs by over 93% and allocation costs by over 30%. Additionally, RL-FAWM achieves notable reductions in both allocation-stage and replication-stage join costs while effectively preventing site overloading. These results highlight the potential of reinforcement learning for adaptive, scalable, and constraint-aware data management in digital preservation systems.*

Keywords: *Distributed Heritage Multimedia Database, Reinforcement Learning, Q-learning, Cost-based Optimization, Allocation and Replication Optimization.*

***Corresponding Author**: Shymaa H. Mahmoud

Museums and Archaeology Sites Management Department, Faculty of Archeology, Ain Shams University, Cairo, Egypt

Email address: shymaa.hosny@arch.asu.edu.eg

1. Introduction

As the demand for high-quality multimedia applications continues to grow, distributed multimedia databases have become essential for storing and managing vast collections of video, audio, and image data across geographically dispersed servers [1][2][3][4]. These systems support advanced use cases such as online streaming, virtual museums, and multimedia search engines, where both the volume and variety of data demand are scalable and intelligent data management. Effectively accessing such large databases presents major challenges, particularly in data maintenance, consistency, and organization. Without proper fragmentation and optimization strategies, information retrieval from extensive datasets can become slow and inefficient, leading to degraded performance and user dissatisfaction due to delays and limited accessibility.
Moreover, managing and storing these databases requires substantial infrastructure investments, including high-capacity storage, robust communication links, and distributed processing capabilities [5][6][7]. With the proliferation of data-intensive applications ranging from multimedia archives to financial and scientific analytics, fragment allocation has emerged as a central challenge in distributed database system design. Poor allocation strategies can result in unbalanced workloads, increased inter-site communication, and elevated query latency. Historically, static allocation techniques were widely adopted. These methods assumed fixed query patterns and rarely adapted to changes in workload or access frequency. While effective in controlled environments, static methods often fail to maintain performance in real-time and dynamic settings [8].

In the cultural heritage domain, managing multimedia data introduces even greater complexity. This is due to the involvement of diverse institutions, each operating under different mandates, standards, and archival frameworks. Inconsistencies in data documentation, storage formats, and metadata sharing further limit system interoperability and hinder effective data stewardship. These barriers also constrain knowledge dissemination and prevent systems from serving the needs of varied user groups [9][10].

Technological progress has extended the boundaries of cultural interaction beyond traditional settings. As digital technologies have evolved, cultural expression is now embedded across platforms and disciplines, enabling new interpretations and interactions with cultural content. The emergence of the World Wide Web introduced a transformative phase by facilitating web-based cultural dissemination through virtual exhibitions, digital libraries, and interactive museum platforms [11][12].

To support such complex applications, distributed multimedia database systems must be capable of handling dynamic query workloads that often involve large volumes of data, multi-fragment joins, and mixed read/write operations. The strategy used to allocate data fragments across distributed sites directly influences system performance, as it affects query execution time, communication cost, and storage efficiency.

As queries become more diverse and data volumes increase, fragment allocation and replication become critical to ensure low-latency and efficient access. Fragment allocation is the task of determining where to place individual data fragments within the distributed system. Techniques such as the Vertical Fragmentation Allocation Rule (VFAR) [13][14] allocate and replicate fragments statically based on minimizing individual access costs, such as read or manipulation frequency. However, these methods do not account for global performance factors such as fragment co-access patterns, join relationships, and site capacity limitations, making them suboptimal in complex, real-world multimedia environments. This challenge is further compounded in multimedia workloads, where queries frequently involve multi-fragment joins across heterogeneous media types, including video, audio, and subtitles, each with varying size, synchronization demands, and access frequency.

Conventional fragment allocation and replication models typically aim to reduce access latency by analyzing individual read and manipulation patterns. For instance, VFAR and related techniques optimize fragment placement using local statistics such as access frequency and proximity to users [2][13][14]. However, by treating fragments independently, these models overlook inter-fragment relationships and fail to adapt to dynamic, multi-fragment query behavior. This limitation is particularly evident in multimedia contexts, where complex queries may involve correlated data elements, such as matching time-synchronized video and audio, which require joint optimization to achieve efficient execution.

To overcome these limitations, this research introduces a reinforcement learning based fragment allocation framework specifically designed for distributed multimedia database environments. Unlike static heuristics, the proposed approach utilizes Q-learning to dynamically adapt fragment-to-site assignments based on real-time workload feedback.
It minimizes total execution cost by jointly accounting for read, manipulation, and inter-fragment join operations, while strictly enforcing site capacity constraints. Moreover, the model preserves learned allocation policies across episodes, enabling continuous improvement without requiring retraining from scratch. This adaptive capability makes the system particularly well-suited for complex, evolving multimedia workloads typical of heritage information systems.

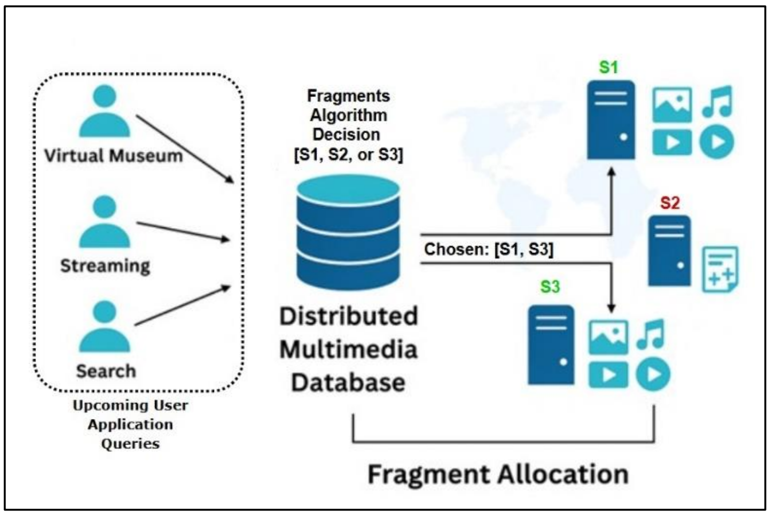

Virtual Museum
Streaming
Search
Upcoming User Application Queries
Fragments Algorithm Decision [S1, S2, or S3]
Distributed Multimedia Database
Chosen: [S1, S3]
S1
S2
S3
Fragment Allocation

Figure 1: Overview of a distributed multimedia database

As shown in Figure 1, the proposed system architecture must support efficient fragment allocation to serve diverse user applications and query patterns in a distributed environment. The figure illustrates a high-level view of a distributed multimedia database system designed to support various user applications such as Virtual Museum, Streaming, and Search. These applications issue queries that access multimedia fragments stored across multiple servers. The figure shows how fragments are allocated across three servers (S1, S2, and S3) based on RL-FAWM (Reinforcement Learning based Fragment Allocation and Replication for Workload-aware Multimedia Systems). In this example, fragments are assigned to servers S1 and S3 to ensure responsive, efficient access that matches the content demands of each application. The figure highlights how user access patterns influence fragment placement in distributed multimedia systems.

The structure of this paper is as follows: Section 2 provides a review of related work; Section 3 details the proposed methodology; Section 4 presents and analyzes the experimental results; and Section 5 Conclusion and Future Work.

2. Related Work

Fragment allocation and replication is a fundamental process in distributed database systems, crucial for improving query execution efficiency, minimizing communication overhead, and balancing storage loads across multiple sites. Over the years, various techniques have been developed to address this challenge. However, most existing approaches assume static workloads, lack runtime adaptability, or overlook cost factors critical to complex and media-rich environments. This section reviews prominent research in this area, with a focus on their allocation strategies and corresponding limitations.

This study [13] proposes an integrated vertical fragmentation, allocation, and replication scheme optimized at design time. Site assignments are decided via cost-based heuristics after fragment formation. Despite achieving initial cost reduction, it does not offer any mechanism for policy adjustment or feedback-based updates once the system is deployed, limiting its applicability under changing query distributions.

The researchers [14] proposed a vertical fragmentation and allocation framework specifically designed for distributed multimedia databases. Their method utilizes an Enhanced CRUD matrix to capture site-specific access behavior and applies a modified Prim's Minimum Spanning Tree algorithm, enhanced with Fibonacci heaps, to efficiently group attributes into fragments. Allocation and replication decisions are guided by communication and storage cost metrics across network nodes. While the approach is effective for static environments and minimizes system cost at design time, it lacks adaptability to runtime workload variations and does not incorporate feedback from actual query execution.

In the work presented in [16] enhanced vertical fragmentation through the Differential Bond Energy Algorithm (DBEA), combining bond energy principles with differential evolution to improve attribute grouping. While this method leads to well-structured fragments and higher affinity measures, it separates allocation from the fragmentation process. The model offers no defined strategy for site assignment, and the allocation phase does not reflect live query behavior, limiting its use in systems that require responsive data placement based on workload feedback.

Introduced research [17] a K-means clustering–based methodology for vertical fragmentation and allocation in distributed database systems. The approach groups queries with similar access patterns and forms fragments, accordingly, assigning them to network sites to improve data locality. Although this technique avoids the need for large access logs and is computationally efficient during system initialization, the allocation remains static once deployed. It does not provide mechanisms for online refinement or respond to changes in access frequency, making it less suitable for dynamic or interactive multimedia systems.

This research proposed a fragmentation method tailored for multimedia databases, integrating content-based query semantics, such as audio and image similarity, into the fragmentation logic. The resulting fragments are allocated using heuristic rules that prioritize semantic relevance and access efficiency. Although this approach is innovative in capturing multimedia content relationships, it does not adapt allocation in response to shifting query patterns or system performance and lacks learning mechanisms that reflect ongoing operational needs.

In [18] present a vertical fragmentation method specifically designed for multimedia databases, where content-based query semantics such as audio and image similarity are integrated into the fragmentation logic. Their approach seeks to improve query performance by aligning fragment structures with content-based access patterns, thus enhancing retrieval efficiency for multimedia queries. However, while the method effectively addresses the fragmentation process, the authors acknowledge that allocating these fragments across distributed sites remains an open challenge. This limitation is especially significant in large-scale or geographically dispersed environments. The proposed method does not incorporate runtime workload adaptation or feedback mechanisms. In their discussion of future work, the authors suggest that integrating adaptive allocation strategies, including those based on machine learning or reinforcement learning, could further optimize system performance by dynamically responding to changing query workloads and system conditions.

As this work [19] introduced RL_Qptimizer, a reinforcement learning–based query optimizer that improves join order selection in centralized relational databases. Their model uses Q-learning and deep Q-networks (DQN) to adjust execution plans based on actual performance feedback, demonstrating dynamic adaptability at the query optimization level. However, their work does not address fragment allocation or data placement across distributed systems. It focuses on plan generation for given data layouts, leaving the challenge of fragment placement, especially in multimedia or geographically dispersed systems, unresolved.

The authors [20] present an adaptive vertical partitioning approach for distributed databases, where a partitioning evaluator serves as the core cost model. This evaluator assesses candidate partitioning schemes by calculating both query I/O cost and migration cost, thereby providing a comprehensive measure of partitioning effectiveness under dynamic workloads. The partitioning evaluator directly guides the system's optimization process, serving as the reward function in the adaptive partitioning algorithm. While this approach effectively adapts to evolving workload patterns and improves query efficiency, it primarily focuses on partitioning at the attribute level and does not address joint optimization for fragment allocation, join costs, or strict site capacity constraints. In contrast, our work introduces a reinforcement learning-based allocation framework that incorporates a richer cost model, explicitly modeling join operations and capacity limitations to enable globally optimal, workload-aware fragment placement.

In summary, while previous studies have advanced fragmentation, allocation and replication in various directions, including clustering, evolutionary search, semantic modeling, and cost-based heuristics, most suffer from common limitations: static decision-making, lack of runtime feedback, absence of join cost modeling, or failure to integrate allocation and replication with system behavior. To overcome these gaps, this research introduces a reinforcement learning based allocation model that dynamically learns fragment placement policies from real query execution. By jointly minimizing read, manipulation, and join costs, while respecting site capacity constraints and maintaining prior learning, the proposed model offers a flexible, cost-aware solution for real-world distributed multimedia environments.

3. RL-FAWM (Reinforcement Learning based Fragment Allocation and Replication for Workload-aware Multimedia Systems)

Figure 2 illustrates the proposed architecture for adaptive fragment allocation and replication in distributed multimedia databases using reinforcement learning (RL). This architecture is designed to minimize execution costs dynamically based on query patterns and system constraints. Each of the layers is described in details in each of the following subsections.

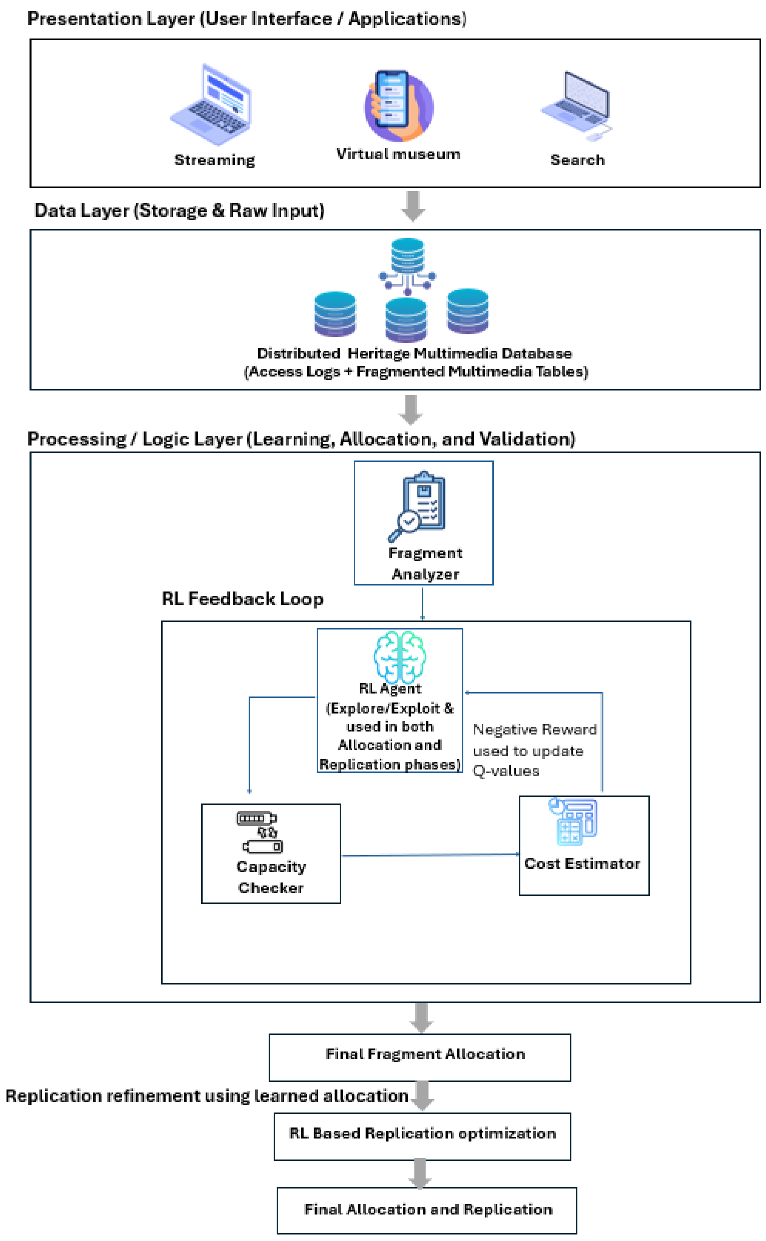

Presentation Layer (User Interface / Applications)
Streaming
Virtual museum
Search
Data Layer (Storage & Raw Input)
Distributed Heritage Multimedia Database
(Access Logs + Fragmented Multimedia Tables)
Processing / Logic Layer (Learning, Allocation, and Validation)
Fragment Analyzer
RL Feedback Loop
RL Agent (Explore/Exploit & used in both Allocation and Replication phases)
Negative Reward used to update Q-values
Capacity Checker
Cost Estimator
Final Fragment Allocation
Replication refinement using learned allocation
RL Based Replication optimization
Final Allocation and Replication

Figure 2 : RL-FAWM -Based Fragment Allocation and Replication Architecture

3.1 Presentation Layer (User Interface / Applications)

This layer represents diverse end-user applications, such as virtual museum platforms, streaming services, and multimedia search engines. These applications generate a wide variety of queries, driving the workload that must be efficiently managed by the underlying data system.

3.2 Data Layer (Storage & Raw Input)

Serving as the central data repository, this component stores fragmented multimedia data such as images, descriptions, and logs of user query access. The database is responsible for supplying both content and historical access information necessary for workload analysis.

3.3 Processing / Logic Layer (Learning, Allocation, and Validation)

This layer contains the entire fragment allocation decision-making pipeline.

3.3.1 Fragment Analyzer

The Fragment Analyzer processes the access logs and existing fragment metadata to generate structured workload matrices. These include the read frequency matrix (FRM), manipulation matrix (FMM), and fragment co-access frequency matrix (FCAM), which together capture usage patterns, modification intensities, and inter-fragment relationships.

3.3.1.1 Read Matrix (FRM)

A synthetic example dataset is used across tables 1 to table 4. Table 1 presents the frequency of read operations from each site to each fragment (FRM). Tables 2 through table 4 complement this by showing manipulation frequencies, co-access patterns, and communication costs, together illustrating the overall system workload.

Table 1 Read matrix

	F1	**F2**	**F3**
S1	3	6	9
S2	6	3	3
S3	9	9	6

FRM (S1, F1) = 3 milliseconds→ Site S1 read fragment F1 three times. For the whole system, total read cost is determined using Eq (1) as [13][14].

$$C_{read} = \sum_{s\in S} \sum_{f\in F} FRM(s,f).CC(s,loc(f)) \qquad (1)$$

Where loc(f) is the site to which fragment f is assigned. For example, if fragment F1 is placed at S2, and site S1 reads it, that access incurs cost = FRM (S1, F1) × CC (S1, S2) = 3 × 6 = 18 milliseconds.

3.3.1.2 Manipulation Matrix (FMM)

Table 2 tracks the frequency of update/write operations (INSERT/UPDATE/DELETE).

Table 2 Manipulation Matrix (FMM)

	F1	F2	F3
S1	2	3	1
S2	3	2	1
S3	1	2	3

FMM (S2, F2) = 2 milliseconds → Site S2 performed two updates to fragment F2. Total manipulation cost calculated with Eq (2) as [13][14].

$$C_{manip} = \sum_{s\in S}\sum_{f\in F} FMM(s,f).CC(s,loc(f)) \qquad (2)$$

3.3.1.3 Co-access Matrix (FCAM)

Reflects how often fragment pairs are joined in the same query can be found in Table 3.

Table 3 Co-access Matrix (FCAM)

	F1–F2	F2–F3	F1–F3
FCAM	15	15	12

FCAM (F1, F2) = 15 milliseconds → Fragments F1 and F2 were joined fifteen times. The total join cost was computed using Eq (3). If two fragments are co-located (loc(fi)∩loc(fj)≠∅), join cost is zero since the join can be executed locally without communication

$$C_{join} = \sum_{(fi,fj)\epsilon FCAM} FCAM(fi,fj)\begin{cases} 0, & \text{if } loc(fi) \cap loc(fj) \neq \emptyset, \\ \min\limits_{sp\in loc(fi),sq\in loc(fj)} . cc(Sp,Sq), & \text{otherwise}, \end{cases} \qquad (3)$$

Consider three fragments: F1 allocated to site S1, F2 allocated to site S2 and F3 allocated to site S3.*Let the communication cost CC (S1, S2) =5, CC (S2, S3) =4, CC (S1, S3) =7milliseconds. Then Pair (F1, F2), 15 ×5=75, Pair (F2, F3),15 ×4=60 and Pair (F1, F3), 12 ×7=84 so Cjoin=75+60+84=219 millisecond, the contribution of this fragment pair to the total join cost is calculated as Eq (3). This value represents the communication cost incurred when the system needs to join these two fragments across their respective sites.*

3.3.1.4 Communication Matrix (CC)

The network or access cost between any two sites [13][14] is provided in Table 4.

Table 4 Communication Matrix (CC)

	S1	S2	S3
S1	0	6	12
S2	6	0	10
S3	12	10	0

CC (S1, S2) = 6 milliseconds → Cost of communication between site S1 and S2 is 6 units.

3.3.2 RL Agent Component

The Reinforcement Learning (RL) agent serves as the core decision-making component in the fragment allocation and replication framework. It formulates the allocation and replication task as a Markov Decision Process (MDP), where each state encodes the complete mapping of fragments to sites and each action corresponds to assigning a specific fragment to a candidate site. The agent receives workload matrices, including read frequency, manipulation intensity and co access patterns, from the Fragment Analyzer to inform its decisions. Its operation integrates two closely interconnected mechanisms: placement optimization and feedback-driven learning. Together, these mechanisms establish the closed feedback loop through which fragment placement decisions are proposed, validated, enacted, and progressively refined.

Firstly, Placement Optimization. The RL agent operates in two stages. In the initial allocation stage, each fragment is assigned to exactly one site, establishing a non-replicated baseline distribution. Once this allocation is complete, the agent proceeds to the replication refinement stage, where actions may introduce new replicas or remove redundant ones. At each iteration, the agent selects a candidate placement action based on the current state. Before execution, the proposed action is validated by the Capacity Checker to ensure that storage constraints are not violated. If the action passes validation, the RL agent enacts the update, modifying the fragment-to-site mapping. This state transition constitutes an environment step, setting the stage for performance evaluation.

Secondly, Feedback-Driven Learning Once an allocation or Replication change is applied, the Cost Estimator evaluates the new configuration and calculates the resulting total execution cost. The adaptive decision-making process is governed by a value-based Q-learning algorithm, which enables the agent to iteratively improve its strategy. The agent maintains a Q table in which each entry estimates the expected long-term reward (the negative of total execution cost) for taking a specific allocation or replication action in a given state. The Cost Model/Estimator computes this reward by combining read, update and join costs with any penalties for capacity violations. During learning, the agent updates its Q-values using the standard Q-learning update formula Bellman Eq 4 [21][22][32].

$$Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma \cdot a' \max Q(s',a') - Q(s,a)] \qquad (4)$$

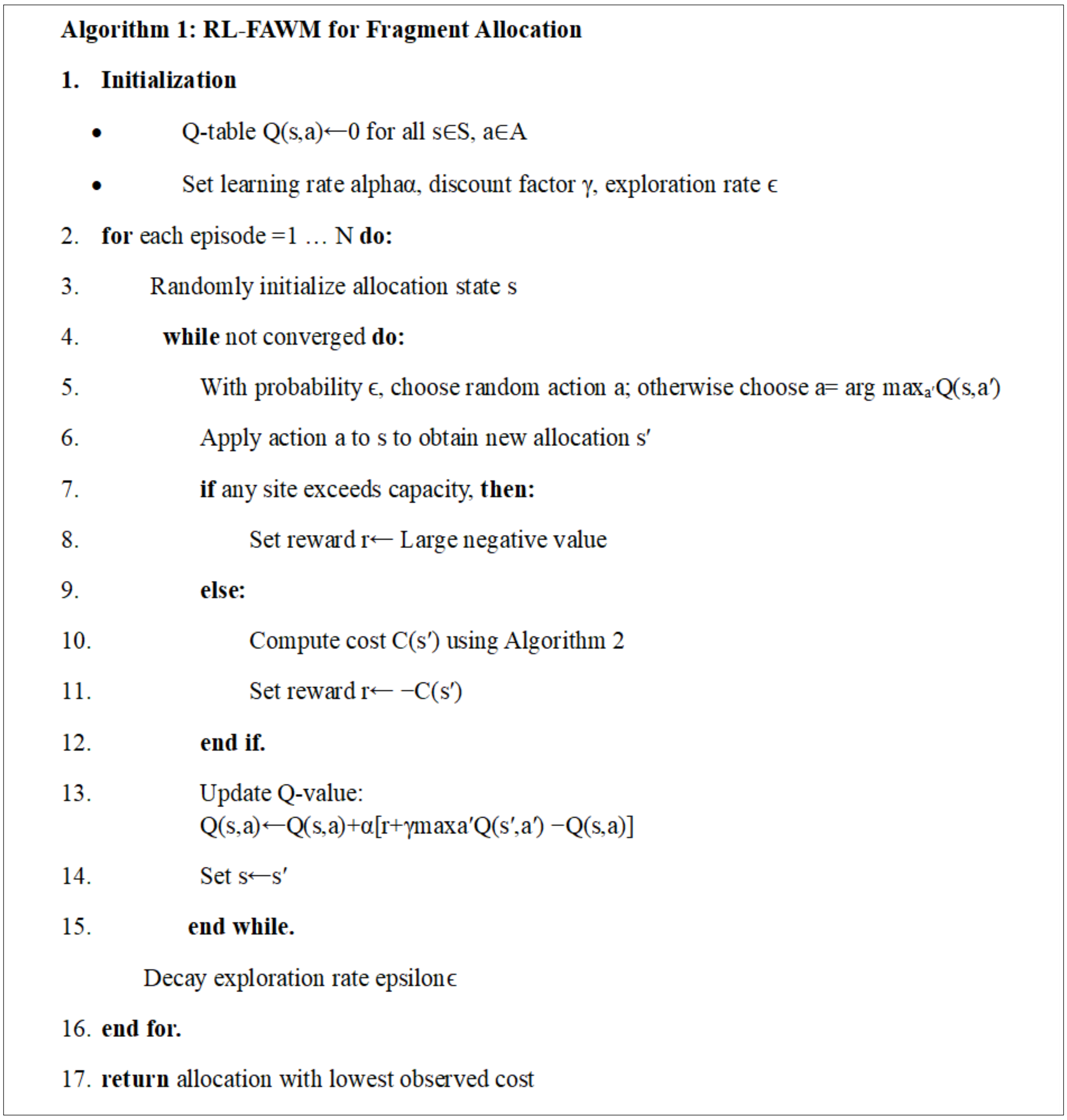
Here, $Q(s,a)$ is the current value of taking action a in state s, alpha α is the learning rate that controls how quickly the agent updates its beliefs, gamma γ is the discount factor for future rewards, r is the immediate reward received (which in this system is the negative total cost), and s' is the resulting new state after taking action a.

To strike a balance between exploring new possibilities and exploiting known good decisions, the agent follows an epsilon-greedy policy. This means that with some probability, it will try a random action to discover potentially better allocations, while most of the time it will choose the action that currently has the highest Q-value. The reinforcement learning cycle can be summarized in Table 5, which details the core steps that govern the agent's interaction with the environment.

Through repeated execution of these steps, the RL agent gradually converges toward fragment placements that minimize execution cost while respecting site constraints. Once training stabilizes, the learned allocation policy is deployed in the distributed multimedia database to support adaptive and cost-efficient fragment management. Figure 3 presents the full Q learning allocation loop in pseudocode form.

Table 5 RL feedback Loop Steps

Step	RL Feedback Loop Component	Description
Step 1	Action Proposal (RL Agent)	The agent selects a new allocation action based on the current state.
Step 2	Capacity Validation (Capacity Checker)	The proposed action is validated against site capacity constraints.
Step 3	Placement Update (RL)	If valid, the action is applied to update the fragment-site mapping.
Step 4	Cost Calculation (Cost Estimator)	The updated allocation is evaluated using the total cost model.
Step 5	Reward Feedback (RL Agent)	The negative reward is used to update the Q-value for the executed action.



Algorithm 1: RL-FAWM for Fragment Allocation

```
1.  Initialization
    •   Q-table $Q(s,a) \leftarrow 0$ for all $s \in S$, $a \in A$
    •   Set learning rate alpha$\alpha$, discount factor $\gamma$, exploration rate $\epsilon$
2.  for each episode =1 … N do:
3.      Randomly initialize allocation state s
4.      while not converged do:
5.          With probability $\epsilon$, choose random action a; otherwise choose a= arg $\max_{a'} Q(s,a')$
6.          Apply action a to s to obtain new allocation $s'$
7.          if any site exceeds capacity, then:
8.              Set reward $r \leftarrow$ Large negative value
9.          else:
10.             Compute cost $C(s')$ using Algorithm 2
11.             Set reward $r \leftarrow -C(s')$
12.         end if.
13.         Update Q-value:
            $Q(s,a) \leftarrow Q(s,a)+\alpha[r+\gamma \max a' Q(s',a') - Q(s,a)]$
14.         Set $s \leftarrow s'$
15.     end while.
        Decay exploration rate epsilon$\epsilon$
16. end for.
17. return allocation with lowest observed cost
```

Figure 3 RL-FAWM

3.3.3 Capacity Checker

The Capacity Checker module enforces storage constraints on each site during the fragment allocation process, as defined in Eq. (5). For every candidate allocation generated by the RL Agent, it calculates the cumulative storage usage of all fragments and their replicas placed on each site:

$$Usage(s)\sum_{f\epsilon Fs} xf,s\ .size(f) \qquad \leq capacity(s) \qquad \forall s \in S \quad (5)$$

Where xf,s 1 if fragment f is placed on site s, otherwise 0.F is the set of all fragments, S the set of sites, size(f) the storage requirement of fragment f, and capacity(s) the maximum storage capacity of site s. If any site is overloaded, the proposed action is immediately rejected and not forwarded to the Cost Estimator.

This proactive validation ensures that the RL Agent explores only feasible and deployable allocation decisions, thereby enhances training stability and accelerates convergence. By discarding invalid states before they reach the Cost Estimator, the Capacity Checker eliminates the inefficiencies of penalty-based approaches that impose costs only after violations occur. Consequently, the system generates more realistic and resilient fragment (and replica) distributions during both training and evaluation phases [21].

3.3.4 Cost Estimator

The Cost Estimator is a central module in the RL-FAWM architecture, responsible for evaluating the quality of each candidate fragment allocation, where allocation may involve replicating fragments across multiple sites. It computes the total execution cost of a given allocation plan, and this value is used as a negative reward for the RL Agent, directly guiding its learning and optimization process. Unlike static models that capture only local costs, the Cost Estimator integrates real workload patterns through structured matrices: the Read Frequency Matrix (FRM), Manipulation Matrix (FMM), Fragment Co-access Matrix (FCAM), and Communication Cost Matrix (CC), as illustrated in Figure 4. The overall execution cost is defined as Eq (6):

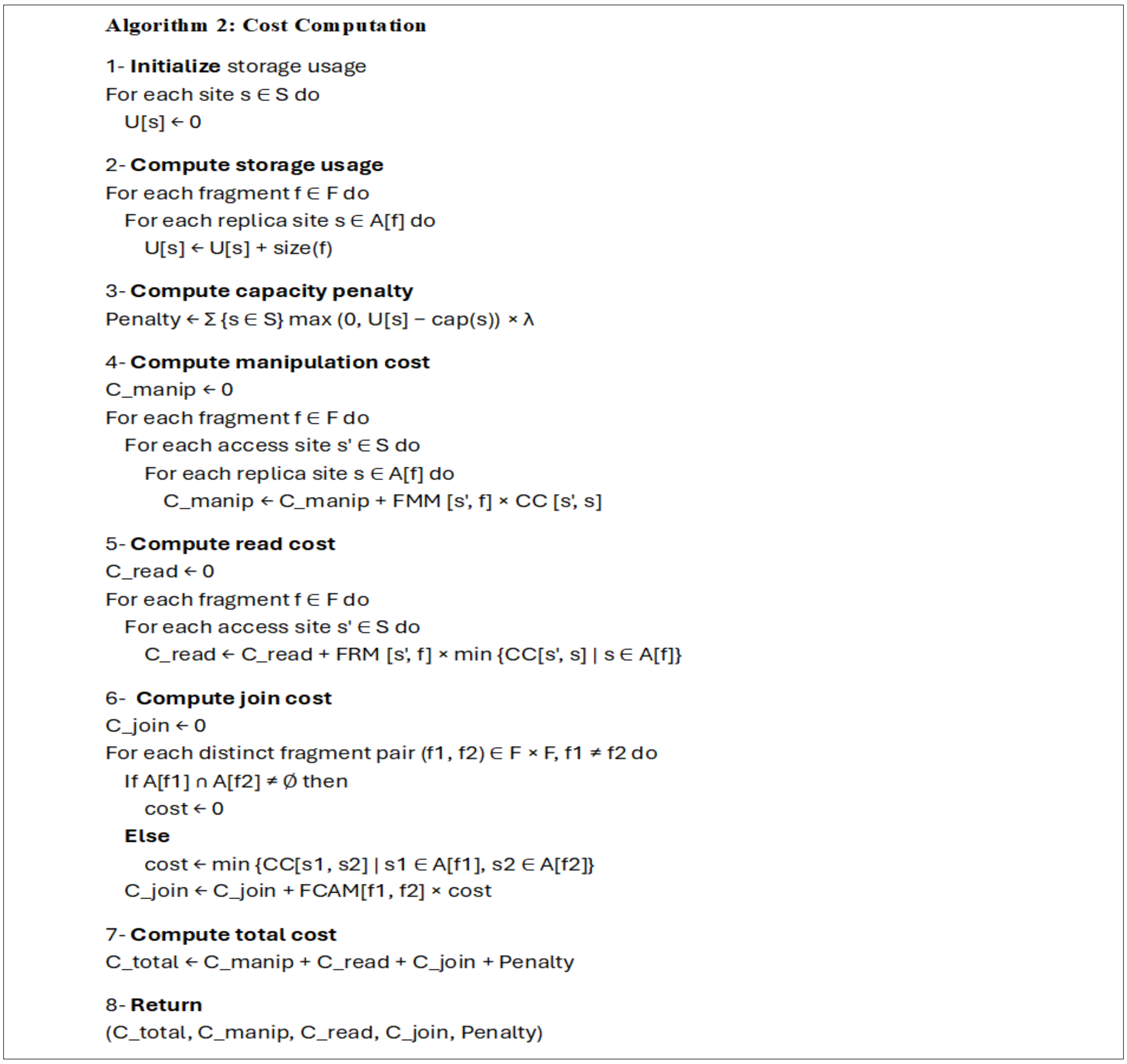
$$C_{total} = C_{read} + C_{manip} + C_{join} + C_{penalty} \qquad (6)$$

Each term is calculated using fragmented access patterns and current assignments. For instance, the read cost for fragment F1 assigned to S1, accessed by S2, as Eq (1). Manipulation cost uses FMM according to Eq (2). In this work, the above cost functions are extended to handle replication by considering multiple replica sites for each fragment. For instance, the read cost is minimized by directing each access to the nearest replica, while manipulation cost propagates updates to all replicas. The join cost sums FCAM frequencies multiplied by communication costs, but only when joined fragments are stored at different sites (3).

To strictly enforce site capacity constraints, a penalty is added when a site is overloaded using Eq (7).

$$C_{penalty} = \sum_{\boldsymbol{S\epsilon Sites}} \boldsymbol{max(0, usage(s) - capacity(s))} \times \boldsymbol{\lambda} \qquad \textbf{(7)}$$

where λ is a scaling factor that strongly discourages invalid allocations. For example, to prevent site overloads 100 milliseconds per MB overflow. If no site is overloaded, the penalty is zero. The total cost output is used as a negative reward during Q-learning.



```
Algorithm 2: Cost Computation
1- Initialize storage usage
For each site s ∈ S do
  U[s] ← 0
2- Compute storage usage
For each fragment f ∈ F do
  For each replica site s ∈ A[f] do
    U[s] ← U[s] + size(f)
3- Compute capacity penalty
Penalty ← Σ {s ∈ S} max (0, U[s] − cap(s)) × λ
4- Compute manipulation cost
C_manip ← 0
For each fragment f ∈ F do
  For each access site s' ∈ S do
    For each replica site s ∈ A[f] do
      C_manip ← C_manip + FMM [s', f] × CC [s', s]
5- Compute read cost
C_read ← 0
For each fragment f ∈ F do
  For each access site s' ∈ S do
    C_read ← C_read + FRM [s', f] × min {CC[s', s] | s ∈ A[f]}
6- Compute join cost
C_join ← 0
For each distinct fragment pair (f1, f2) ∈ F × F, f1 ≠ f2 do
  If A[f1] ∩ A[f2] ≠ Ø then
    cost ← 0
  Else
    cost ← min {CC[s1, s2] | s1 ∈ A[f1], s2 ∈ A[f2]}
  C_join ← C_join + FCAM[f1, f2] × cost
7- Compute total cost
C_total ← C_manip + C_read + C_join + Penalty
8- Return
(C_total, C_manip, C_read, C_join, Penalty)
```

Figure 4 Cost computation

3.4 Final Fragment Allocation

After training converges, the RL agent selects the best-known fragment-to-site assignment observed during exploration. This final allocation minimizes the total execution cost across read, manipulation, and join operations while also adhering to site capacity constraints. In contrast to static approaches such as VFAR, which make one-time decisions without feedback, the RL-based method evolves through continuous interaction with the system. As a result, it produces a more adaptive and cost-efficient allocation strategy that is well suited for dynamic multimedia workloads.

3.5 RL-Based Replication Optimization

After determining the primary allocation of fragments, replication becomes a critical step for improving query performance and data availability in distributed multimedia databases. Replication can reduce read latency, increase fault tolerance, and balance system load. However, if not optimized carefully, replication may violate site capacity limits or introduce unnecessary overhead. This section compares two approaches to replication: a traditional heuristic method that applies greedy rules after allocation, and a reinforcement learning–based strategy that jointly considers total execution cost and capacity constraints.

3.5.1 Background and Conventional Heuristic

In classical distributed database design, the allocation and replication of data fragments are typically treated as two separate and sequential processes. After vertical fragmentation divides relations into column-based groups, each fragment is first assigned to a primary site to minimize manipulation costs. This initial allocation step is usually guided by the Fragment Manipulation Matrix (FMM), which captures the intensity of manipulation operations across sites.

In a later phase, replication is introduced to improve data accessibility and reduce the cost of remote read operations. The Fragment Read Matrix (FRM), which reflects read frequencies at each site, is used to determine suitable locations for placing additional replicas. This two-phase heuristic reflects common practices in much of the classical replication literature [13][14][15].

The heuristic process generally proceeds as follows: for each fragment, the site with the lowest overall manipulation cost is selected as the primary allocation site. This cost is calculated by summing the write frequencies from all sites and the corresponding communication costs to the candidate site. After the primary allocation, replicas are added at other sites where they can most effectively reduce read costs. These decisions are based on total read frequency and communication cost from all sites. The heuristic avoids redundant replication by ensuring that no fragment is replicated at its primary site.

Although this sequential approach simplifies the problem, it often fails to find globally optimal solutions. Allocation decisions are made without considering their impact on future replication, and replication choices do not account for manipulation or join costs. As a result, the final system performance may be suboptimal when compared to methods that integrate allocation and replication into a unified optimization process.

3.5.2 RL-FAWM Based Replication Strategy

RL-FAWM first performs fragment allocation using a Q-learning agent, considering read, manipulation, and join costs. Once an optimal allocation is established, a replication refinement step is applied using reinforcement learning to further reduce system-wide communication overheads and execution costs.

Unlike classical heuristic-based approaches[13][14], this method integrates replication into a unified, cost-aware optimization framework. In this phase, Q-learning is employed to iteratively learn which fragments should be replicated across which sites. The environment is defined as a set of fragment-to-site mappings, where each fragment can reside on one or more sites.

The agent interacts with the system by attempting to add or remove replicas at selected sites for each fragment, using an epsilon-greedy policy to balance exploration and exploitation during replication decisions [20][21][22]. For instance, suppose the agent has previously observed that replicating Fragment F3 on Site S2 yields the lowest system cost and thus has the highest Q-value. With epsilon set to 0.1, the agent will replicate F3 on S2 with a 90% probability (i.e., 1 - epsilon), leveraging its learned knowledge. However, with the remaining 10% probability (epsilon), it explores other options by randomly selecting a different site (such as S1 or S3). This strategy allows the agent to continue discovering potentially better replication configurations while still benefiting from prior successful choices. After each action, the system computes a total cost (as described in Section 3.3.4), which is then used as the reward signal for Q-value updates.

```
Algorithm 3: RL-FAWM – Joint Fragment Allocation and Replication Optimization
Ensure
        • Assignment of each fragment $f \in F$ to one or more sites $S_f \subseteq S$
    1. Initialize
          Q-table for all possible allocation–replication configurations.
    2. For each episode do
          1. use the non-replication allocation as the starting configuration $\{S_f\}_{f \in F}$
          2. For each fragment $f \in F$ do
                1. Action selection: choose action a (add or remove replica of fragment f at site s)
                   using $\epsilon$-greedy policy.
                2. Apply action to obtain new configuration s′
                3. Constraint check:
                ▪ If any site capacity constraint is violated, then
                      ▪ Assign heavy negative penalty reward
                      ▪ Continue to next fragment
                ▪ Else
                      ▪ Compute total system cost $C_{total}$:
                            ▪ Read cost: Served by nearest replica
                            ▪ Manipulation cost: Updates sent to all replicas
                            ▪ Join cost: Joins between fragments on different sites
                            ▪ Penalty: For any capacity overflows
                      ▪ Set reward = -C_total
                4. Q-value update:
                         $Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma a' \max Q(s',a') - Q(s,a)]$
                5. Set $s \leftarrow s'$
                6. End For (fragments)
               7. Decay exploration rate epsilon$\epsilon$
    3. End For (episodes)
    4. Return best fragment–site allocation and replication plan
```

Figure 5 RL-FAWM Replication optimization

Any configuration violating storage capacity constraints receives a significant penalty (see Section 3.3.3), steering the agent away from infeasible states. As shown in Figure 5, the agent begins with a warm-start replica configuration derived from the initial allocation and progressively refines its policy through episodic learning. The search process is directed toward cost-effective and capacity-compliant configurations, where the learned Q-values capture the expected long-term benefit of each allocation or replication action.

This method enables a scalable and adaptive replication mechanism, particularly well-suited for dynamic or large-scale distributed systems. By embedding cost-awareness directly into the learning process and supporting multi-replica configurations, the RL-FAWM -based approach improves overall system performance while remaining flexible for future extensions or real-time adaptation.

4. Results and Discussion

The experimental evaluation of the RL-FAWM model was implemented in Python and executed on an Intel(R) Core (TM) i5-7300U CPU at 2.60GHz. The system was tested using a real-world multimedia dataset from the Alexandria Library Museum, selected as a case study due to its cultural and archival significance.

Bibliotheca Alexandrina Antiquities Museum was established after the discovery of significant artifacts from the Hellenistic, Roman, and Byzantine eras during excavation work. Officially opened in 2002, emphasizing Egypt's multicultural history through modern design and educational programs. Additionally, the museum has contributed 1,324 records to heritage multimedia databases, enhancing access to its collections and promoting cultural awareness both on-site and online, including virtual tours. This section presents the evaluation of the proposed RL-FAWM (Reinforcement Learning based Fragment Allocation and Replication for Workload-aware Heritage Multimedia systems) architecture. The analysis focuses on allocation and replication quality, execution cost reduction, convergence behavior, and constraint enforcement. The system was evaluated in a synthetic, simulated distributed heritage multimedia database environment consisting of three fragments and three sites, with workload behavior modeled using structured matrices (FRM, FMM, FCAM, and CC), shown in table 1,2,3 and 4 respectively.

4.1 Allocation Process Overview

The fragment allocation process begins with the RL Agent (see Section 3.3.2) generating an initial allocation of fragments to sites, using random sampling to ensure broad coverage of the state space. Each allocation is then passed through the Capacity Checker (Section 3.3.3), which ensures that no site exceeds its defined storage constraint. Valid allocations are subsequently evaluated by the Cost Estimator (Section 3.3.4), which computes the total execution cost according to Eq (6), incorporating read, manipulation, join, and penalty costs.

The Q-learning agent maintains a Q-table, where each state-action pair corresponds to a potential fragment-to-site mapping and its estimated long-term value. To balance exploration and exploitation, the agent adopts an epsilon-greedy (ε-greedy) strategy: with probability ε, it explores by randomly selecting a site for a fragment, while with probability $1 - \varepsilon$, it exploits the current Q-table by choosing the site with the highest estimated value. For instance, if ε is set to 0.2, the agent will randomly explore 20% of the time and rely on known best choices 80% of the time. This allows the agent to discover new, potentially better allocations while still refining its learned strategy. The total cost computed for each valid allocation is converted into a negative reward, which is then used to update the Q-values according to the Bellman Equation (4). Through repeated interactions over multiple episodes, the agent incrementally improves its allocation policy, ultimately minimizing total execution cost while respecting storage capacity constraints across all sites.

4.2 Matrix-Based Cost Evaluation

The quality of fragment allocations in RL-FAWM is evaluated using structured workload matrices that reflect real or simulated query behavior. These include the Read Frequency Matrix (FRM) as table 1, Manipulation Frequency Matrix (FMM) as table 2, Fragment Co-access Matrix (FCAM) shown in table 3, and Communication Cost Matrix (CC) shown in table 4. Together, these matrices form the core input to the Cost Estimator (Section 3.3.4), which computes the total execution cost using Eq (5). This cost includes the read, manipulation, and join costs, along with a penalty using Eq (7) for exceeding site capacity.

For each valid fragment-site allocation proposed during Q-learning, the Cost Estimator calculates the total cost, which is then used as a negative reward signal to update the Q-table. This cost-driven feedback enables the RL agent to iteratively refine its policy, learning to prefer allocations that reduce global workload execution cost while satisfying system constraints.

4.2.1 Workload Matrices

The following tables summarize the key data used by the RL agent and the Cost Estimator, Frequency of reads from each site to each fragment as table 1, table 2 Frequency of updates (INSERT/UPDATE/DELETE) from each site to each fragment. Fragment Co-access Matrix (FCAM) Indicates how often fragments were joined together described by table 3, Communication Cost Matrix (CC) Represents the unit cost of transferring data between each site as table 4. These matrices were used by the Cost Estimator (Section 3.3.4) to compute the total execution cost using Eq (6). The resulting cost value was then provided to the RL Agent (Section 3.3.2) as a negative reward during training, guiding the Q-learning process toward more efficient fragment allocation and replication.

4.2.2 Fragment Sizes and Site Capacities

In addition to workload patterns, the system must enforce storage constraints to avoid overloading any site. The Capacity Checker (Section 3.3.3) using Eq (5) uses table 6 and table 7.

Table 6 Fragment Sizes

Fragment	**Size (MB)**
F1	30
F2	30
F3	50

Table 7 Site Capacities

Site	**Capacity (MB)**
S1	55
S2	60
S3	50

Each time the RL agent proposes a new allocation, the total size of all fragments assigned to each site is computed. If this total exceeds the site's defined capacity, the allocation is immediately discarded. By enforcing constraints preemptively, the agent avoids deploying invalid configurations while still recording them in the Q-table, allowing it to learn from both valid and invalid options. This promotes stable convergence and leads to higher-quality final allocations.

4.3 RL vs VFAR: Cost Comparison

This section compares the allocation quality and cost-efficiency of the proposed RL-FAWM framework with the traditional VFAR heuristic [13][14], considering both the read-oriented and manipulation-oriented versions. The evaluation relies on the structured cost model introduced in Section 3.3.4 and computed according to Eq (6)، which aggregates manipulation, read, join, and penalty costs for each allocation strategy.

Table 8 compares the allocation results of the traditional VFAR heuristic against the proposed RL-FAWM approach in a scenario with three fragments and three sites. For each method, the final allocation vector is shown along with the corresponding manipulation, read, join, penalty, and total execution costs:

Table 8 Allocation Results

Method	**Allocation**	**Manip.**	**Read**	**Join**	**Penalty**	**Total**
VFAR Allocation (Read) [13][14]	['S3', 'S3', 'S1']	152	288	324	1000	1764
VFAR Allocation (Manipulation) [13][14]	['S2', 'S1', 'S3']	80	372	390	0	842
RL-FAWM (Total Cost Opt.)	['S2', 'S2', 'S1']	102	324	162	0	588

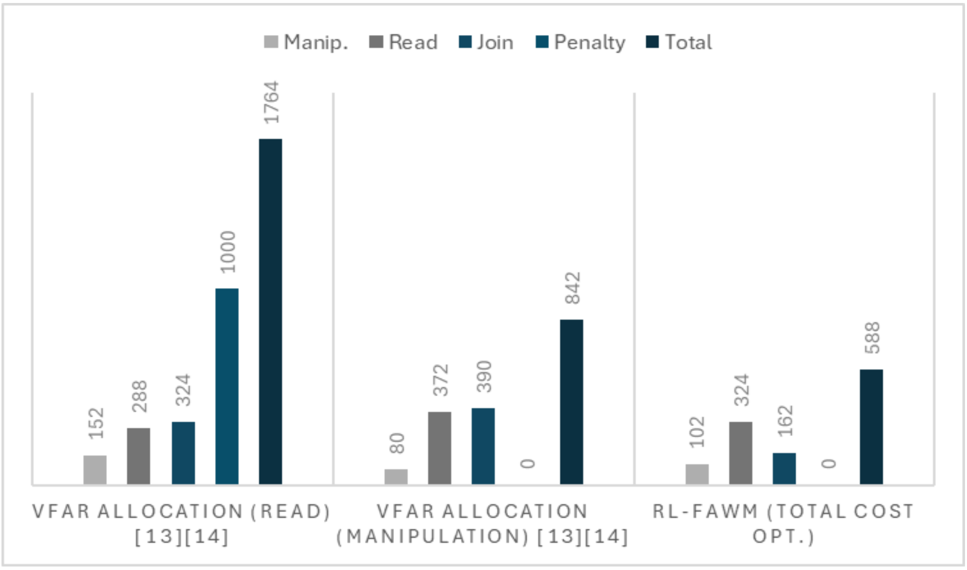


Figure 6 Allocation Result Times(milliseconds)

As shown in Table 8 and Figure 6, the RL-FAWM approach achieves the lowest total execution cost of 588 milliseconds, reducing costs by 30% compared to VFAR (Manipulation) and by 66.7% compared to VFAR (Read). This is accomplished by effectively balancing manipulation, read, and join costs while strictly enforcing site capacity constraints, thereby eliminating penalty costs. In contrast, VFAR (Read) achieves a lower read cost but suffers from substantially higher manipulation, join, and especially penalty costs due to site overloading. VFAR (Manipulation) lowers manipulation cost but at the expense of increased read and join costs.

These results confirm that RL-FAWM delivers a more adaptive and globally optimized allocation than traditional VFAR heuristics, improving system efficiency for distributed heritage multimedia databases under realistic workload and capacity constraints. The performance gap underscores the strength of value-based reinforcement learning for multi-objective optimization in resource-constrained environments, as RL-FAWM learns and leverages trade-offs across multiple cost dimensions rather than optimizing a single metric in isolation.

4.4 Replication Optimization :Comparative Insights

This section provides a comparative analysis of the proposed RL-FAWM replication strategy against the classical VFAR-style heuristic, focusing on total system execution cost. The VFAR method treats replication as a separate step following fragment allocation, without fully integrating cost components across system operations. In contrast, RL-FAWM performs joint optimization using a reinforcement learning framework that holistically evaluates read, manipulation, join, and penalty costs.

Table 9 *Replication Strategies*

Method	Replication Allocation	Manipulation	Read	Join	Penalty	Total Cost
VFAR-style Replication [13][14]	{'S2', 'S3'}, {'S3', 'S1'}, {'S3', 'S1'}	232	54	0	8500	8786
RL-FAWM Replication	[{'S2'}, {'S2'}, {'S3', 'S1'}]	124	252	162	0	538

As shown in Table 9, the RL-FAWM approach reduces the total execution cost from 8786 milliseconds under the VFAR-style replication to just 538 milliseconds, representing a 93.9% improvement. This significant gain is achieved through more efficient resource utilization and a cost-aware replication layout that prevents capacity violations and lowers both manipulation and join costs.

While RL-FAWM exhibits a higher read cost compared to VFAR, this trade-off is intentional and justified. The reinforcement learning model prioritizes global system optimization over local cost minimization. By accepting a moderately increased read cost, RL-FAWM achieves substantial reductions in manipulation, join, and penalty costs, which contribute more heavily to total execution time and system performance degradation. Specifically, the approach reduces manipulation costs by nearly 50% and eliminates penalties caused by site capacity violations, leading to more reliable and balanced resource utilization across all sites.

These findings highlight the effectiveness of RL-FAWM's integrated, cost-sensitive replication strategy. By embedding operational constraints and multiple cost dimensions directly into the learning process, the reinforcement learning model adapts to workload dynamics and capacity limits, offering a scalable, flexible, and robust solution for data replication in distributed multimedia database environments. This makes RL-FAWM particularly suitable for complex, real-world systems where multiple competing factors must be optimized simultaneously.

5. Conclusion and Future Work

This research introduced Reinforcement Learning–based Fragment Allocation and Replication for Workload-aware Multimedia Systems (RL-FAWM), a Q-learning framework designed to optimize the placement of data fragments in heritage multimedia databases. These systems, such as those managing the digital collections of the Bibliotheca Alexandrina Antiquities Museum, require sophisticated strategies to efficiently store and serve complex, high-volume cultural assets.

This research introduced RL-FAWM, a reinforcement learning–driven framework for intelligent fragment allocation and replication in distributed multimedia databases. Unlike traditional heuristics that treat allocation and replication as separate tasks and often overlook global cost optimization, RL-FAWM jointly considers read, manipulation, join, and capacity constraints to adaptively learn effective data placement strategies. By modeling the allocation and replication process as a Markov Decision Process and leveraging structured workload matrices, the RL agent continuously improves system performance under dynamic conditions.

Experimental results demonstrate that RL-FAWM reduced replication costs by over 93% and allocation costs by over 30% compared to the VFAR heuristic, while also lowering both allocation-stage and replication-stage join costs and preventing site overloading. This unified optimization framework offers a robust foundation for scalable, workload-aware data management in complex digital heritage environments. Beyond improving technical efficiency, RL-FAWM supports the sustainable management of digital heritage repositories, enhancing both long-term preservation and user access.

For future work, the RL-FAWM framework can be extended to accommodate larger and more heterogeneous distributed environments, such as national digital libraries or interlinked museum networks. Potential enhancements include adaptive fragment reallocation in response to changing usage patterns, support for sites with varying capabilities (such as different storage or bandwidth limits), and broader validation across multilingual and multicultural datasets to ensure generalizability and scalability.

References

1. Eźechiel KK, Kant S, Agarwal R (2019) A systematic review on distributed databases systems and their techniques. J Theor Appl Inf Technol 96(1)
2. Özsu, M. T., & Valduriez, P. (2020). Principles of Distributed Database Systems (4th ed.). Springer.
3. Li, X., Liu, J., & Wu, S. (2022). "Distributed multimedia databases: Challenges and new directions." ACM Computing Surveys.
4. Shorfuzzaman, M., et al. (2021). "Edge intelligence-enabled multimedia data management: A review." IEEE Access.
5. Mahmoud, Shymaa H., et al. "Optimizing Multi-Media Database Performance: A Comprehensive Review of Fragmentation Methods and Dynamic Access Patterns." 2023 Eleventh International Conference on Intelligent Computing and Information Systems (ICICIS). IEEE, 2023.

6. Siddiqa, Aisha, Ahmad Karim, and Abdullah Gani. "Big data storage technologies: a survey." Frontiers of Information Technology & Electronic Engineering 18 (2017): 1040-1070.
7. Özsu, M. Tamer, and Patrick Valduriez. Principles of distributed database systems. Vol. 2. Englewood Cliffs: Prentice Hall, 1999.
8. Chandra Gope, Dejan. "Dynamic data allocation methods in distributed database system." American Academic & Scholarly Research Journal 4.5 (2012).
9. Fensel D. Ontologies. In: Fensel D, editor. Ontologies: a silver bullet for knowledge management and electronic commerce. Berlin: Springer; 2001. p. 11–8.
10. Ranjgar, Babak, et al. "An ontological data model for points of interest (POI) in a cultural heritage site." Heritage Science 10.1 (2022): 13.
11. Bearman, D.; Trant, J. Interactivity comes of age: Museums and the World Wide Web. Mus. Int. 1999, 51, 20–24. [CrossRef]
12. Poulopoulos, Vassilis, and Manolis Wallace. "Digital technologies and the role of data in cultural heritage: The past, the present, and the future." Big Data and Cognitive Computing 6.3 (2022): 73.
13. Raouf, Ahmed E. Abdel, Nagwa L. Badr, and M. F. Tolba. "An optimized scheme for vertical fragmentation, allocation and replication of a distributed database." 2015 IEEE Seventh International Conference on Intelligent Computing and Information Systems (ICICIS). IEEE, 2015.
14. Sharma, Deepak Kumar, et al. "Modified minimum spanning tree based vertical fragmentation, allocation and replication approach in distributed multimedia databases." Multimedia Tools and Applications 81.26 (2022): 37101-37118.
15. Sharma, Deepak Kumar, et al. "Modified minimum spanning tree based vertical fragmentation, allocation and replication approach in distributed multimedia databases." Multimedia Tools and Applications 81.26 (2022): 37101-37118.
16. Mehta, Shikha, et al. "Differential bond energy algorithm for optimal vertical fragmentation of distributed databases." Journal of King Saud University-Computer and Information Sciences 34.1 (2022): 1466-1471.
17. Amer, Ali A. "On K-means clustering-based approach for DDBSs design." Journal of Big Data 7.1 (2020): 31.
18. 15. Ortiz-Ballona, Aldo Osmar, et al. "A Vertical Fragmentation Method for Multimedia Databases Considering Content-Based Queries." Handbook on Decision Making: Volume 3: Trends and Challenges in Intelligent Decision Support Systems. Cham: Springer International Publishing, 2022. 3-23
19. Ramadan, Mohamed, et al. "Rl_qoptimizer: a reinforcement learning based query optimizer." IEEE Access 10 (2022): 70502-70515.
20. Liu, Pengju, et al. "AVPS: Automatic Vertical Partitioning for Dynamic Workload." International Conference on Intelligent Computing. Singapore: Springer Nature Singapore, 2024.
21. Zhang, T., Liu, Y., Lin, M., & Shahabi, C. (2023). RL-QOptimizer: A Reinforcement Learning-Based Query Optimizer for Distributed Databases. Proceedings of the VLDB Endowment, 16(12), 3451–3464.
22. Sutton, Richard S., and Andrew G. Barto. Reinforcement learning: An introduction. Vol. 1. No. 1. Cambridge: MIT press, 1998.
23. Watkins, Christopher JCH, and Peter Dayan. "Q-learning." Machine learning 8 (1992): 279-292.