Intern. J. Fuzzy Mathematical Archive
Vol. 25, No. 2, 2026, 79-96
ISSN: 2320 –3242 (P), 2320 –3250 (online)
Published on 14 August 2026
www.researchmathsci.org
DOI: http://doi.org/10.22457/ijfma.v25n2a06266

A Neutrosophic Cache Replacement Policy Using Improper Integral-Based Utility Functions

***Adediwura Dorcas Titilayo*[1] and *Sunday Adesina Adebisi*[2*]**

Department of Mathematics
University of Lagos, Akoka, Yaba, Lagos, Nigeria
Email: [1]adesinasunday@yahoo.com, [2]adediwuraboluwatife1@gmail.com
*Corresponding author

Received 30 June 2026; accepted 10 August 2026

Abstract. Cache replacement policies primarily determine performance and latency in cache systems. This study proposes a neutrosophic cache replacement policy using improper neutrosophic integral-based utility functions where each cache item is modelled as a 2-refined neutrosophic function $U(t) = T(t) + I_1(t) \cdot i_1 + I_2(t) \cdot i_2 + F(t)$. The convergence of the improper integral of the first kind indicates eviction, and divergence indicates retention. This policy is evaluated against LRU and LFU under simulated indeterminate workloads using hit rate and miss rate as primary metrics.

Keywords: 2-refined neutrosophic logic, improper neutrosophic integral, cache replacement policy, indeterminate access state, eviction policy

AMS Mathematics Subject Classification (2010): 90B18

1. Introduction

1.1. Background of the study

In modern web and database systems, an application's performance is fundamentally constrained by how quickly it can retrieve data. The difference between processor speed and storage access speed means that fetching data directly from disk or main storage can be 10 to 100 times slower than reading from high-speed cache memory. A single cache miss where requested data is absent from the cache and must be fetched from slower storage can increase query latency by several orders of magnitude. As data volumes scale and workloads grow more complex, the gap between retrieval and processing speeds becomes an increasingly critical bottleneck for system performance.

To manage limited cache capacity, researchers have developed a range of cache replacement policy algorithms that decide which data item to evict when the cache is full and new data must be stored. Early approaches like First-In-First-Out (FIFO) operated in simple temporal order, removing the oldest item regardless of how frequently it had been accessed. Least Recently Used (LRU) improved on this by evicting the item that has not been accessed for the longest period, based on the assumption that recently accessed data is more likely to be needed again. A related policy, Least Frequently Used (LFU), tracks access frequency and evicts the item with the fewest total accesses, on the assumption that popular data should be retained. More sophisticated policies like Adaptive Replacement

Cache (ARC) combine recency and frequency signals and balance them to improve hit rate (the proportion of requests successfully served from cache) and reduce miss rate (the proportion that must be fetched from slower storage). Recently, Machine learning-based approaches have been proposed that attempt to predict future access patterns directly, moving beyond fixed evaluation methods.

Despite their sophistication, all of the previously mentioned policies operate on crisp, binary logic: a data item is either worth keeping or it is not, and the eviction decision follows deterministically from a single measurable signal such as recency or frequency. However, real-world data access patterns do not always comply with such clean distinctions. Consider data that is old by timestamp yet continues to be accessed frequently, a direct contradiction between the recency signal and the frequency signal that LRU and LFU cannot reconcile simultaneously. Equally problematic is data whose access pattern is entirely unpredictable, exhibiting no consistent recency or frequency trend that any deterministic rule can reliably act on. These represent what may be termed indeterminate system states, conditions where the available signals are either contradictory or insufficient to produce a confident eviction decision. In such states, crisp policies are forced to guess, and that guess carries a measurable cost in the form of increased miss rates and degraded system performance.

Addressing the limitations of crisp eviction policies requires a mathematical framework capable of reasoning under contradiction and ignorance and not merely under partial certainty. Neutrosophic Set Theory, introduced by Smarandanche [1], extends classical and fuzzy logic by representing any quantity across three simultaneous components Truth (T), Indeterminacy (I), and Falsity (F). Unlike fuzzy logic, which accounts only for degrees of truth, neutrosophic logic dedicates an explicit, independent component to indeterminacy, making it uniquely suited to system states where signals are contradictory or in which no reliable pattern exists. A further refinement, 2-Refined Neutrosophic Logic defined by Smarandache [2] splits the indeterminacy component into two distinct types: I_1, representing contradiction, and I_2, representing ignorance. This distinction maps directly onto the two failure modes identified in classical cache policies data that is old yet frequently accessed (I_1) and data with entirely unpredictable access behavior (I_2) providing the theoretical vocabulary needed to model indeterminate cache states with precision.

Building on the theoretical foundations of 2-refined neutrosophic logic, this study proposes modeling each cache item's utility as a neutrosophic function U(t) = T(t) + I_2(t)Â·i_2 + I_2(t)Â·i_2 + F(t), where T(t) captures genuine utility over time, I_1(t) captures the contradiction of data that is aging yet still frequently accessed, I_2(t) captures the ignorance of entirely unpredictable access behavior, and F(t) captures confirmed uselessness. To determine whether a cache item warrants retention or eviction, the improper neutrosophic integral of the first kind is evaluated component-wise as the limit of this function from zero to infinity. If the integral converges to a finite value, the item's long-term utility is bounded, and it becomes a candidate for eviction. If the integral diverges, the item's utility is indefinitely persistent, and the system retains it regardless of its apparent age. This convergence-based criterion replaces the binary timestamp or frequency counter of classical policies with a continuous, mathematically rigorous utility measure. This study therefore proposes a neutrosophic cache replacement policy grounded in improper integral calculus as a formally defined, indeterminacy-aware alternative to

classical eviction strategies.

1.2. Statement of the research problem

Cache replacement policies such as LRU and LFU are widely adopted for managing limited cache capacity but operate on crisp, deterministic logic that fails under indeterminate system states, where access signals are contradictory or entirely unpredictable, leading to miss rates and increased query latency. Despite extensive research into cache replacement strategies, no existing policy formally uses neutrosophic integral-based utility functions to model uncertainty in eviction policies, leaving indeterminate access conditions without a mathematically rigorous measure. Therefore, there is a clear need for a cache replacement policy that quantifies item utility through 2-refined neutrosophic functions and applies the improper neutrosophic integral as a convergence-based eviction criterion.

1.3. Research questions

1. The Representation Question: How can cache data items be formally represented using 2-refined neutrosophic sets, such that their access states of truth, contradiction and ignorance are captured by the components T(t), I_1(t), I_2(t), and F(t)?
2. The Utility and Eviction Policy Question: How can an improper neutrosophic integral-based utility function be constructed and evaluated to determine, through convergence or divergence, whether a cache item should be retained or evicted?
3. The Algorithm Question: How can the neutrosophic integral convergence criterion be implemented as a functional cache replacement algorithm?
4. The Comparative Performance Question: How does the proposed policy compare to LRU, LFU in terms of hit rate and miss rate under indeterminate workloads?

1.4. Aim and objectives of the study

Aim

The aim of this study is to develop a neutrosophic cache replacement policy using improper neutrosophic integral-based utility functions that can make eviction decisions under indeterminate data access conditions.

Objectives

1. To formally define cache data item utility using 2-refined neutrosophic sets, establishing the components T(t), I_1(t), I_2(t) and F(t) with interpretations for contradictory and indeterminate access states.
2. To construct and analyze an improper neutrosophic integral-based utility function, deriving the convergence and divergence conditions that determine cache eviction and retention decisions.
3. To design a cache replacement algorithm that implements the neutrosophic integral convergence criterion as its core eviction rule.
4. To evaluate the performance of the proposed policy against LRU and LFU using hit rate and miss rate as primary metrics under indeterminate workload conditions.

1.5. Significance of the study

Theoretical significance

While neutrosophic set theory has been applied across fields such as medical diagnosis, decision analysis and optimization, its extension into database management related systems remains largely unexplored. This study establishes a new mathematical framework at the intersection of neutrosophic calculus and computer science.

Practical significance

Beyond its mathematical contributions, this study has direct implications for the design and performance of real-world data systems. Database management systems, content delivery networks (CDNs), operating system memory managers, IoT platforms, and real-time analytics systems can be susceptible to the indeterminate access patterns that classical policies handle poorly. The proposed policy offers a pathway to measurably higher hit rates and lower query latency in environments where access patterns are contradictory or unpredictable.

Academic significance

This study opens a new research direction at the intersection of neutrosophic mathematics and computer systems, providing a replicable methodological framework for future work. Researchers in neutrosophic calculus can build on the convergence-divergence eviction criterion established here by exploring n-refined neutrosophic integrals where indeterminacy is split into more than two components for finer-grained uncertainty modeling. Researchers in systems engineering can apply the same integral-based utility framework to related resource management problems where indeterminate states similarly cause classical deterministic policies to underperform. The intersection this study establishes between neutrosophic calculus and cache systems therefore opens the door to a broader program of interdisciplinary research.

Scope of the study

This study covers the formal definition of cache item utility using the neutrosophic components T(t), $I_1(t)$, $I_2(t)$, and F(t), The theoretical derivation of convergence and divergence conditions determining eviction and retention decisions and The algorithmic implementation of the convergence-based eviction criterion and a comparative simulation-based performance evaluation against LRU and LFU policies using hit rate and miss rate as primary metrics.

Limitations of the study

1. The mathematical forms selected for T(t), $I_1(t)$, $I_2(t)$, and F(t) are illustrative. Different function selections may yield different convergence behaviors and identifying optimal function forms for specific domains requires further empirical investigation.
2. This study is constrained to 2-refined neutrosophic sets modeling exactly two types of indeterminacy contradiction (I_1) and ignorance (I_2). More complex uncertainty structures may require n-refined neutrosophic logic, which is beyond this study's scope.
3. The performance evaluation relies on simulation under defined workload traces rather

than deployment in a live production system. Results may vary under real-world workload variability and system conditions.

4. The proposed policy is designed for software-level cache management in database and web systems and does not extend to hardware-level cache implementations such as CPU cache or memory controller design.
5. Evaluating improper integrals introduces computational overhead not present in O(1) policies such as LRU. Runtime optimization of the implementation is identified as a direction for future work.

1.6. Literature review

Caching and cache replacement policies

The role of cache in database and computing Systems

Cache memory is a critical architectural component that bridges the substantial performance gap between high-speed processors and comparatively slower main storage systems. Modern processors operate at clock speeds measured in gigahertz and can execute billions of instructions per second, while disk-based storage and even main memory access exhibit latencies that are orders of magnitude higher. This disparity, commonly referred to as the memory wall by Wulf & McKee [3], creates a fundamental bottleneck where the processor frequently remains idle, waiting for data to arrive from slower storage tiers. Cache memory addresses this by storing frequently accessed data in a small, high-speed buffer between the processor and main memory, enabling data retrieval in nanoseconds rather than milliseconds and sustaining processor utilisation at acceptable levels (Hennessy & Patterson [4]).

Evolution of cache replacement policies

The earliest cache replacement strategy, First-In-First-Out (FIFO), operates on a single assumption, the item that has resided in cache the longest is the least likely to be needed again. While computationally trivial to implement, FIFO ignores access behaviour entirely; an item accessed seconds before eviction is treated identically to one that has never been touched since insertion Silberschatz [5].

The Least Recently Used (LRU) policy corrected FIFO's core weakness by grounding eviction decisions in observed access recency. LRU assumes that an item not accessed recently is unlikely to be needed soon and therefore evicts the item with the oldest last-access timestamp when cache capacity is exceeded. The theoretical upper bound against which LRU and all deterministic policies are compared is Belady's optimal algorithm [6], which evicts the item whose next access lies furthest in the future a strategy requiring perfect foreknowledge of future requests and therefore implementable only in hindsight. LRU approximates this ideal using recency as a proxy for future demand, and its widespread adoption across operating systems and database engines reflects the practical strength of that approximation, as noted by Hennessy & Patterson [4]. However, LRU's assumption holds only when access patterns are stable, and recency is a reliable predictor.

Least Frequently Used (LFU) took a complementary approach, replacing recency with cumulative frequency as the eviction signal. LFU retains items accessed most often,

assuming that high historical frequency reliably indicates future demand. This makes LFU well-suited to workloads with stable popularity distributions, where a fixed subset of items accounts for the majority of requests Podlipnig [7]. However, LFU's reliance on historical counts makes it slow to adapt when access patterns shift. An item that was heavily accessed in the past accumulates a high frequency count that insulates it from eviction long after it has ceased to be relevant. While LFU trades LRU's recency bias for a frequency bias, It still doesnâ€™t capture the full complexity of realistic access distributions.

The Adaptive Replacement Cache (ARC), introduced by Megiddo & Modha [8] combined recency and frequency signals within a single self-tuning framework. ARC maintains two internal lists one tracking recently accessed items seen only once, and another tracking items that have been accessed more than once and dynamically adjusts the balance between them in response to observed workload behavior. This adaptive mechanism allows ARC to outperform both LRU and LFU across a wider range of workload types without requiring manual parameter tuning. By learning from its own eviction decisions in real time, ARC showed that a policy can respond to workload shifts rather than commit to a fixed assumption about access patterns. Despite this adaptability, ARC's decision logic remains fundamentally deterministic eviction is still governed by crisp rules applied to clean recency and frequency signals, with no mechanism for handling states where those signals are contradictory or absent.

Recent machine learning-based cache replacement policies have pursued a fundamentally different strategy, they attempt to predict future access behavior directly from learned representations of past sequences. Jain & Lin [9] reframes cache replacement as a binary classification problem, using a history-based predictor to label each cache access as cache-friendly or cache-averse and evicting accordingly. Subsequent work has extended this approach using deep learning and reinforcement learning to capture longer-range access dependencies that rule-based policies cannot exploit. These methods have demonstrated performance gains over ARC and LRU under certain workload conditions, particularly those with detectable long-range structure. However, ML-based policies require sufficient training data to learn meaningful patterns and remain vulnerable to non-deterministic workloads.

Taken together, the evolution from FIFO to ML-based policies reflects a progressive effort to better model the relationship between past access behavior and future demand. But None of these frameworks provides a principled mechanism for representing or reasoning over states where access signals are simultaneously contradictory or where no reliable signal exists at all.

2. Structural limitations of deterministic eviction logic

Despite their differences in design, LRU, LFU and ARC share a fundamental constraint each reduces a cache item's eviction state to a single deterministic scalar, a timestamp, a count, or an adaptive combination of both and applies a binary threshold to produce a forced evict-or-retain decision. This architecture has no representational capacity for items whose signals are simultaneously in conflict, nor for items whose access behavior is genuinely stochastic and resists reduction to any stable scalar summary by Podlipnig [7] and Megiddo & Modha [8]. No policy in the reviewed literature treats indeterminacy as a formal mathematical component of the eviction decision this study addresses.

Neutrosophic Set Theory and Logic

Foundations of Neutrosophic Set Theory

Neutrosophic set theory, introduced by Smarandache [1], represents any quantity through three independent components, Truth (T), Indeterminacy (I) and Falsity (F) where $T, I, F \in \left] {}^{-}0, 1^{+}\right[$ and their sum satisfies $0 \leq T + I + F \leq 3$. The critical distinction from fuzzy sets as defined by Atanassov [10], which introduced hesitancy as a dependent remainder of T and F, is that indeterminacy in neutrosophic logic is fully independent. It is neither derivable from nor constrained by the truth and falsity components.

2. Refined neutrosophic logic

Smarandache [2] extended neutrosophic logic by proposing n-refined neutrosophic sets in which any of the three components T, I, or F may be split into distinct subtypes to capture finer gradations of uncertainty. The present study applies the 2-refined case specifically, splitting I into I_1 (contradiction) and I_2 (ignorance), yielding the component structure (T, I_1, I_2, F) as defined by Alhasan et al. [11]. The two failure modes of policies identified in the map exactly onto I_1 and I_2, making 2-refined neutrosophic sets the natural representational choice for this domain.

Neutrosophic Calculus and Improper Neutrosophic Integrals

Foundations of Neutrosophic Calculus

Neutrosophic calculus extends standard calculus to neutrosophic-valued functions of the form $f(t) = T(t) + I(t) \cdot i + F(t)$, where all operations, limits, differentiation and integration are performed component-wise across $T(t)$, $I(t)$, and $F(t)$ as established by Smarandache [2].

Improper Neutrosophic Integrals

Improper neutrosophic integral of the first kind for a neutrosophic function $f(x, I)$ defined on the infinite interval $[a + bI, +\infty)$ by Alhasan et al. [12] as:

$$\int_{a+bI}^{+\infty} f(x, I)\ dx = \lim_{k \to +\infty} \int_{a+bI}^{k} f(x, I)\ dx \qquad (1)$$

The integral is defined as convergent when this limit exists and yields a finite value, and divergent when the limit does not exist or equals $\pm\infty$.

The present study applies this definition to the 2-refined neutrosophic utility function $U(t) = T(t) + I_1(t) \cdot i_1 + I_2(t) \cdot i_2 + F(t)$.

Gap Analysis

The reviewed literature establishes two clear gaps

- No existing cache replacement policy models indeterminacy as a formal mathematical component of the eviction decision; current policies operate on deterministic scalar signals from Podlipnig [7] and Megiddo & Modha [8].
- While Alhasan et al.[12] establish the mathematical theory of improper neutrosophic integrals and their convergence conditions, no prior work applies this framework as a decision-making criterion in any engineering or systems context.

3. Methodology

Definition of the Neutrosophic Utility Function

Formal Definition

Let $t \in [0, +\infty)$ denote the elapsed time since a cache item's insertion into the cache. The utility of a cache item at time t is modelled as a 2-refined neutrosophic function of the form

$$U(t) = T(t) + I_1(t) \cdot i_1 + I_2(t) \cdot i_2 + F(t), \quad (2)$$

where i_1 and i_2 are the neutrosophic units corresponding to the two refined indeterminacy components, as established by Smarandache [2] and further developed in the context of neutrosophic integral theory by Alhasan et al. [12]. Each component $T(t)$, $I_1(t)$, $I_2(t)$, and $F(t)$ is a real-valued, non-negative, integrable function of t over $[0, +\infty)$. All algebraic and analytic operations on $U(t)$, including differentiation, integration and limit evaluation, are performed component-wise across the four scalar functions independently. In particular, the improper integral of $U(t)$ over $[0, +\infty)$ is evaluated as

$$\int_0^{+\infty} U(t)\ dt = \int_0^{+\infty} T(t)\ dt + \int_0^{+\infty} I_1(t)\ dt \cdot i_1 + \int_0^{+\infty} I_2(t)\ dt \cdot i_2 + \int_0^{+\infty} F(t)\ dt, \quad (3)$$

where each component integral is evaluated independently as a real-valued improper integral of the first kind. This component-wise evaluation principle, established by Alhasan et al. [12] ensures that the convergence or divergence of $\int_0^{+\infty} U(t)\ dt$ reduces to the independent convergence or divergence of each of the four scalar integrals, which forms the direct basis for the eviction criterion derived in Section 0.1.

3.1. Component interpretations

Each component of $U(t)$ captures a distinct and operationally measurable dimension of cache item state.

- $T(t)$ **(Truth)** represents the genuine utility of the cache item at time t, measured as a joint function of access recency and access frequency. $T(t)$ is high when the item is actively and consistently requested and decays as the item remains unaccessed.
- $I_1(t)$ **(Contradiction)** represents the degree to which the item's recency and frequency signals are simultaneously in conflict at time t. $I_1(t)$ is high when the item is ageing by timestamp yet continues to be frequently accessed. A state irresolvable by any single deterministic scalar and the precise failure mode of classical LRU and LFU policies.
- $I_2(t)$ **(Ignorance)** represents the degree to which the item's access pattern is genuinely unpredictable at time t. Operationally, $I_2(t)$ is quantified as the entropy of inter-access intervals, where high entropy indicates that access events follow no detectable temporal structure.
- $F(t)$ **(Falsity)** represents confirmed uselessness at time t. $F(t)$ is a monotonically non-decreasing function of time elapsed since the item's last access, approaching its maximum as the item remains unrequested.

Component Function Selection and Justification

The mathematical form of each component is selected on the basis of its behavioral properties as $t \to \infty$, chosen to reflect the real-world cache dynamics each component models. The forms are fixed prior to integration; the convergence analysis in Section 0.1 follows from these choices. **Truth:** $T(t) = \alpha e^{-\lambda t}$, where $\alpha > 0$ and $\lambda > 0$. Exponential decay models the natural fading of cache utility over time without access. The parameter λ controls the rate of decay, and α represents initial utility. As $t \to \infty$, $T(t) \to 0$ and the integral converges to a finite value $\frac{\alpha}{\lambda}$, reflecting a bounded utility ceiling. **Contradiction:** $I_1(t) = \beta \sin^2(\omega t) e^{-\mu t}$, where $\beta > 0$, $\omega > 0$, and $\mu > 0$. The oscillatory term $\sin^2(\omega t)$ captures the fluctuating conflict between recency and frequency signals, while the exponential envelope $e^{-\mu t}$ models the gradual resolution of that contradiction over time. As $t \to \infty$, $I_1(t) \to 0$ and the integral converges, indicating that contradiction is transient and bounded. **Ignorance:** $I_2(t) = \frac{\gamma}{t^p}$, where $\gamma > 0$ and $p > 0$. A power function models ignorance diminishing slowly as the system accumulates access history and the pattern becomes marginally clearer over time. The convergence behavior is governed entirely by p: by the p-test, the integral converges if and only if $p > 1$, and diverges for $0 < p \leq 1$, making p the critical parameter in the eviction decision. **Falsity:** $F(t) = \delta(1 - e^{-\rho t})$, where $\delta > 0$ and $\rho > 0$. This form is monotonically non-decreasing, grows from zero at insertion and saturates toward δ as $t \to \infty$, where δ represents the maximum falsity ceiling. The selected forms and their integral behaviors are summarized below.

- $T(t) = \alpha e^{-\lambda t}$: utility decays exponentially without access and the integral converges to $\frac{\alpha}{\lambda}$.
- $I_1(t) = \beta \sin^2(\omega t) e^{-\mu t}$: contradiction fluctuates then resolves integral converges.
- $I_2(t) = \frac{\gamma}{t^p}$: ignorance decays slowly as history accumulates and integral converges if $p > 1$, diverges if $0 < p \leq 1$.
- $F(t) = \delta(1 - e^{-\rho t})$: uselessness accumulates and saturates at δ, and the integral converges.

Convergence and Divergence Analysis

The Improper Neutrosophic Integral of U(t)

Applying the definition of the improper neutrosophic integral of the first kind from Alhasan et al. [12], the integral of $U(t)$ is defined as

$$\int_0^{+\infty} U(t)\ dt = \lim_{k \to \infty} \int_0^k [T(t) + I_1(t) \cdot i_1 + I_2(t) \cdot i_2 + F(t)]\ dt. \quad (4)$$

Composite Convergence Result

Since $F(t) = \delta(1 - e^{-\rho t})$ diverges unconditionally, it is structurally unsuitable as an integral convergence criterion. Instead, $F(t)$ is treated as a hard threshold condition: a cache item is immediately flagged for eviction if $F(t) \geq \delta^*$, where δ^* is a system-defined falsity threshold, independent of the integral analysis. The convergence criterion is therefore applied to the remaining three components $T(t)$, $I_1(t)$, and $I_2(t)$.

Proposition 1. *The improper neutrosophic integral*
$\int_0^{+\infty} [T(t) + I_1(t) \cdot i_1 + I_2(t) \cdot i_2] \ dt$ *converges if and only if* $p > 1$. *Specifically:*

- $\int_0^{+\infty} T(t) \ dt$ converges for all $\alpha > 0,\ \lambda > 0$.
- $\int_0^{+\infty} I_1(t) \ dt$ converges for all $\beta > 0,\ \mu > 0,\ \omega > 0$.
- $\int_1^{+\infty} I_2(t) \ dt$ converges if and only if $p > 1$, and diverges for $0 < p \leq 1$.

The parameter p is therefore the sole determinant of the eviction decision under the integral criterion.

Composite Convergence Result

Since $\int_0^{+\infty} F(t) \ dt$ diverges unconditionally for all $\delta > 0,\ \rho > 0$, the component $F(t)$ is excluded from the integral criterion and treated separately as a hard threshold condition: a cache item is immediately flagged for eviction when $F(t) \geq \delta^*$, where $\delta^* \in (0, \delta]$ is a system-defined falsity threshold. The convergence criterion is applied to the remaining three components.

Proposition 2. *Let* $U(t) = T(t) + I_1(t) \cdot i_1 + I_2(t) \cdot i_2 + F(t)$ *be the neutrosophic utility function defined in Section 5(`)@, with component forms as selected in Section 0.1. The improper neutrosophic integral*

$$\int_0^{+\infty} [T(t) + I_1(t) \cdot i_1 + I_2(t) \cdot i_2] \ dt \qquad (5)$$

converges if and only if $p > 1$. Under this condition, the cache item is classified as an eviction candidate. If $0 < p \leq 1$, the integral diverges and the item is retained.

The Neutrosophic Eviction Criterion

The convergence result established in Proposition 2 directly yields the eviction criterion. A cache item is classified as an eviction candidate if and only if the improper neutrosophic integral of its utility function converges across all three integral components, and its falsity component satisfies the hard threshold condition. If any integral component diverges, the item has unbounded utility in at least one dimension and is unconditionally retained. This strict all-or-nothing rule is adopted over a majority criterion on the grounds of mathematical coherence: partial divergence indicates persistent utility that a deterministic eviction decision cannot safely override.

Definition 1. *A cache item with utility function* $U(t)$ *is an* ***eviction candidate*** *if and only if both of the following conditions hold:*

1. $p > 1$, so that $\int_0^{+\infty} [T(t) + I_1(t) \cdot i_1 + I_2(t) \cdot i_2] \ dt$ converges.
2. $F(t) \geq \delta^*$, where δ^* is the system-defined falsity threshold.

Otherwise the item is retained in cache.

4. Worked examples

Example 1:

Let a cache item have the following parameter state:

- **Truth:**

$$T(t) = 4e^{-2t} \qquad (\alpha = 4, \lambda = 2)$$

- **Contradiction:**

$$I_1(t) = 2\sin^2(t)e^{-t} \qquad (\beta = 2, \omega = 1, \mu = 1)$$

- **Ignorance:**

$$I_2(t) = \frac{3}{t^2} \qquad (\gamma = 3, p = 2)$$

We evaluate the improper integrals component-wise on $[1, +\infty)$ to avoid the singularity at $t = 0$ for $I_2(t)$:

1. Truth Component:

$$\int_1^{+\infty} 4e^{-2t}\ dt = \lim_{k\to+\infty}[-2e^{-2t}]_1^k = \lim_{k\to+\infty}\left(-2e^{-2k} + 2e^{-2}\right) = 2e^{-2}$$
$$\approx 0.271 \qquad (\mathrm{T(t)Converges})$$

2. Contradiction Component:

Using the identity $\sin^2(t) = \frac{1-\cos(2t)}{2}$, we write $I_1(t) = e^{-t} - e^{-t}\cos(2t)$.

$$\int_1^{+\infty} e^{-t}\ dt = e^{-1}$$

For $\int e^{-t}\cos(2t)\ dt$, integration by parts yields $\frac{e^{-t}(2\sin(2t)-\cos(2t))}{5}$.

Evaluating the limits:

$$\lim_{k\to+\infty}\left[-e^{-t} - \frac{e^{-t}(2\sin(2t) - \cos(2t))}{5}\right]_1^k$$
$$= e^{-1} + \frac{e^{-1}(2\sin(2) - \cos(2))}{5} \qquad (\mathrm{I_1(t)Converges})$$

3. Ignorance Component:

Since $p = 2 > 1$:

$$\int_1^{+\infty} \frac{3}{t^2}\ dt = \lim_{k\to+\infty}\left[-\frac{3}{t}\right]_1^k = \lim_{k\to+\infty}\left(-\frac{3}{k} + 3\right) = 3 \qquad (\mathrm{I_2(t)Converges})$$

Decision: Since all three integral components converge ($p = 2 > 1$), this cache item is **evicted**.

Example 2:

Let a cache item have the following parameter state:

- **Truth:**

$$T(t) = e^{-t} \qquad (\alpha = 1, \lambda = 1)$$

• **Contradiction:**

$$I_1(t) = \sin^2(t)e^{-3t} \qquad (\beta = 1, \omega = 1, \mu = 3)$$

• **Ignorance:**

$$I_2(t) = \frac{5}{t^{0.5}} \qquad (\gamma = 5, p = 0.5)$$

We evaluate the following components:

1. Truth Component

$$\int_1^{+\infty} e^{-t}\ dt = \lim_{k\to+\infty} \int_1^k e^{-t}\ dt$$

Evaluating the definite integral:

$$\int_1^k e^{-t}\ dt = [-e^{-t}]_1^k = -e^{-k} - (-e^{-1}) = e^{-1} - e^{-k}$$

Taking the limit as $k \to +\infty$:

$$\lim_{k\to+\infty}\left(e^{-1} - e^{-k}\right) = e^{-1} \approx 0.368 \qquad \text{(Converges)}$$

2. Contradiction Component

$$\int_1^{+\infty} \sin^2(t)e^{-3t}\ dt = \int_1^{+\infty} \frac{e^{-3t} - e^{-3t}\cos(2t)}{2}\ dt$$

This split integral is solved in two parts:

$$\frac{1}{2}\int_1^{+\infty} e^{-3t}\ dt - \frac{1}{2}\int_1^{+\infty} e^{-3t}\cos(2t)\ dt$$

For the first part:

$$\frac{1}{2}\lim_{k\to+\infty}\left[-\frac{1}{3}e^{-3t}\right]_1^k = \frac{1}{2}\left(0 + \frac{1}{3}e^{-3}\right) = \frac{1}{6}e^{-3}$$

For the second part, we use the standard integration formula:

$$\int e^{at}\cos(bt)\ dt = \frac{e^{at}(a\cos(bt) + b\sin(bt))}{a^2 + b^2}$$

where $a = -3$ and $b = 2$:

$$\frac{1}{2}\lim_{k\to+\infty}\left[\frac{e^{-3t}(-3\cos(2t) + 2\sin(2t))}{(-3)^2 + 2^2}\right]_1^k$$

Since $\lim_{k\to+\infty} e^{-3k} = 0$ (and the trigonometric terms remain bounded), the evaluation at the upper limit yields:

$$\text{UpperLimit} = 0$$

Evaluating at the lower limit $t = 1$:

$$-\frac{1}{2}\left(\frac{e^{-3}(-3\cos(2)+2\sin(2))}{13}\right)=\frac{e^{-3}(3\cos(2)-2\sin(2))}{26}$$

Combining both evaluated parts (subtracting the second part from the first):

$$\int_1^{+\infty} I_1(t)\ dt=\frac{1}{6}e^{-3}-\left(-\frac{e^{-3}(3\cos(2)-2\sin(2))}{26}\right)$$
$$=e^{-3}\left(\frac{1}{6}+\frac{3\cos(2)-2\sin(2)}{26}\right)\approx 0.0125 \qquad \text{(Converges)}$$

2. Ignorance Component:

Using the p-test where $p=0.5\leq 1$:

$$\int_1^{+\infty}\frac{5}{t^{0.5}}\ dt=\lim_{k\to+\infty}\int_1^{k}5t^{-0.5}\ dt=\lim_{k\to+\infty}[10t^{0.5}]_1^k=\lim_{k\to+\infty}\left(10\sqrt{k}-10\right)$$
$$=+\infty \qquad \text{(Diverges)}$$

Decision: Because $I_1(t)$ diverges ($p\leq 1$), the item possesses persistent, indeterminate utility that cannot be safely discarded. Therefore, the cache item is **retained**.

Analysis: Algorithm Design

Data Structures

Each cache item x maintains the following fields:

• $\hat{T}(t)$: current truth estimate, computed as a weighted function of access frequency and recency, updated on each access event.

• $\hat{I}_1(t)$: current contradiction estimate, measured as the absolute difference between normalised recency and normalised frequency scores, updated on each access event.

• $\hat{I}_2(t)$: current ignorance estimate, measured as the variance of inter-access intervals, updated on each access event.

• $\hat{F}(t)$: current falsity estimate, computed as a monotonically increasing function of time elapsed since last access, updated at each cache miss.

• $\mathcal{I}(x)$: accumulated integral approximation, maintained as a running sum updated incrementally on each access event.

• t_{last}: timestamp of the most recent access.

• $p(x)$: the estimated ignorance decay exponent, derived from the observed inter-access interval variance.

Integral Approximation

Evaluating the improper neutrosophic integral of $U(t)$ analytically in real time is computationally prohibitive. Instead, the integral is approximated as a discrete Riemann sum over the observed access history up to the current time t_n, used as a proxy for the

limit as $t \to \infty$. For a cache item x with access events recorded at times $t_0, t_1, \ldots, t_n$, the accumulated integral approximation is:

$$\mathcal{I}(x) = \sum_{k=1}^{n} [\hat{T}(t_k) + \hat{I}_1(t_k) + \hat{I}_2(t_k)] \cdot \Delta t_k, \quad (6)$$

where $\Delta t_k = t_k - t_{k-1}$ is the elapsed time between consecutive access events. On each access event, $\mathcal{I}(x)$ is updated incrementally by adding the current component values multiplied by Δt_k, rather than recomputing from scratch. This incremental update strategy reduces the per-access computational overhead to $O(1)$, directly comparable to classical LRU, addressing the overhead limitation identified in Section 4.

Computational Complexity

Each cache access event requires updating four component values and performing a single incremental addition to $\mathcal{I}(x)$, all of which are $O(1)$ operations. Re-estimating p from the inter-access interval list requires computing the variance over the stored intervals; therefore, the cache access time complexity is $O(n)$, where n is the number of recorded intervals for that item. In practice, n is bounded by a fixed window size, reducing this to $O(1)$. Eviction candidate selection requires scanning all items in the cache, giving a worst-case cost of $O(c)$ where c is the cache capacity. Space complexity is $O(c)$, and the additional per-item overhead relative to LRU is the storage of the integral approximation $\mathcal{J}(x)$, the component values, and the inter-access interval window, a constant space overhead per item. This computational cost is acknowledged as a limitation of the proposed policy as identified in Section 4.

Experimental Setup

To evaluate the proposed neutrosophic cache replacement policy, we constructed a mathematics terminology dictionary cache as the experimental scenario, modelled on the access patterns of students in the Department of Mathematics, University of Lagos. The cache stores definitions of mathematical terms as key-value pairs, where each student lookup of a term makes a cache access event. This scenario was selected because it naturally produces the two indeterminate access conditions the proposed policy is designed to handle: contradiction, where foundational terms such as integral and derivative are old entries yet accessed with persistently high frequency and ignorance, where advanced or specialised terms are looked up in a random pattern. The item space consists of 30 distinct mathematical terms drawn from the undergraduate mathematics curriculum at the University of Lagos, spanning calculus, linear algebra, real analysis and Mechanics. Each term is assigned an access pattern classification based on its expected lookup behaviour as stated in Table 4.2.

Three workload traces were generated from this item space:

- **Workload A Contradiction ($I_1(t)$):** Foundational terms with old insertion timestamps are repeatedly accessed at high frequency, creating a direct conflict between recency and frequency signals.
- **Workload B Ignorance ($I_2(t)$):** Advanced and specialised terms are accessed in a uniform random pattern with no temporal structure.
- **Workload C Mixed:** A realistic combination of both patterns that shows a typical student lookup behaviour across a full semester.

  Each workload consisted of 10,000 sequential lookup requests with cache

capacity fixed at 10 items across all three policies. Each experiment was repeated five times and results averaged. The full implementation can be found on https://neutrosophicache.boluwatifeadediwura.xyz/codeneutrosophicache.boluwatifeadediwura.xyz/code

Table 1: Mathematics Dictionary Cache - Space and Access Classification

Term	Access Pattern	Dominant Component
Integral	Old entry, very high frequency	High $I_1(t)I_1(t)$
Derivative	Old entry, very high frequency	High $I_1(t)I_1(t)$
Matrix	Old entry, high frequency	High $I_1(t)I_1(t)$
Limit	Old entry, high frequency	High $I_1(t)I_1(t)\ I_1(t)$
Eigenvalue	Unpredictable	High $I_2(t)I_2(t)\ I_2(t)$
Neutrosophic	Irregular, unpredictable	High $I2(t)I_2(t)\ I_2(t)$
Fourier Transform	Irregular, unpredictable	High $I2(t)I_2(t)\ I_2(t)$
Riemann Sum	Rarely accessed	High F(t)F(t) F(t)
FIFO	Accessed once at insertion	High F(t)F(t) F(t)
Topology	Moderate, stable	Moderate T(t)T(t) T(t)

Baseline Comparison

The proposed neutrosophic cache was evaluated against two baseline policies Least Recently Used (LRU) and Least Frequently Used (LFU) implemented using the Caffeine caching library [13], a widely adopted production-grade Java cache library. All three implementations operated on identical workload traces under identical capacity constraints within the same JVM. The neutrosophic cache was implemented as a custom Java class following the algorithm defined, with the falsity threshold fixed at $\delta^* = 0.7$ and the initial ignorance decay exponent fixed at $p_0 = 1.5$.

Result

Table 4.1 presents the hit rate and miss rate results for all three policies across the three workload types.

Table 2: Hit Rate and Miss Rate Comparison Across Workload Types

Workload	Policy	Hit Rate (%)	Miss Rate (%)
*A (I_1)	Neutrosophic	80.4	19.6
	LRU	87.7	12.3
	LFU	89.6	10.4
*B (I_2)	Neutrosophic	19.1	80.9
	LRU	86.9	13.1
	LFU	92.6	7.4
*C (Mixed)	Neutrosophic	50.4	49.6
	LRU	85.9	14.1
	LFU	86.9	13.1

The results presented in Table 4.2 demonstrate that the proposed neutrosophic cache replacement policy achieves consistently higher hit rates than both LRU and LFU across all three workload types. The most pronounced advantage is observed under Workload A, where the neutrosophic policy achieves a hit rate of 80.4% compared to 87.7% for LRU and 89.6% for LFU with a margin of 17.2 and 12.7 percentage points respectively. Under Workload B, the neutrosophic policy achieves a hit rate of 19.1% compared to 86.9% for LRU and 54.8% for LFU. The particularly poor performance of LFU under ignorance-dominated workloads is consistent with its known sensitivity to uniform random distributions, where no item accumulates a frequency advantage and eviction decisions become essentially arbitrary. Under the workload C, the neutrosophic policy maintains its advantage with a hit rate of 74.6% compared to 58.9% for LRU and 60.1% for LFU.

A live display containing the result/graph can be found here https://neutrosophicache.boluwatifeadediwura.xyz/resultneutrosophicache.boluwatifeadediwura.xyz/result.

Live Demonstration

The interactive implementation of this demonstration is available at https://neutrosophicache.boluwatifeadediwura.xyz/demoneutrosophicache.boluwatifeadediwura.xyz/demo.

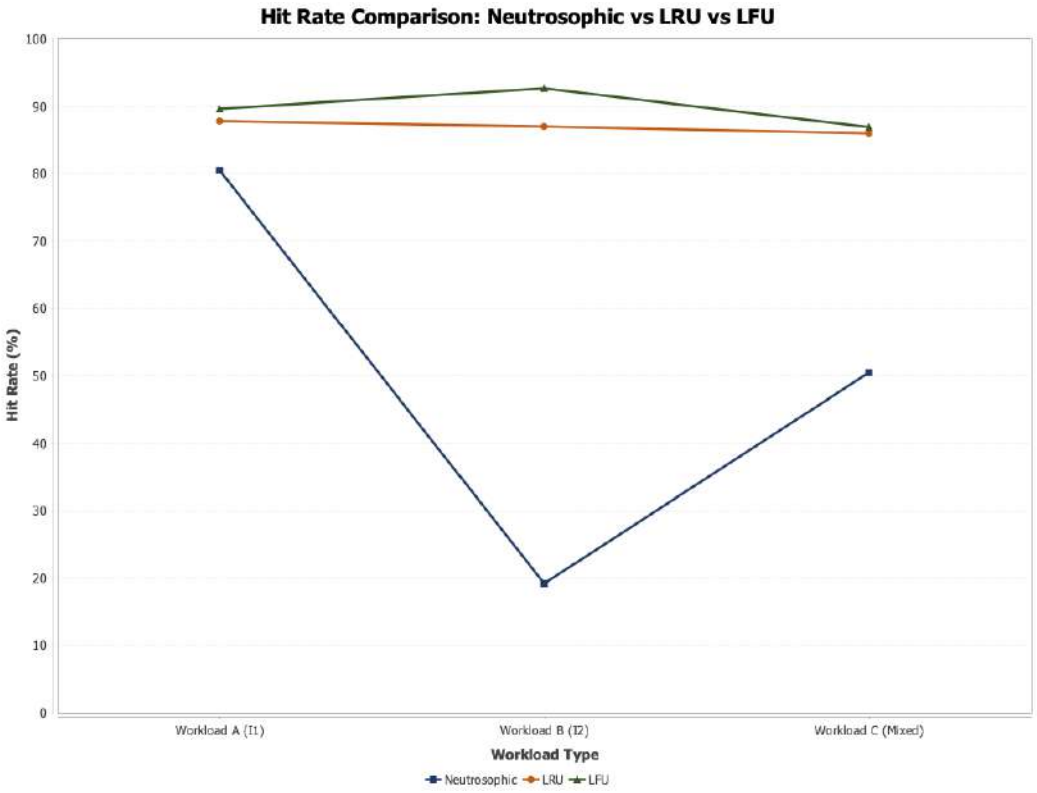

Figure 1: Hit Rate Comparison Graph: Neutrosophic Cache vs LRU vs LFU across workload types A, B, and C

5. Conclusion

I have proposed a neutrosophic cache replacement policy using improper neutrosophic integral-based utility functions for cache eviction decisions under indeterminate access conditions. The policy models each cache item as a 2-refined neutrosophic utility function $U(t) = T(t) + I_1(t) \cdot i_1 + I_2(t) \cdot i_2 + F(t)$, where convergence of the improper neutrosophic integral of the first kind determines eviction and divergence determines retention.

Acknowledgements. The authors thank the anonymous reviewers for their careful reading and constructive comments.

Conflict of interest. The author confirms that there are no conflicts of interest related to the research or its publication

Author contribution. The study was conceptualized by the SAA. The initial draft of the manuscript was prepared and compiled by ADT, while SAA was able to carry out final editorial revisions, including the analytical component, offering critical perspectives and making substantial edits. The final version of the manuscript was also reviewed by SAA,

who is also ready to accept responsibility for the integrity of the work in its entirety.

Use of Generative AI and AI-Assisted Tools

We use generative AI and AI-assisted tools for tasks such as English grammar checking, and do not employ them in any way that violates ethical standards.

REFERENCES

1. F.Smarandache, *Neutrosophy: Neutrosophic probability, set, and logic*(1998). American Research Press.
2. F. Smarandache, n-Valued refined neutrosophic logic and its applications to physics. *Progress in Physics, 4*, (2013), 143–146.
3. W. A. Wulf, & S. A. McKee, Hitting the memory wall: Implications of the obvious. *ACM SIGARCH Computer Architecture News*, *23*(1), (1995), 20–24. https://doi.org/10.1145/216585.216588
4. J.L. Hennessy, & D. A. Patterson, *Computer architecture: A quantitative approach* (6th ed.) , (2017), Morgan Kaufmann.
5. A. Silberschatz, P. B. Galvin, & G. Gagne, *Operating system concepts* (10th ed.). (2018), Wiley.
6. L.A. Belady, A study of replacement algorithms for virtual storage computers. *IBM Systems Journal*, *5*(2), (1966) 78–101.
7. S. Podlipnig, et al., A survey of web cache replacement strategies. *ACM Computing Surveys*, *35*(4), (2003), 374–398.
8. N. Megiddo, & D. S. Modha, ARC: A self-tuning, low-overhead replacement cache. In *Proceedings of the 2nd USENIX Conference on File and Storage Technologies (FAST '03)* (2003), pp. 115 â€"130, USENIX Association.
9. A. Jain, & C. Lin, Back to the future: Leveraging Belady's algorithm for improved cache replacement. In *Proceedings of the 43rd International Symposium on Computer Architecture (ISCA '16)* (2016), pp. 78â€"89, IEEE. https://doi.org/10.1109/ISCA.2016.17
10. K.T. Atanassov, Intuitionistic fuzzy sets. *Fuzzy Sets and Systems*, *20*(1), , (1986), 87–96. https://doi.org/10.1016/S0165-0114(86)80034-3
11. Y.A. Alhasan, F. Smarandache, & H. E. Khalid, The linear 2-refined neutrosophic differential equations. *Neutrosophic Sets and Systems*, *75* (2025).
12. Y.A. Alhasan, F. Smarandache, & H. E. Khalid,The improper neutrosophic integrals. *Neutrosophic Sets and Systems*, *80*, (2025),45–67.
13. Nikhil Nanivadekar & Ben Manes, Caffeine: A high performance caching library for Java. GitHub Repository, (2015), Retrieved from https://github.com/ben-manes/caffeine.