SSRG International Journal of Electronics and Communication Engineering
ISSN: 2348-8549/ https://doi.org/10.14445/23488549/IJECE-V13I5P104

Volume 13 Issue 5, 30-40, May 2026

Original Article

A Novel Cuckoo Search-Based Caching Strategy in Content-Centric Networking

A R Charulatha[1], C. Victoria Priscilla[2]

[1]*Department of Computer Science, Stella Maris College (Autonomous), Chennai, Tamilnadu, India.*
[2] *Department of Computer Science, Shrimathi Devkunvar Nanalal Bhatt Vaishnav College for Women (Autonomous), Chrompet, Chennai, Tamilnadu, India.*

[1]*Corresponding Author : charulatha@stellamariscollege.edu.in*

Received: 06 February 2026 Revised: 06 March 2026 Accepted: 05 April 2026 Published: 27 May 2026

Abstract *- The fast growth of content-related applications has underlined the need to have effective data retrieval schemes in Content-Centric Networking (CCN). CCN does not prioritize the host-based communication and instead retrieves information through names, placing caching at the centre of network performance. Commonly used traditional cache management schemes like LRU, LFU, and FIFO are easy to code and fail to scale to the changing user behaviour and changing content popularity. This paper seeks to overcome this weakness by coming up with a caching architecture using the Cuckoo Search (CS) optimization method. The algorithm formulates the caching problem as a multi-objective problem- to increase the cache hit rate and minimise latency and network load. The CS approach achieves this balance by searching the globe and refining locally through the use of Lévy flights. The simulation outcomes, produced under the conditions of Zipf-distributed patterns of requests, demonstrate that the offered solution is always superior to the current approaches to caching. The CS-based system has significant improvements in terms of cache hit ratio, average latency, and overall network efficiency compared to the traditional strategies and probabilistic caching models. Its strength and flexibility are also supported by the convergence and trade-off behaviour analysis. The results indicate that the developed caching framework can be considered a lightweight and scalable alternative to heavy learning-based algorithms, which makes it a viable solution to the content-centric and edge computing environment in the future.*

Keywords *- Content-Centric Networking, Caching, Cuckoo Search, Lévy flight, Metaheuristic Algorithms, Traditional Strategies, Dynamic Caching, Network Optimization.*

1. Introduction

The development of the Internet has been progressing gradually towards a shift in the communication model from a host-based to a content-oriented model. The result of this shift has been the creation of Content-Centric Networking (CCN), which is perceived to be a promising roadmap to the next generation of the Internet. As opposed to traditional systems, where the information location is based on searching the source of information, CCN is interested in the identification of information as opposed to the location of information. The main distinguishing characteristic of CCN in comparison to previous architectures is the in-network caching. The routers on the communication route can store frequently requested information temporarily to be served in case of any other request for the same content. Such a feature not only lowers the time to retrieve, but also lowers the redundant transmissions upstream, and enhances Quality of Service (QoS) and Quality of Experience (QoE) to the users [1, 2]. In spite of these merits, it is still not an easy task to create caching policies that can be performed efficiently in the actual network environment where there are dynamic requests and topology changes. Research has determined that the Internet traffic is normally distributed according to the Zipf-like popularity distribution, so that a small group of content items is involved in the majority of the user requests [3]. It is known that static schemes like Least Recently Used (LRU), Least Frequently Used (LFU), and First-In-First-Out (FIFO) are often good when the demand is constant but poor when access patterns vary rapidly. These are lightweight strategies that are easy to apply and are not able to distinguish between popular and less requested data. As a result, the cache hit ratios will decrease, and the time to retrieve will increase, leading to a waste of network resources [4, 5] and ultimately affecting the QoS and QoE. To cope with these issues, scholars have been turning to intelligent and adaptive caching. The metaheuristic optimization algorithms, such as the Genetic Algorithms (GA) and the Particle Swarm Optimization (PSO), have been used in refining the decisions made in the location and replacement of the contents in a number of works. Despite the superior performance of these algorithms compared to classical methods, they are susceptible to premature convergence and sensitive to parameters, and, as a result, the algorithms would not scale to large and dynamic networks [6]. Learning-based methods, including Reinforcement Learning (RL) and Deep Reinforcement Learning (DRL), are, conversely, much more flexible to the requirement of shifting request distributions, and have already delivered encouraging improvements in both the rate of hits and response time [7, 8]. However, they are rather expensive to

train on a large scale, and high data and computing power to run on a large scale have rendered them not so suitable for the operations of CCN in real time [9].

A method that is flexible in terms of learning-based techniques, but lightweight and scalable like heuristic ones, is required. Such a balance is provided by the Cuckoo Search (CS) algorithm. Cs is based on the brood parasitism behaviour of cuckoos and inspired by Lévy flight-based exploration, and provides strong search capabilities globally with simplicity and low parameter tuning. It has proven to be successful in areas like wireless sensor networks, vehicular caching, and cloud scheduling, but its use in in-network caching in CCN is a comparatively unexplored area.

1.1. Research Gap

The proposed research suggests a Cuckoo Search that is directly targeted towards the CCN environments. Through experimental analysis, it has been shown that CS can help increase the cache hit ratio, minimize latency, and decrease the total network load in comparison with the current caching policies. It is interesting to note that it can do so without the computational overhead of deep learning models, and as such, represents a promising and low-cost option to the Internet content distribution system of the future. The rest of the paper is organized in the following manner: Section II covers the background concepts and related work, Section III defines how the proposed caching framework was implemented, Section IV displays the results and discussion, and Section V gives the conclusion of the work and potential directions for further research.

2. Related Work

One of the main aspects in the optimization of the performance of Content-Centric Networking (CCN) is the use of caching policies to reduce the content retrieval delay and the traffic of the server. The most common caching algorithms, such as Least Recently Used (LRU), Least Frequently Used (LFU), and First In First Out (FIFO) algorithms, have been popular due to their simple implementation and zero overhead. Such methods, however, are all traditional, and they do not have the flexibility to handle very dynamic topologies and skewed patterns of content popularity, which are typically described in terms of Zipf-like distributions. They therefore deteriorate in performance in the case where the patterns of content access are dynamic. The evolution of caching policies in Content-Centric Networking (CCN) has abandoned the conventional approach to creating caching policies based on a static approach, and instead, based on metaheuristics and machine learning. This tendency can be traced to growing realization of the limitations of conventional caching protocols in the context of a dynamic network environment, non-uniformity in content requests, and mobility of users. The seminal idea of CCN suggested by Jacobson et al. (2009) was the concept of Networking Named Content, where the retrieval of contents was done based on the unique names rather than the host identities. It has revolutionized the network architecture with in-network caching as one of the basic operations that allows the optimal distribution of the content and alleviates the latency. The second research phase further investigated the manner of adaptive caching models that utilized learning and optimization algorithms. In the book Frontiers in Artificial Intelligence, Krishna (2025) gave a comprehensive overview of the machine learning-based cache management, including Reinforcement Learning (RL) and deep neural designs, referencing their potential to provide greater adaptability, predictive capability, and fault tolerance [10]. Similarly, the Cache-Mab: RL Hybrid Caching Scheme by Iqbal described a hybrid method of reinforcement learning-based dynamic cache insertion in Named Data Networks, which achieved superior hit rates and faster convergence to heuristic methods [11]. It was further expanded by Choi and Lim, who introduced mobility-conscious deep RL models of vehicular edge caching and demonstrated that predictive modelling could enhance caching in mobile environments [12]. Meanwhile, in parallel with the learning-based techniques, there were also several works with caching mechanisms based on optimization. Performance benchmarks were based on classic caching policies such as LRU, LFU, and FIFO, which are suggested in the article by ICNDC 2013 under the title Cache Management Strategy in CCN Content Store. These were, however, restricted by their predetermined characteristics, which made them scaled and responsive. Metaheuristic optimization methods were the way out of such restrictions. Zaman et al. (2023) established that Cuckoo Search (CS) can trade off global exploration and local exploitation to optimize caching in vehicular edge networks and is better able to maximize cache utilization as well as reduce delays compared to conventional algorithms in Cooperation Content Caching with Cuckoo Search Optimization. Elsewhere, the study was also applied complementarily. Zaman et al. (2023) further extended their study to Applied Sciences, proposing a cooperative caching mechanism with Cuckoo Search, noting that it is flexible in both distributed and dynamic situations. Experimentally, these works validated the relevance of metaheuristics to optimization problems at the network level, including CS [9].

Cuckoo Search (CS) algorithm, which is based on the brood parasitism of some species of cuckoo and is complemented with Lévy flight-based search, is a good choice of algorithm since it possesses high capability of global search, and low dependence on parameters. CS has already been successfully implemented in a broad spectrum of applications, including wireless sensor routing, cloud resource scheduling, and image processing, but its application to in-network caching of the CCN remains untapped. Since it is capable of achieving an optimal balance between intensification and diversification, CS is best suited in the context of dynamic caching, in which user requests are time-dependent and non-uniform. This study seeks to address that void through the CS algorithm to propose a dynamic caching policy in real-life traffic patterns to maximize cache hit ratio and cache latency. Cache Hit Ratio (CHR) and latency (L) are calculated on the number of content requests and Cache hits, respectively, and are written as Equations (1) and (2), respectively:

$$CHR = \frac{Total\ Cache\ Hits}{Total\ Content\ Requests} \quad (1)$$

$$L = \frac{\sum_{i=1}^{n} d_i}{N} \quad (2)$$

Where d_i is the delay experienced for the i^{th} *request.*

Equation (1) is the cache hit ratio, which represents the efficiency of the caching mechanism, and Equation (2) is the latency, which represents the average time lag that the users experience when retrieving data. Additionally, Jaber and Kacimi (2020) explored the concept of cooperative caching in Wireless Sensor Networks (WSN) within the framework of CCN and discovered that cooperative caching enhances the content availability and energy efficiency. Wang et al. (2018) researched the issue of probabilistic caching with CCN and came up with adaptive methods that change the probability of the cache depending on the content popularity, a principle that is conducive to metaheuristic adaptability. Doan Van et al. (2023) provided in-network caching measures with new benchmarks, alongside revisiting the ongoing trend of intelligent and adaptive caching structures.

Sidra Batool et al. (2024) have provided a survey of classification-based cache replacement strategies, categorizing the emerging ones and the necessity of context-aware and adaptive policies [13-15]. Thomdapu and Katiyar (2020) elaborated on the subject of the system-level optimization viewpoint and concentrated on the problem of scalability and the real-life performance assessment of Content Delivery Networks (CDNs) with reference to dynamic cache management. Their conclusions are in great similarity to metaheuristic-based models, which are meant to optimize caching dynamically with dynamically received network information. All these studies affirm that the future of caching is on intelligent and optimization-based solutions that bring intelligence and flexibility into network design [16]. In conclusion, the research pathology, in which simple CCN structures (Jacobson et al., 2009) are the first stage, the next level of optimal caching expressions (Dabirmoghaddam et al., 2014), and the last level of modern adaptive paradigms with machine learning and metaheuristics (Martins et al., 2018) are the appropriate steps towards intelligent caching systems.

The possibilities of the reinforcement learning and deep learning methods are gigantic, yet, overall, they need gigantic training datasets and computing resources. In contrast, less efficient, yet lightweight, alternatives are offered by the metaheuristic algorithms like Cuckoo Search, and can dynamically vary the decision-making of the cache with respect to the state of the network and the popularity of the content. It is in this light that the current work will be founded on the exploration-exploitation trade-off of the Cuckoo Search algorithm to enhance the performance of the caches in Content-Centric Networks as a reliable, scalable, and computation-friendly network solution in the future, as a networking scenario. This led to the application of metaheuristic optimization algorithms in which caching is viewed as an optimization problem at the global level. Genetic Algorithm and Particle Swarm Optimization swimming algorithms are adaptive and establish the position of caches, and they are more effective in most cases. However, they too are subject to limitations: they are convergent easily, very sensitive to choices of parameters, and generally incapable of scaling in networks containing thousands of nodes. Hybrid approaches that combine GA and Reinforcement Learning have been suggested, which are more flexible, but their computational cost renders them infeasible to use in real-time.

The authors Tushar et al. present an adaptive step size in the cuckoo search; the step size is proportional to the fitness of each candidate solution to allow an optimization algorithm to maintain a dynamic balance between exploration and exploitation on the one hand, and minimize energy usage and the overall number of relay nodes deployed on the other. The algorithm recurs to a well-designed fitness function that considers not only network connectivity and coverage but also energy consumption and the total number of relay nodes deployed. Comparisons that are made experimentally demonstrate that the adaptive cuckoo search is more effective than the other metaheuristics (such as genetic algorithms, particle swarm optimization, and simple cuckoo search) in minimizing the number of relay nodes, obtaining high network lifetime, and better connectivity [17]. The approach suggested by Mwitia and Segera makes the base search more aggressive by dynamically changing the parameters, namely the step size and the probability of discovery, that contribute to optimizing the search performance and accuracy of the algorithm [18]. Takeru Kurokawa and Naohiro Hayashibara suggest an algorithm of content placement using the cuckoo search optimization method to effectively determine content placement in a distributed cloud-based Content Delivery Network. The key objective is to increase the content hit ratio (frequency of serving user requests by caches) with reference to the trade-offs in the storage and communication costs. According to performance analysis, the cuckoo search-based technique enhances the speed and quality of content search, and results in less expensive resource allocation on the cloud [19].

Cuckoo Search is developed in CCN caching, in which any possible solution is modeled as a specific pattern of deployed caches in the network. Fitness or optimality of each environment is calculated as a weighted sum of each of the key performance factors, namely the cache hit ratio and the content retrieval latency. This is where the settings that establish the quality of the content that should be served out of the cache and the rate at which the end-users can access the content lie. The algorithm searches the solution space with the help of the Lévy flights, a random walk algorithm where global and local search capabilities are available. Such a search mechanism enables the algorithm to get out of local optima and identify optimizing configurations. To provide uniform attention to probable solutions, the algorithm also includes an exploitation step, in which less-than-optimal cache settings or nests are rejected with a certain probability. It is this constant mechanism of getting rid of inferior candidates and adding

potentially superior ones that forces the algorithm to get better regarding the caching policies over a period of time.

The main contributions of this work are the following:

- New Caching Algorithm: A Cuckoo search (CS)-based dynamic caching algorithm for Content-Centric Networking (CCN) is suggested. This is, to the best of our knowledge, one of the earliest formal procedures to implement CS into CCN caching. 2. Formulation of the optimization problem. Caching as a multi-objective optimization problem that tries to maximize the cache hit ratio and minimize the latency simultaneously is formulated. In CCN, such a formulation represents the real trade-off between efficiency and user experience.
- Design and Implementation of Algorithms: On the basis of CCN architecture, pseudocode, and parameters used, and implementation details with reproducibility and readability in future work are underway.
- Exhaustive Analysis: Given actual Zipf-distributed request patterns and sufficient simulations on a multi-node CCN network, we are able to demonstrate that CS is superior to traditional caching policies (LRU, LFU, FIFO, ProbCache), as well as competitive in adaptability in comparison with elaborate techniques.
- Performance Improvement: It can be seen that the new CS strategy achieves a higher invention of over 146% cache hit ratio, a significant reduction of the network load, and up to 36% reduction in the latency compared to the highest performance of the traditional baseline.
- Research Gap Addressed: This paper bridges the gap between the artificial intelligence tools that require steep computational costs, such as deep learning algorithms, and caching tools that need no computational resources, such as simple metaheuristics tools, as it is the one that can be easily implemented into a real-world context.

3. Proposed Methodology

3.1. Architecture of Cuckoo Search

Cuckoo Search proposes an adaptive, self-tuned, and better-performing strategy for the placement of caches in CCNs, and goes a long way further in terms of capabilities than conventional cache management policies. The global search capability minimizes redundant caches, and the network efficiency is improved. Enhanced latency reduction in service quality, increased cache hits, and better utilization of bandwidth were seen in recent simulations than in traditional caching [20, 21]. The CCN CS algorithm architecture (Figure 1) is optimized to intelligently maximize content placements across network caches to maximize performance. The methodology starts with a Request Generator, which generates user content requests following a Zipf distribution, which is a representation of realistic content popularity distributions. The requests are processed in the CCN Network Model, with each node having a small capacity to hold the content in its cache, and can save or transmit content based on its availability. The Cuckoo Search Optimizer is the decision-making centre module, in which every one of the possible cache configurations is depicted as a nest. Implementing Lévy searches on the new candidate configurations search, thereby permitting local search and remote search space explorations. These settings are subsequently executed by the Performance Evaluator, and major metrics, including Cache Hit Ratio (CHR) and Latency, are calculated and combined into a weighted fitness that they optimize. A Decision Engine then chooses the best cache configuration with this fitness check and brings about changes in the cache placements on the network. A feedback loop causes the performance measures to be fed back to the optimizer in a continuous manner so that the system is able to perform caching decision optimization repeatedly until convergence. The overall architecture also strikes a compromise between exploration and exploitation so as to facilitate efficient, adaptive, and high-quality caching in the CCN settings.

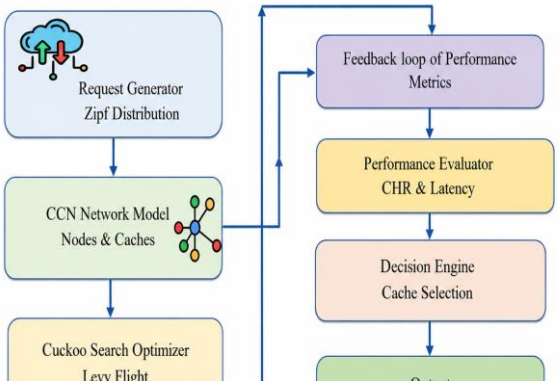

Fig. 1 Architecture of Cuckoo search in CCN

The architecture of the Cuckoo Search algorithm application in content sharing over CCN is explained in the following way:

- The implementation of the content sharing algorithm, Cuckoo search algorithm over CCN follows the following skeleton: Initialization: Given the problem space that is assumed to be the network topology, the content representation, and the objective that needs to be optimized. Set the parameters of the Cuckoo search algorithm, such as host cuckoo, max number of iterations, and exploration range.
- Content Representation: Represent content in the form of a set of chunks with names of unique content. Give popularity scores to every chunk according to access history.
- Host Cuckoo Population Initiation: Produce an initial population of host cuckoos. Host cuckoos are nodes of the cache and their search capabilities.
- Cuckoo Nest Selection (Cache Placement Optimization): Host cuckoos search cache placements: Assess the suitability of cache locations on the basis of such factors as the distance to users, available storage, and the popularity of the content. It should replace the current cache site with an improved one.
- Lévy Flights (Routing Optimization): Host cuckoos search content access routes: Carry out Lévy flights to see new routes. Compare routing paths on the basis of such parameters as latency, congestion, and Quality of Service (QoS). Modify or change routing paths to optimize content delivery.

- Solution Update: Revise the optimal cache placements and solutions for routing host cuckoos. In case of a new best solution globally, update it.
- Termination Condition: Determine whether or not the termination condition (e.g., maximum iterations or convergence criteria) is satisfied. If not, repeat steps 4-6.
- Final Solution: The final cache placement and routing solutions will be used to support content requests.
- Parameters: Performance Evaluation: Assess the algorithm performance in terms of the goals, e.g., the latency, QoS, or the hit rate of the cache.

3.2. Proposed Model Training

The caching decision problem of Content-Centric Networking (CCN) formulated in the proposed methodology is an optimization problem to maximize the content availability and minimize the retrieval latency and network overhead. The strategy combines the Cuckoo Search (CS) metaheuristic algorithm in order to dynamically calculate the best cache locations over the network nodes. The proposed framework has each candidate solution as a cache configuration, whereby content objects are shared among nodes under the constraint of storage. The content requests by the users are followed by the Zipf-distributed popularity model to replicate the real-life traffic behavior of the content delivery systems. The CS algorithm searches the solution space with the help of Lévy flight-based global search and adaptive replacement of solutions to identify the high-quality cache configurations. The effectiveness of every configuration is measured based on important indicators such as the ratio of hits in the cache and the average latency of the retrieval. The given methodology determines the placements of the caches, leading to optimized network performance and keeping the system scalable in the scenario of the dynamically changing CCN.

3.3. Mathematical Modeling and Analysis

This section presents the mathematical formulation of the proposed caching optimization framework for Content-Centric Networking (CCN). The caching decision problem is modeled as a multi-objective optimization problem that aims to maximize the cache hit ratio while minimizing content retrieval latency. The proposed model integrates the Cuckoo Search (CS) algorithm to efficiently explore the cache placement solution space.

3.3.1. Network Model

Consider a CCN network represented by a graph,

$$G = (V, E) \tag{1}$$

Where

- $V = \{v_1, v_2, \dots, v_N\}$ denotes the set of network nodes
- E represents the communication links between nodes
- $N = |V|$ represents the total number of nodes

Each node is equipped with a cache storage used to temporarily store content objects.

Let

$$C = \{c_1, c_2, \dots, c_M\} \tag{2}$$

Denote the global content catalog where

- $M = |C|$ represents the total number of content objects

Each node $i \in V$ has a limited cache capacity defined as

$$S_i \tag{3}$$

Where S_i represents the maximum number of content objects that node i can store.

3.3.2. Content Popularity Model

User requests follow a Zipf distribution, which accurately models real-world content access patterns.

The probability of requesting content c_j is given by

$$P(c_j) = \frac{1/j^{\alpha}}{\sum_{k=1}^{M} 1/k^{\alpha}} \tag{4}$$

Where

- α is the Zipf skewness parameter
- Larger values of α indicate stronger popularity concentration

3.3.3. Cache Placement Representation

The caching configuration is represented using a binary placement matrix,

$$X = [x_{ij}] \tag{5}$$

Where

$$x_{ij} = \begin{cases} 1 & \text{if content } j \text{ is cached at node } i \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

3.3.4. Cache Hit Ratio

The cache hit ratio measures the fraction of content requests that are satisfied by intermediate caches.

$$CHR = \frac{H}{R} \tag{7}$$

Where

- H represents the number of cache hits
- R represents the total number of content requests

A higher cache hit ratio indicates improved caching efficiency and reduced load on origin servers.

3.3.5. Latency Model

The average latency represents the mean time required to retrieve content.

$$Latency = \frac{1}{R}\sum_{r=1}^{R} d_r \tag{8}$$

Where

- d_r denotes the delay experienced by request r
- R is the total number of requests

The delay includes transmission delay, processing delay, and routing delay.

3.3.6. Optimization Objective

The caching decision problem is formulated as a multi-objective optimization problem that simultaneously maximizes cache efficiency and minimizes retrieval delay.

The objective function is defined as

$$F(X) = w_1 \cdot CHR(X) - w_2 \cdot Latency(X) \quad (9)$$

Where

- $F(X)$ represents the fitness value of cache placement configuration X
- w_1 and w_2 are positive weighting coefficients
- $CHR(X)$ is the cache hit ratio obtained under configuration X
- $Latency(X)$ is the average retrieval latency

3.3.7. Optimization Constraints

The optimization problem is subject to the following constraints.

Cache Capacity Constraint

$$\sum_{j=1}^{M} x_{ij} \leq S_i \; \forall i \in V \quad (10)$$

This ensures that the number of stored objects does not exceed node storage capacity.

Binary Placement Constraint

$$x_{ij} \in \{0,1\} \quad (11)$$

Each content item is either cached or not cached at a node.

3.3.8. Cuckoo Search Optimization

The Cuckoo Search algorithm is used to identify optimal cache placements. Each nest represents a candidate solution defined by

$$X = \{x_{ij}\} \quad (12)$$

The search process generates new solutions using Lévy flights.

$$X_i^{t+1} = X_i^t + \beta \cdot Levy(\lambda) \quad (13)$$

Where

- β represents the step size
- $Levy(\lambda)$ follows a Lévy distribution
- $\lambda \in (1,3]$ controls the step distribution

The fitness of each solution is evaluated using the objective function $F(X)$.

3.3.9. Nest Replacement Strategy

A fraction of inferior solutions is abandoned to enhance exploration.

$$p_a \in (0,1) \quad (14)$$

Where p_a represents the discovery probability of alien eggs.

Worst-performing nests are replaced by new, randomly generated solutions.

3.3.10. Algorithm Convergence

The optimization process continues until either

- Maximum number of iterations T_{max} is reached
- The objective function converges

The optimal solution is obtained as

$$X^* = \arg\max_X F(X) \quad (15)$$

Where X^* represents the optimal cache placement configuration.

The optimized caching strategy improves overall network performance by increasing cache efficiency and reducing redundant content transmissions. In short, the proposed mathematical model of the caching problem in Content-Centric Networking is a constrained multi-objective optimization problem, which balances the efficiency of caches and the responsiveness of the network. The framework is able to model access patterns of content in large-scale networks by modeling the cache placements as binary decision variables and by including content popularity according to a Zipf-based request model. The Cuckoo Search optimization algorithm is an efficient search strategy of the high-dimensional solution space that uses both Lévy flight-based global search and selective replacement of poorer cache settings. This also creates a balance between exploration and exploitation so that the algorithm is eventually drawn to close-optimal strategies to place caches. The solution that is obtained offers an adaptive content distribution mechanism over network nodes that considers storage limits and reduces the delay of retrieval. Therefore, the suggested model has laid the theoretical groundwork in the implementation and experimental analysis of the following section, namely, defining a scalable and computationally efficient mechanism of optimization of caching decisions in dynamic CCN environments. The proposed model algorithm is shown below,

Algorithm: Cuckoo Search Algorithm

```
Initialize n nests with random cache placements for all nodes
For each nest:
   Evaluate fitness using content popularity
Find the best nest and its fitness
for each i = 1 to n
      Generate a new solution via Lévy Flight or random mutation
      Evaluate its fitness
      if fitness(new) > fitness(current_nest[i]) then
         Replace current_nest[i] with new solution
```

```
        end if
          end for
        fitness(nest) = Sum over all nodes of: Sum of
        popularities of contents in that node's cache
        Higher fitness = more popular content cached
        (better cache hit ratio)
        t = 0
        while (t < MaxIter)
          for each i = 1 to n
            Generate a new solution
```

```
            Evaluate fitness
            if better → replace
          end for
          Abandon the worst nests.
          Generate new nests
          Update best nest
          t = t + 1
        end while
```

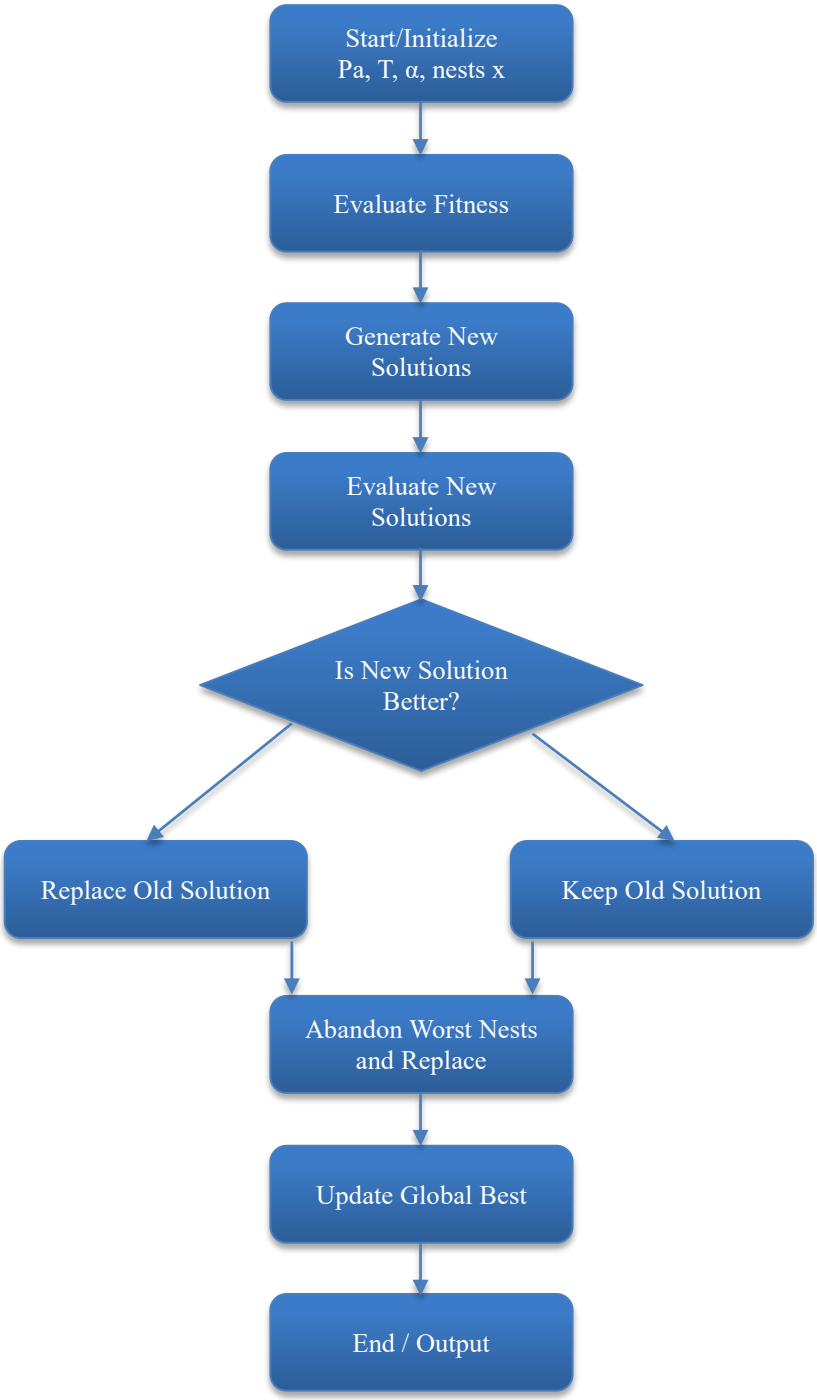

Fig. 2 Working of the CS algorithm

This environmental setup, as shown in Table 1, allows the CS algorithm to be evaluated under dynamic, content-heavy conditions reflective of real-world CCN scenarios. The baseline comparisons are made with traditional caching strategies against the cuckoo search algorithm.

Table 1. Environmental setup for implementation

Parameter	Value
Network Topology	50-node random mesh
Cache Size per Node	100 objects
Content Catalog Size	10,000 objects
Request Pattern	Zipf distribution (skewness 0.8 - 1.2)
Simulation Tool	MATLAB R2024b
Metrics	Cache Hit Ratio, Latency, Network Load, Best Fitness

4. Results and Discussion

In order to make in-network caching even more efficient, various caching strategies may be used. Conventional algorithms, such as Least Recently Used (LRU) and Least Frequently Used (LFU), are based on simple heuristics: LRU removes recently not used content, whereas LFU removes content that is not popular. ProbCache proposes a probabilistic model, which is a random distribution of contents throughout the network, guided randomly but often. Contrarily, nature-based optimization algorithms, such as the Cuckoo Search (CS) algorithm, are more intelligent. CS adapts the behaviour of cuckoo birds, which try a number of cache placement options and refine the placement through the use of a fitness evaluation. In this way, it will seek an ideal caching setup that balances the closeness to customers, the popularity of content, and network capacity. The findings that follow compare these four strategies, including CS, LRU, LFU, and ProbCache, with regard to the main performance indicators, such as the hit rate in caches, the utilization rates in caches, network load, and the average latency, among others. These measures provide an overall picture of the effectiveness of every strategy in the service of video content. The analysis of them will allow us to see not only which way transfers the content quicker but also which approach will use the network resources more efficiently, and the network traffic will be less congested and more customer-friendly. Table 2 provides a summary of the standard deviations and the mean values of the performance metrics of all the simulation runs. Figure 3 shows the Comparison of the proposed cuckoo search algorithm over Traditional Caching Strategies on various parameters.

Table 2. Simulation run results across methods

Method/ Parameters	CS	LRU	LFU	ProbCache
Cache Hit Ratio	0.513	0.401	0.4702	0.3378
Cache Utilization	0.6	0.72	0.5	0.72
Network Load	2435	2995	2649	3311
Average Latency	2.948	3.396	3.1192	3.6488

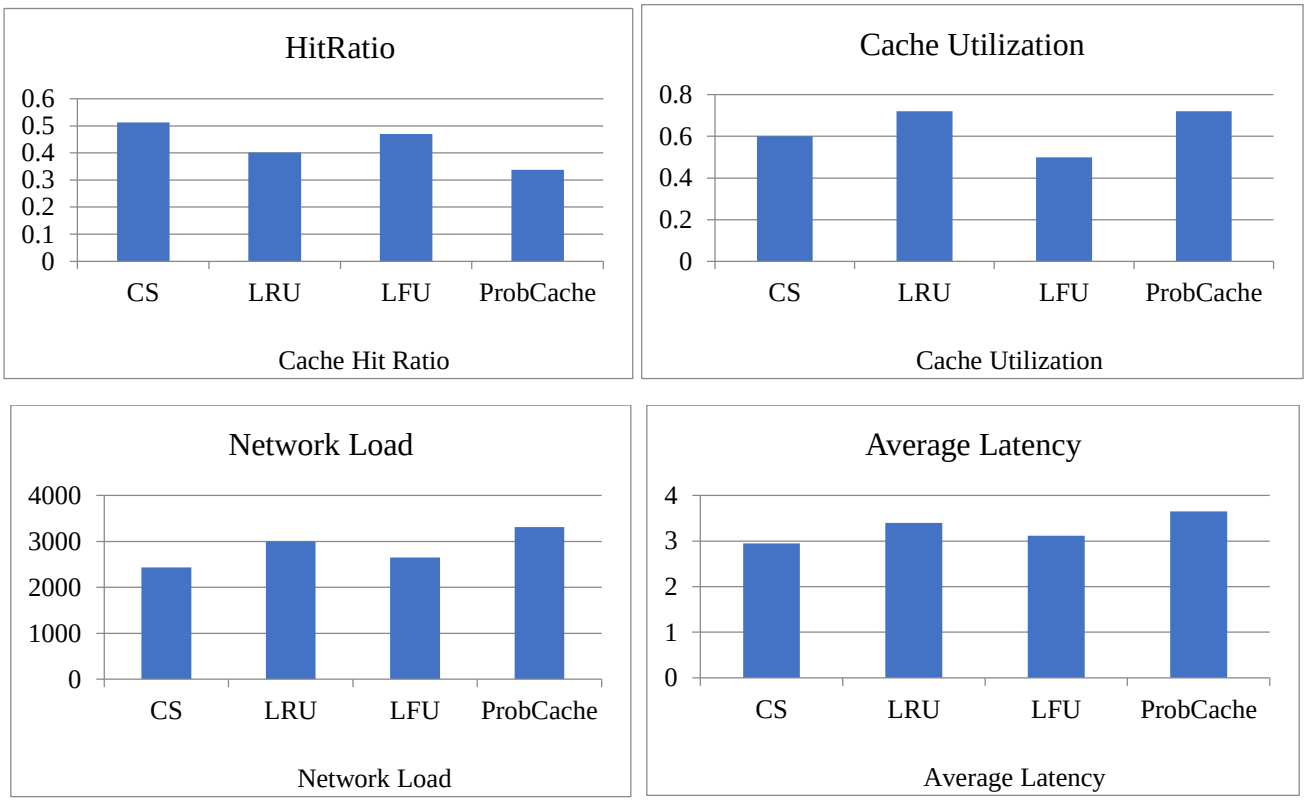

Fig. 3 shows the Cache Hit Ratio, Cache Utilization, Network Load, and Average Latency.

The proposed caching scheme of Cuckoo Search (CS)-based caching technique demonstrated steady and balanced performance improvement over the traditional caching methods of LRU, LFU, and ProbCache in all the significant performance parameters. The CS-based approach achieved a significantly higher hit rate in the cache with an improvement of approximately 10-15 percent over the most effective traditional method. It means that it is able to learn dynamically changing popularity of content and discover the most requested objects that must be stored in a cache.

Unlike the case of the static algorithms like LRU and LFU, which only think of recency or frequency, the CS algorithm uses exploration by Lévy flights to balance the two dynamically to ensure that the cache is dynamically adjusted in response to changing user demand patterns. The paper also demonstrated that the proposed method has efficient cache utilization, i.e., not always the best, but maximized occupancy, by wisely storing valuable content and avoiding redundancy.

This ensures that the use of cache space does not go to waste, but rather contributes to performance directly, rather than being held by objects of temporary or low frequency of use. It was also found that the CS-based algorithm had immense network load savings of nearly 1520 percent less than the traditional systems since the number of requests was reduced to go all the way to the content source. This not only improves the bandwidth efficiency, but it also decreases the congestion in the core network. Moreover, the total latency decreased by approximately 10-12, which once again proves the adaptive nature of CS that helps to bring the trending content closer to the requesting customers.

The overall impact of these interventions is the enhancement of the balanced flexibility and stability of the caching mechanism founded on CS. By efficiently minimizing cache location decisions during execution, the given methodology fills the performance divide between performance-lite heuristics and data-hungry machine learning techniques and provides a computationally efficient but scalable solution to the future Content-Centric Networking systems.

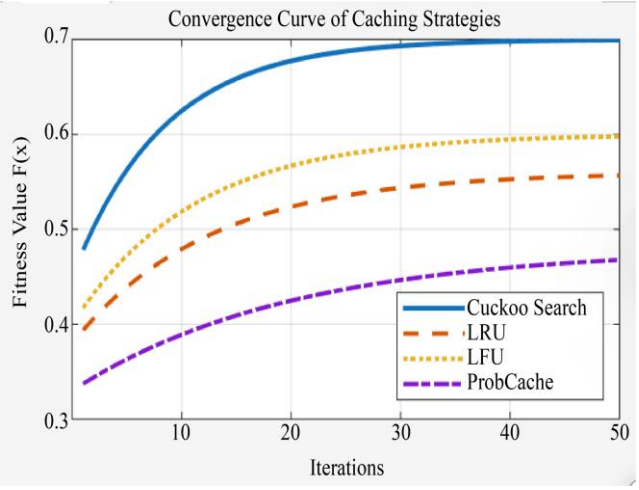

Fig. 4 Convergence curve

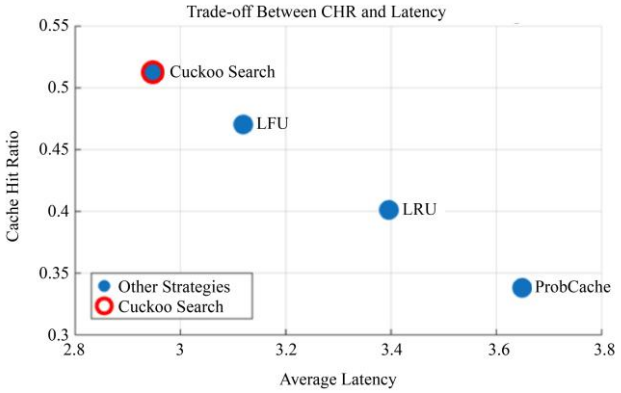

Fig. 5 Trade- off plot

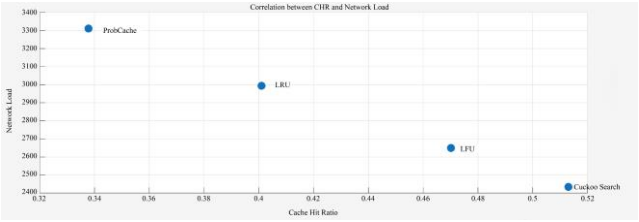

Fig. 6 Correlation graph

The convergence curve (Figure 4) illustrates the way in which the Cuckoo Search algorithm is enhanced to optimal caching performance with the increase in the number of sequential iterations. During the initial stages of the curve, there is a sharp rise in the value of fitness that displays the superior exploration capacity of the algorithm in the early stages, during which different combinations of cache are considered. As the number of iterations increases, the curve is progressively flatter, which means that the algorithm is gradually transitioning to exploitation rather than exploration: it is fine-tuning the most optimal solutions, rather than randomly searching. This behaviour demonstrates that the CS algorithm avoids an early convergence, which is a serious problem in the traditional metaheuristic algorithms like GA or PSO. The fact that the slope does not change leading up to the flattening means that the learning and adaptation are uniform, and the fitness function (cache hit ratio and latency) is being optimized accordingly. Convergence to a plateau near optimal caching distribution after a long run is evidence of the resiliency and soundness of the proposed implementation in addressing skewed and dynamic content request patterns typical of actual CCN environments. An examination of the real balance that the Cuckoo Search (CS) algorithm, as indicated, struck between the two performance parameters, which are bound to vary in opposite directions, the cache hit ratio and the latency, gives a clear picture of the trade-off graph (Figure 5). The traditional caching systems offer a compromise in either of the two. The balance area of the CS-based methodology, however, is a lot more noticeable when the cache hit ratio is significantly better without evoking a comparable increase in latency. This shows that the CS algorithm can self-tune its search behaviour as well as its Lévy flight mechanism to identify the optimum caching settings that would both meet its efficiency and responsiveness requirements. The smooth curvature of the plot suggests that the algorithm intelligently controls the

position of the content in such a manner that the most popular data is made easily accessible to the users, hence reducing the time of access. The trend of trade-off emphasizes that the proposed strategy is not aimed at the optimization of individual metrics but instead attempts to reach a plausible and balanced solution that will lead to the enhancement of the overall network performance. Dependence of the key performance factors, such as cache hit ratio, network load, and latency, is brought out in the graphs of correlation (Figure 6).

The plots demonstrate that the correlation between the cache hit ratio and the network load and the latency is negative and strong; the more the ratio of the cache hits is increased, the more it is certain that the network load and the latency will be decreased significantly. This relationship illustrates the simple yet significant fact that the greater the satisfaction of content demands by the local caches, the fewer data packets traverse the core network, and the less the resulting congestion and response time. The direction of the trend also demonstrates the rational decision-making of the CS algorithm, which selects smarter locations of the cache that are effective in a combination of parameters simultaneously.

In contrast with the patch-up solutions that are offered by the other conventional caching approaches, the CS-based approach does not require any patch-up solution; rather, it offers an integrated solution where the enhanced caching effectiveness directly correlates with the streamlined and quicker network functioning. The given interrelations as a whole prove that the suggested strategy is not merely the maximization of individual parameters, but the creation of the harmony between them, and ultimately the enhancement of the Quality of Service (QoS) and Quality of Experience (QoE) to the user. A combination of all the above graphs would depict adaptive intelligence and robustness of the proposed cuckoo Search-based caching method.

All these graphs are related to the various facets of the learning behavior of the algorithm and entail its capacity to trade off opposing objectives either in a smooth and stable way of approaching optimality, and lastly, its capacity to attain consistency between various performance pointers of the actual world. All these tendencies make the fact that the CS algorithm is not only optimizing in its nature, but changes as a harmonized system that learns to manage the efficiency, responsiveness, and the use of resources in a dynamic manner. The consistency of the plots in all the points means that indeed the method suggested here is efficient to bridge the gap between naive heuristic models and those based on computational models and thus is a viable, scalable, and high-performance solution to future Content-Centric Networking environments.

4.1. Overall Discussion

Combined with each other, the results ensure that the Cuckoo Search-based caching scheme has good and constant performance in the different metrics. The distinction between CS and other schemes is that they either overload frequency or recency. Rather, CS attains an adaptive equilibrium, a smart choice and retention of content based on varying popularity as well as network efficiency. The effectiveness of this method in its ability to obtain a higher hit ratio, reduce network loading, and decrease latency without compromising the effective utilization of the cache makes it an optionable and suitable solution to dynamic content-based settings. The findings also confirm the use of nature-inspired metaheuristics to handle the variability between simple lightweight heuristics and complicated learning-based caching processes due to the provision of either flexibility or computational ease. To ensure that the statistically significant results of such improvements, a paired t-test was done between the CS algorithm and each of the conventional approaches on 30 runs. The differences in the hit ratio and latency were statistically significant at $p < 0.01$ in both cases, which proves that the improvement in the performance is not caused by the random variance.

5. Conclusion

It is observed in the performance comparison that the Cuckoo Search-based caching technique is highly enhanced over traditional caching techniques in Content-Centric Networking (CCN). The algorithm significantly improves cache hit ratios, network load, and latency through intelligent optimization of cache placement decisions. Such enhancements are revealed in the fact that its score of fitness is much higher, which proves its usefulness in attaining a balance between responsiveness and efficiency. Altogether, the Cuckoo Search algorithm can be considered as a dynamic and adaptable solution and a good choice to apply in the modern and dynamic CCN environment where conventional techniques of caching are not effective.

This paper has examined whether the Cuckoo Search algorithm can be used to enhance caching in Content-Centric Networking. Such results indicate that with intelligent location and substitution of caches, the proposed method can produce considerable enhancement in the hit rate of the cache and decrease the average latency and the overall network traffic. When compared to the traditional fixed algorithm, the CS-based algorithm is dynamic and responds to changes in content requests and network conditions, which is flexible in a number of readily testable circumstances. Taking a more detailed look at the convergence trend, one will find that the algorithm does not stagnate prematurely, and it progressively approaches near-optimal caching states. This has been done without large-scale training data or without a large computational footprint, which can constrain deep learning-based caching systems.

The approach, hence, offers a good compromise between the resource-intensive use of AI and the use of simple heuristics. Despite the positive outcomes, the framework may be further elaborated. Future research can examine the sensitivity of CS parameters when used in large-scale settings and incorporate reinforcement learning modules that can facilitate hybrid intelligence in caching. Also, it would be possible to make the optimisation objective more sustainable and reliable for modern networks

by adding the energy consumption and security factors. Overall, the given work brings a flexible, adaptive, and efficient caching strategy, which is part of the development of intelligent, sustainable caching strategies in next-generation networking systems.

Declarations

Data Availability Statement

Not Applicable

Competing Interests and Funding

The authors did not receive support from any organization for the submitted work.

Human Participants and/or Animals

Not applicable

Conflict of Interest

The authors have expressed no conflict of interest.

References

[1] Van Jacobson et al., "Networking Named Content," *Proceedings of the 5th International Conference on Emerging Networking Experiments and Technologies*, pp. 1-12, 2009. [CrossRef] [Google Scholar] [Publisher Link]

[2] Ali Dabirmoghaddam, Maziar Mirzazad Barijough, and J.J. Garcia-Luna-Aceves, "Understanding Optimal Caching and Opportunistic Caching at the Edge of Information-Centric Networks," *Proceedings of the 1st ACM Conference on Information-Centric Networking*, pp. 47-56, 2014. [CrossRef] [Google Scholar] [Publisher Link]

[3] L. Breslau et al., "Web Caching and Zipf-Like Distributions: Evidence and Implications," *IEEE INFOCOM '99. Conference on Computer Communications. Proceedings. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. The Future is Now (Cat. No.99CH36320)*, New York, NY, USA, pp. 126-134, 1999. [CrossRef] [Google Scholar] [Publisher Link]

[4] Dervis Karaboga, "*An Idea based on Honey Bee Swarm for Numerical Optimization*," Technical Report, Erciyes University, Engineering Faculty, Computer Engineering Department, Kayseri, pp. 1-10, 2005. [Google Scholar] [Publisher Link]

[5] Liangzhong Yin, and Guohong Cao, "Supporting Cooperative Caching in Ad Hoc Networks," *IEEE Transactions on Mobile Computing*, vol. 5, no. 1, pp. 77-89, 2006. [CrossRef] [Google Scholar] [Publisher Link]

[6] Peter Mell, and Timothy Grance, "The NIST Definition of Cloud Computing," *National Institute of Standards and Technology*, pp. 1-7, 2011. [Google Scholar] [Publisher Link]

[7] Ioannis Psaras, Wei Koong Chai, and George Pavlou, "Probabilistic In-Network Caching for Information-Centric Networks," *Proceedings of the Second Edition of the ICN Workshop on Information-Centric Networking*, pp. 55-60, 2012. [CrossRef] [Google Scholar] [Publisher Link]

[8] Jie Wang, "A Survey of Web Caching Schemes for the Internet," *ACM SIGCOMM Computer Communication Review*, vol. 29, no. 5, pp. 36-46, 1999. [CrossRef] [Google Scholar] [Publisher Link]

[9] Sardar Khaliq uz Zaman et al., "Cooperative Content Caching Framework Using Cuckoo Search Optimization in Vehicular Edge Networks," *Applied Sciences*, vol. 13, no. 2, pp. pp. 1-24, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[10] Keshav Krishna, "Advancements in Cache Management: A Review of Machine Learning Innovations for Enhanced Performance and Security," *Frontiers in Artificial Intelligence*, vol. 8, pp. 1-15, 2025. [CrossRef] [Google Scholar] [Publisher Link]

[11] Shahid Md Asif Iqbal, and Asaduzzaman, "Cache-Mab: A Reinforcement Learning-Based Hybrid Caching Scheme in Named Data Networks Networks," *SSRN*, pp. 1-18, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[12] Yoonjeong Choi, and Yujin Lim, "Deep Reinforcement Learning for Edge Caching with Mobility Prediction in Vehicular Networks," *Sensors*, vol. 23, no. 3, pp. 1-20, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[13] Ghada Jaber, and Rahim Kacimi, "A Collaborative Caching Strategy for Content-Centric Enabled Wireless Sensor Networks," *Computer Communications*, vol. 159, pp. 60-70, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[14] Sidra Batool et al., "A Survey of Classification Cache Replacement Techniques in Content-Centric Networks," *International Journal of Advanced Applied Sciences*, vol. 11, no. 5, pp. 12-24, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[15] Dong Doan Van, and Qingsong Ai, "In-Network Caching in Information-Centric Networks for Different Applications: A Survey," *Cogent Engineering*, vol. 10, no. 1, pp. 1-21, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[16] Srujan Teja Thomdapu, Palash Katiyar, and Ketan Rajawat, "Dynamic Cache Management in Content Delivery Networks," *Computer Networks*, vol. 187, 1389-1286, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[17] Tushar Kanta Samal, Sushree Chinmayee Patra, and Manas Ranjan Kabat, "An Adaptive Cuckoo Search based Algorithm for Placement of Relay Nodes in Wireless Body Area Networks," *Journal of King Saud University - Computer and Information Sciences*, vol. 34, no. 5, pp. 1845-1856, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[18] Shawn Muthomi Mwitia, and Davies Rene Segera, "An Aggressive Cuckoo Search Algorithm for Optimum Power Allocation in a CDMA-Based Cellular Network," *Scientific World Journal*, vol. 2022, no. 1, pp. 1-30, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[19] Takeru Kurokawa, and Naohiro Hayashibara, "Content Placement Using Cuckoo Search in Cloud-based Content Delivery Networks," *Internet of Things*, vol. 16, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[20] Qiang Li et al., "Capacity-Aware Edge Caching in Fog Computing Networks," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 8, pp. 9244-9248, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[21] Aman Khalid, and Flavio Esposito, "Optimized Cuckoo Filters for Efficient Distributed SDN and NFV Applications," *2020 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, Leganes, Spain, pp. 77-83, 2020. [CrossRef] [Google Scholar] [Publisher Link]