Revenue-Aware Seamless Content Distribution In Satellite-Terrestrial Integrated Networks

Haftay Gebreslasie Abreha, *Student Member, IEEE,* Ilora Maity, *Senior Member, IEEE,* Youssouf DRIF, *Member, IEEE,* Christos Politis, *Member, IEEE,* and Symeon Chatzinotas, *Fellow, IEEE*

***Abstract*—With the surging demand for data-intensive applications, ensuring seamless content delivery in Satellite-Terrestrial Integrated Networks (STINs) is crucial, especially for remote users. Dynamic Ad Insertion (DAI) enhances monetization and user experience, while Mobile Edge Computing (MEC) in STINs enables distributed content caching and ad insertion. However, satellite mobility and time-varying topologies cause service disruptions, while excessive or poorly placed ads risk user disengagement, impacting revenue. This paper proposes a novel framework that jointly addresses three challenges: (i) service continuity- and topology-aware content caching to adapt to STIN dynamics, (ii) Distributed DAI (D-DAI) that minimizes feeder link load and storage overhead by avoiding redundant ad-variant content storage through distributed ad stitching, and (iii) revenue-aware content distribution that explicitly models user disengagement due to ad overload to balance monetization and user satisfaction. We formulate the problem as two hierarchical Integer Linear Programming (ILP) optimizations: one content caching that aims to maximize cache hit rate and another optimizing content distribution with DAI to maximize revenue, minimize end-user costs, and enhance user experience. We develop greedy algorithms for fast initialization and a Binary Particle Swarm Optimization (BPSO)–based strategy for enhanced performance. Simulation results demonstrate that the proposed approach achieves over a 4.5% increase in revenue and reduces cache retrieval delay by more than 39% compared to the benchmark algorithms.**

***Index Terms*—Satellite edge computing (SEC), content caching, content distribution, dynamic ad insertion.**

I. Introduction

With the growing data-intensive applications, mobile data traffic is surging. According to the *Ericsson Mobility Report 2024*, 5G mobile subscriptions are predicted to reach *5.6 billion* by *2029*, with video content comprising 80% of the total traffic demand [1]. Satellite-Terrestrial Integrated Networks (STINs) enhance broadband connectivity in remote and underserved areas by combining satellite coverage with terrestrial network capacity [2]. However, as video demand grows, traditional cloud-based content delivery over satellite networks encounters challenges such as long delays, bandwidth limits, and feeder link congestion. This drives satellite operators to integrate mobile edge computing (MEC) with satellite systems, advancing the concept of Satellite Edge Computing (SEC). For instance, Axiom Space and Red Hat take edge computing into orbit to support space-based cloud services [3]. SEC reduces feeder link congestion and delivery delays by allowing satellite nodes to cache content and process data onboard. However, satellites have limited resources, making it challenging to handle massive service requests while meeting user requirements. Therefore, integrating MEC-enabled ground gateways and user terminals is essential for caching content. However, existing content distribution schemes [4]–[6] handle the time-varying topology challenge by designing strategies for each time slot but often overlook services that span multiple slots, leading to the eviction of cached content in use and service disruptions.

Advertising is a key revenue driver in content distribution, accounting for a significant share of profits. According to the *2021 Google report*, over 80% of the company's revenue comes from online advertising [7]. Dynamic Ad Insertion (DAI) has become a key tool for content creators and distributors to maximize advertising monetization. DAI dynamically stitches ads into primary content using standards like HTTP Live Streaming (HLS) and Dynamic Adaptive Streaming over HTTP (MPEG-DASH) [8]. However, misaligned SCTE-35 markers (an ad insertion signaling standard) can disrupt the user experience through mistimed ad placements and improper ad selection, affecting user engagement. This underscores the need for comprehensive optimization to maximize ad monetization while minimizing user disengagement. Furthermore, beyond DAI standardization, some works focus on network architecture, primarily exploring Client-Side Ad Insertion (CSAI) and Server-Side Ad Insertion (SSAI), which define the ad stitching process at the server and user device, respectively [9]. The SSAI architecture is preferred over CSAI for its robustness against ad blockers, as it stitches ads directly at a cloud server [9]. Traditional SSAI-based schemes insert ads before caching to ensure personalized delivery. However, this approach consumes substantial feeder link bandwidth, as identical content with different ads must be offloaded and cached in multiple versions. This redundancy strains feeder links and increases storage demands due to duplicate content variations needed for personalized ad delivery.

Furthermore, while ad insertion increases revenue for content providers, a poorly designed DAI strategy can result in user drop-off during content playback, ultimately reducing overall revenue. User disengagement is influenced by several factors, including ad duration, frequency, and relevance. For example, the 2021 Conviva report shows

This work was supported in whole, or in part, by the Luxembourg National Research Fund, via project INSTRUCT (ref. IPBG19/14016225).

Haftay Gebreslasie Abreha, Ilora Maity, Youssouf DRIF and Symeon Chatzinotas are with the Interdisciplinary Center for Security, Reliability and Trust (SnT), University of Luxembourg, 1855 Luxembourg City, Luxembourg (e-mail: haftay.abreha@uni.lu; ilora.maity@uni.lu; youssouf.drift@uni.lu, symeon.chatzinotas@uni.lu).

Christos Politis is with SES S.A., Chateau de Betzdorf, Luxemburg (e-mail: christos.politis@ses.com)

that nearly 20% of users abandon content after just a 5-second ad delay [10]. Other contributing factors include the type of content; for example, live sports are more sensitive to ad interruptions compared to on-demand videos [11]. User demographics also play a role; for instance, younger audiences generally exhibit higher sensitivity to ads than older viewers [12], [13]. These insights underscore the importance of designing an effective DAI strategy that carefully balances monetization objectives with user experience to reduce disengagement and maximize long-term revenue.

We utilize multi-layer satellite networks, comprising Geostationary Earth Orbit (GEO) and Low Earth Orbit (LEO) satellites, MEC-enabled user terminals (e.g., VSATs), and MEC-enabled gateways to enhance content distribution and DAI. A multi-layer satellite network enables vast and efficient content delivery by leveraging GEO satellites for broad coverage and distributing content to numerous LEO satellites. In turn, LEO satellites provide fast, real-time responses to end users and facilitate cache-to-cache updates, minimizing dependence on remote cloud-based content servers. This hierarchical structure leverages the strengths of each layer to optimize service delivery. Despite these advantages, there remains a lack of research addressing the integration of DAI into satellite-based content delivery systems. Existing content caching and distribution strategies fail to address challenges unique to multi-layer satellite networks, such as dynamic user requests, time-varying link quality, and shifting satellite topologies. Moreover, most existing works analyze content caching and distribution strategies in isolated time slots, neglecting content requests that persist across multiple time slots, such as those affected by satellite handovers. Managing such scenarios requires a caching and distribution strategy that can adapt to the dynamic transitions inherent to satellite networks. Another significant challenge is orchestrating DAI and content caching efficiently. Centralized orchestration, whether at a GEO satellite or a ground control station, often struggles with high latencies and limited adaptability to topology changes and real-time response demands. On the other hand, fully distributed approaches, where decision-making occurs at individual cache nodes, can result in suboptimal performance due to limited global visibility, increased communication overhead, and the added complexity of online decision-making for new unassociated requests. Therefore, a balanced, collaborative approach is needed to optimize monetization and ensure seamless user experiences.

To address the aforementioned challenges, this work proposes a revenue-driven and seamless content caching and distribution scheme in STINs, designed to ensure service continuity and support Distributed DAI (D-DAI) for maximizing revenue generation. Unlike existing approaches, this work integrates user disengagement modeling, distributed ad stitching, and service continuity awareness into a unified optimization framework specifically designed for STINs, accounting for challenges such as time-varying network topology and limited edge resources. The user disengagement cost is modeled using four engagement sensitivity factors: content type, ad runtime duration, ad-content language misalignment[1], and user demographics. To the best of our knowledge, existing research does not address the integration of content caching with dynamic ad insertion, despite its critical importance for maximizing revenue. One of the key innovations of this work is the design of a distributed SSAI scheme, where ads are stitched at the node caching the primary content, significantly reducing feeder link load and storage overhead by avoiding redundant storage of multiple content-ad variants. The proposed approach includes algorithms for optimizing content caching placement, greedy content distribution with DAI, enhanced content distribution through Binary Particle Swarm Optimization (BPSO), and managing unassociated content requests in a distributed real-time decision-making framework. Our contributions are summarized as follows:

1) We formulate the problem as a hierarchical Integer Linear Programming (ILP) optimization. The first stage addresses service continuity–aware content caching placement, aiming to maximize the cache hit rate while accounting for edge resource constraints and the time-varying topology of the STIN. The second stage focuses on revenue-driven content distribution with DAI, aiming to maximize provider revenue and minimize end-user costs, while considering resource constraints, topological dynamics of STINs, and user disengagement factors. Furthermore, we present a distributed SSAI-based ad stitching architecture, in which ad insertion is implemented as a network function at cache nodes, leveraging their local computing capabilities for efficient and scalable execution.
2) The proposed hierarchical ILP formulation is NP-hard due to its combinatorial nature. To address this computational complexity, we propose a greedy-based algorithm to efficiently generate near-optimal solutions, followed by a BPSO-based approach to further refine solution quality. Additionally, we present a distributed, real-time decision-making scheme to handle unassociated requests through proximity-based collaboration among edge nodes, thereby improving overall system performance and responsiveness.
3) We conduct extensive simulations using realistic network parameters and datasets to evaluate our solutions under varying cache node storage capacities and dynamic topologies. The simulation results demonstrate that our approach approximates the optimal *ILP* solution while outperforming existing benchmarks in terms of revenue and cache retrieval duration.

The remainder of the paper is organized as follows. Section II presents an overview of contemporary and related work. Section III explains the system model and problem formulation. Section IV presents the proposed approach designed to efficiently solve the *ILP* problem. In Section V, we present simulation settings and results to demonstrate the effectiveness of the proposed solution. Finally, Section

[1]Ad-content language misalignment occurs when the language of the ad does not match the language of the primary content.

VI concludes the paper, offering insights into potential future research directions.

II. Related Works

In this section, we review the existing approaches for content caching and distribution and DAI in STINs.

A. Content Caching and Distribution in STINs

With advances in in-orbit processing and caching, recent studies have focused on content caching and distribution in satellite networks to reduce latency and improve user experience. However, most of these strategies are often developed independently as cache placement and content delivery approaches. Zhu et al. [6] presented a delivery delay minimization strategy for cache placement. In our previous work [14], we studied resource-efficient cache updates considering reconfiguration overhead. However, these works primarily focus on content cache placement, neglecting the critical role of content distribution. In contrast, Bhandari *et al.* [15] proposed a content delivery scheme to minimize the delivery delays. Tang *et al.* [16] proposed a routing scheme using satellite cached content. However, both approaches focus on content delivery but overlook cache placement methods. A unified strategy for joint content caching and distribution is essential for efficient resource utilization and user satisfaction in dynamic STINs. Relying solely on one of these strategies fails to optimize resources or ensure a seamless user experience. In this regard, Jiang *et al.* [17] proposed a density-based cache distribution strategy for STINs, considering dynamic satellite topology and uneven user distribution. Similarly, the authors of [5] studied collaborative caching between LEO satellites and distribution approach, modeling the time-varying topology as a time-varying graph. Shushi *et al.* [18] explored a layered coded cache placement and distribution strategy, enabling collaborative distribution between satellites and base stations. Wei *et al.* [4] proposed UAV-assisted joint caching and user selection to maximize satellite residual energy under UAV resource constraints. Zhu et al. [6] proposed cooperative content placement using dynamic programming and submodular optimization to minimize content retrieval delay.

Although existing works address time-varying topology by analyzing content requests and network topology on a per-time-slot basis, assuming a quasi-static topology within each slot [4]–[6], [17], [18], they do not consider services that persist across multiple time slots. Additionally, these approaches typically rely on centralized orchestrators and periodic decisions, which struggle to manage real-time content distribution when unassociated requests arrive. Content distribution decisions are made at the start of each time slot, with cache misses resulting in requests being forwarded to the cloud for additional processing. Our approach proposes a joint collaborative distributed content caching and distribution scheme that optimizes resource utilization and ensures service continuity. Unlike existing strategies, when a cache miss occurs, our solution enables nodes to collaborate locally with nearby nodes for content delivery, eliminating the need to forward requests to the cloud and significantly enhancing the user experience.

B. DAI in STINs

Advertising is vital for monetizing content delivery, driving standardization efforts like SCTE-35 and SCTE-104 [19] and advancements in ad insertion architectures such as SSAI and CSAI solutions [9]. For example, Pham *et al.* [20] integrated ad insertion standards with MPEG DASH for HTML5-based platforms to enable interoperable ad insertion. Similarly, the authors of [19] explored ad substitution using SCTE-35 in DASH workflows for over-the-top (OTT) services. On the other hand, the authors of [9] proposed SSAI architectures in cloud-based and cloud-assisted content delivery to enhance ad personalization. Seeliger *et al.* [21] demonstrated DAI in OTT streaming workflows. While ad insertion generates revenue for the service provider, disruptions in content delivery due to excessive ads or poor timing can lead to user disengagement and service termination. Balancing monetization and user engagement is key to maximizing revenue while minimizing disengagement. However, existing works do not focus on ad insertion schemes in content distribution within STINs.

The SSAI architecture is preferred over CSAI for its robustness against ad blockers, as it stitches ads directly at the cloud server [9]. Traditional SSAI-based schemes require ads inserted before caching content on satellites, gateways, and user terminals to ensure personalized delivery. However, these methods consume substantial feeder link bandwidth by offloading and caching multiple versions of content with different ads, which strains link capacity and increases storage demands. In contrast, our approach performs ad stitching directly at the cache node where the primary content is stored, treating ad insertion as a network function and leveraging the computing capabilities of these nodes to execute the function [22]–[24] and ensure seamless service delivery. This reduces both storage requirements and feeder link loads in a distributed manner, enabling efficient, real-time ad insertion and seamless personalized content delivery. However, MEC-enabled nodes such as satellites, gateways, and user terminals have limited computing and storage resources, requiring a strategy to manage DAI in STINs. Accordingly, in this work, we mathematically model and optimize seamless content cache distribution with DAI, using a collaborative distributed strategy for real-time content delivery in STINs.

While deep reinforcement learning (DRL) offers strong adaptability for dynamic optimization, it demands high computational resources and long training times. Federated learning (FL), on the other hand, incurs communication and synchronization overheads, making it unsuitable for the intermittent and bandwidth-limited nature of STINs. The proposed hybrid greedy-based BPSO algorithm achieves faster convergence with lower complexity, making it well-suited for resource-constrained and delay-sensitive MEC-enabled STINs.

III. System Model and Problem Formulation

This section presents the mathematical model for collaborative revenue-driven content distribution with DAI in an MEC-enabled STIN. For clarity, the notations and decision variables used throughout the paper are summarized in Tables I and II, respectively.

Table I: Table of Symbols

Symbol	Definition
t	The t^{th} time slot
$G(t)$	Network topology at t
$\mathbb{N}$	Set of nodes in the network
C_n	Processing capacity of $n \in \mathbb{N}$
β_n	Storage capacity of $n \in \mathbb{N}$
$\mathbb{V}$	Set of VSATs (user terminal)
Δt	Time slot duration
$\mathcal{N}_n(t)$	Set of neighboring nodes of $n \in \mathbb{N}$ at t
$\mathbb{E}(t)$	Set of links at t
$\Omega_k^h(t)$	Bandwidth capacity of $e_k^h \in \mathbb{E}(t)$ at t
$D_k^h(t)$	Propagation delay $e_k^h \in \mathbb{E}(t)$ at t
$\gamma_k^h(t)$	Availability of $e_k^h \in \mathbb{E}(t)$ at t
$\mathscr{E}_v^n$	Set of links in shortest path between $v \in \mathbb{N}$ and $n \in \mathbb{N}$ at t
$\mathbb{Q}(t)$	Set of user content requests at t
$node^{i,src}$	Source VSAT for $q_i \in \mathbb{Q}(t)$
r^i	Minimum data rate requirement of $q_i \in \mathbb{Q}(t)$
$\psi^i(t)$	Premium subscription of of $q_i \in \mathbb{Q}(t)$
$\mathscr{F}^i$	Content item requested by of $q_i \in \mathbb{Q}(t)$
$\mathbb{F}$	Set of content items
b^f	Size of $f \in \mathbb{F}$
L^f	Packet size of $f \in \mathbb{F}$
$\mathbb{A}$	Set of ad content items
σ^a	Display duration of $a \in \mathbb{A}$
μ^a	Processing capacity requirement of $a \in \mathbb{A}$
π^f	Service subscription price of $f \in \mathbb{F}$ per bit
$\hat{\pi}^a$	Price per impression for $a \in \mathbb{A}$
$p_n^f(t)$	Popularity of $f \in \mathbb{F}$ at $n \in \mathbb{N}$ at t
$\mathscr{D}_{min}$	Minimum gap between two ad break points in a request

Table II: Decision Variables

Symbol	Definition
$x_n^f(t)$	Binary variable indicating whether $f \in \mathbb{F}$ is cached at $n \in \mathbb{N}$ at t.
$y_{v,n}^f(t)$	Binary variable indicating whether $n \in \mathbb{N}$ serves requests for $f \in \mathbb{F}$ from $v \in \mathbb{V}$ at t.
$z_{v,a}^f(t)$	Binary variable indicating whether $a \in \mathbb{A}$ is inserted into requests for $f \in \mathbb{F}$ from $v \in \mathbb{V}$ at t.

A. System Model

As illustrated in Figure 1, we consider a multi-layer STIN supporting content caching, distribution, and ad insertion. The satellite network comprises SEC-enabled LEO and GEO nodes, facilitating communication through intra-orbit inter-satellite links (ISLs) and inter-orbital links Radio Frequency (RF) or Free Space Optical (FSO) links. The terrestrial network includes a cloud-based content provider (CP), a cloud-based ad provider server (AP), MEC-enabled satellite gateways, and local networks with MEC-enabled terminals like Very Small Aperture Terminals (VSATs). Each gateway is connected to CP and AP via fiber optic cables and communicates with satellites via FSO uplinks. Remote users access the network through RF uplinks using VSATs, ensuring coverage by at least one satellite. GEO satellites collect caching and resource information across the network through the corresponding gateways, distributing content to MEC-enabled ground user terminals and LEO satellites, which handle caching and delivering content to end users. This framework focuses on remote users lacking terrestrial access. We consider VSATs with multi-connectivity capability to track multi-orbit LEO and GEO satellites [25]. A key challenge is the evolving network topology due to the mobility of satellites in LEO constellations, which impacts strategy design. To address this, the topology is divided into sequential time slots, during which remains quasi-static.

The revenue-driven content delivery system comprises four key entities. The *SatCom operator* provides the infrastructure for caching and delivering primary and ad content, including high-quality video caching, transcoding, and personalized ad-stitching. *Content providers*, such as OTT platforms like YouTube, Netflix, and Hulu supply primary content based on user preferences. *Ad content providers* contribute ads to be inserted alongside primary content, promoting their products or services to users. Lastly, the *customers* are end users consuming both primary content and ads. Content providers lease Content Delivery Network (CDN) nodes as a network slice from the SatCom operator to cache content and deploy ad-stitching network functions on satellites, gateways, and VSATs near end users. The service provider also integrates ads from ad providers based on Service Level Agreements (SLAs) to maximize revenue. A distributed set of SDN controllers orchestrates content cache distribution and ad insertion based on data from each gateway, and any standard SDN controller placement algorithm for satellite networks [26] can be used to determine their location in our use case.

1) Network Model

We model the time-varying STIN topology as sequential time slots, each with a duration of Δt time units ($\mathbb{T} = \langle 1, \cdots, t, \cdots, T \rangle$), where T is the recurrence period. During each t, the topology is considered quasi-static and represented by an undirected graph $G(t) = (\mathbb{N}, \mathbb{E}(t))$, where $\mathbb{N}$ denotes the set of all nodes (satellites, gateways, VSATs, content provider server, and ad provider server) and $\mathbb{E}(t)$ represents the set of links within the time slot, connecting the nodes in the network. Each node $n \in \mathbb{N}$ is represented by a tuple $\langle \beta_n, \mathrm{C}_n, \mathcal{N}_n(t) \rangle$, where β_n, C_n, and $\mathcal{N}_n(t)$ represents its storage capacity, computing capacity, and set of neighboring nodes at t, respectively. We define $\mathbb{V} \subseteq \mathbb{N}$ as the set of user terminals. Additionally, we represent the cloud content and ad provider servers by c_p and a_p, respectively. A link $e_k^h \in \mathbb{E}(t)$ between nodes $h \in \mathbb{N}$ and $k \in \mathbb{N}$ at t is defined by $\langle \gamma_k^h(t), \Omega_k^h(t), D_k^h(t) \rangle$, representing its availability, bandwidth, and propagation delay, respectively. A link between nodes is available only if they are visible to each other at a given time slot. The visibility between nodes h and k is defined by their line of sight (LoS), as discussed in our previous

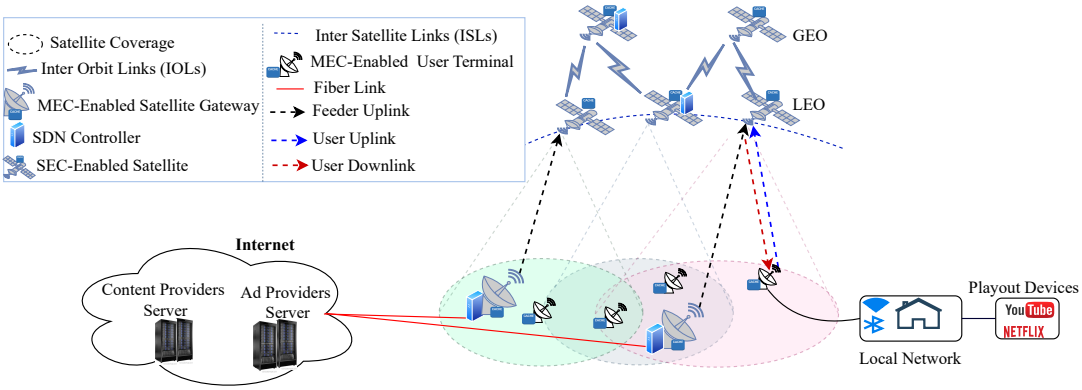


Figure 1: MEC-enabled STIN for supporting content caching and distribution with DAI.

work [14]. We define $\gamma_k^h(t)$ as an indicator function, where $\gamma_k^h(t) = 1$ if e_k^h is available at time t, and $\gamma_k^h(t) = 0$ otherwise. The set of links in the shortest path between nodes h and k at t is represented by a set of links $\mathscr{E}_k^h(t) \in \mathbb{E}(t)$, obtained by *Dijkstra's shortest path* algorithm.

2) Content Request and Ad Insertion Model

We consider a set of content items, denoted as $\mathbb{F}$, which are initially stored on c_p and subsequently cached in nodes closer to end users based on their requests and resource availability. Each content $f \in \mathbb{F}$ is characterized by its storage requirement b^f and packet size L^f, both measured in bits. In addition, we consider a set of ads denoted as $\mathbb{A}$, stored on a_p. Each ad $a \in \mathbb{A}$ is defined by its display duration σ^a (in seconds) and computing requirement μ^a (in vCPUs). We consider a mid-roll ad insertion scheme [27], where ads are stitched after the primary content display starts. Ad insertion takes place at the cache node, which stores the primary content, provided it has sufficient computing capacity. To ensure seamless playback and bypass ad blockers, we adopt a distributed SSAI approach, stitching ads before content delivery for uninterrupted ad display [9].

We consider proximity-based content probability model [14] to define the popularity of a content item $f \in \mathbb{F}$ at a node $n \in \mathbb{N}$ at t, denoted as $p_n^f(t)$. We define $\mathbb{Q}(t)$ as the set of new and ongoing content requests at t. Each request $q_i \in \mathbb{Q}(t)$ is defined as $\langle node^{i,src}, node^{i,dst}, \mathscr{F}^i, r^i, \mathscr{D}^i, \psi^i, t^{i,arr}, \Upsilon^i(t) \rangle$, where $node^{i,src} \in \mathbb{V}$ denotes a source VSAT connected to the local network where the request originates. $node^{i,dst} = c_p$ is a destination node. $\mathscr{F}^i \in \mathbb{F}$ denotes requested content item, r^i is the minimum data rate required to deliver the content, $\mathscr{D}^i$ indicates the estimated display duration, $\psi^i \in \{0,1\}$ is a binary variable indicating the user's subscription status (1 for premium and 0 for non-premium), and $t^{i,arr}$ specifies the arrival time of the request. Furthermore, $\Upsilon^i(t)$ indicates whether the request q_i continues from time slot t to the subsequent time slot $t+1$. Mathematically,

$$\Upsilon^i(t) = \begin{cases} 1, & \text{if } \mathscr{D}^i \geq t\Delta t - t^{i,arr} \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

We assume that all content requests of a specific type have the same data rate requirement. Additionally, all requests from a given VSAT for the same content type are associated with a single cache node. We assume that all such requests from the same VSAT share a common subscription status value for mathematical simplicity. We define $\zeta_{v,i}^f(t)$ if request $q_i \in \mathbb{Q}(t)$ is for $f \in \mathbb{F}$ from $v \in \mathbb{V}$ at t expressed as follows.

$$\zeta_{v,i}^f(t) = \begin{cases} 1, & \text{if } \mathscr{F}^i = f \text{ and } node^{i,src} = v \text{ at t} \\ 0, & \text{otherwise.} \end{cases} \tag{2}$$

We define the binary decision variable $x_n^f(t)$ as the content caching strategy, indicating whether $f \in \mathbb{F}$ is cached on $n \in \mathbb{N}$ at t.

$$x_n^f(t) = \begin{cases} 1, & \text{if the content } f \text{ is cached on node } n \text{ at } t, \\ 0, & \text{Otherwise.} \end{cases} \tag{3}$$

The binary decision variable $y_{v,n}^f(t)$ represents the user-cache association (content distribution) strategy, indicating whether cache node $n \in \mathbb{N}$ serves content requests from user terminal VSAT $v \in \mathbb{V}$ for content $f \in \mathbb{F}$ at t,

$$y_{v,n}^f(t) = \begin{cases} 1, & \text{if } n \text{ serves requests from } v \text{ for } f \text{ at } t. \\ 0, & \text{otherwise.} \end{cases} \tag{4}$$

We set $node^{i,dst} = n$ if $y_{v,n}^f(t) = 1$ for request $q_i \in \mathbb{Q}(t)$ for content $f \in \mathbb{F}$ is generated from VSAT user terminal $v \in \mathbb{V}$. Furthermore, we define a binary decision variable $z_{v,a}^f(t)$ as the DAI strategy, indicating whether ad $a \in \mathbb{A}$ is inserted into the content requests for content $f \in \mathbb{F}$ and requested via the VSAT user terminal $v \in \mathbb{V}$ during t.

$$z_{v,a}^f(t) = \begin{cases} 1, & \text{if } a \text{ is inserted into requests for } f, \text{ via } v \text{ at } t, \\ 0, & \text{otherwise.} \end{cases} \tag{5}$$

3) Content Delivery Delay Model

The content delivery delay is defined as the total time required for a content item to be delivered from a designated cache node to a requesting user terminal. It includes propagation, transmission, and queuing delays experienced along the shortest path between the cache node and the source VSAT where the content request originates.

3.1. Propagation Delay

The propagation delay component of the content delivery delay is defined as the cumulative sum of individual propagation delays across all links in the shortest path between

the cache node and the requesting VSAT. For a content request $q_i \in \mathbb{Q}(t)$, where the requested content is $f \in \mathbb{F}$, originating from user terminal $v \in \mathbb{V}$ and served by cache node $n \in \mathbb{N}$ at time t, the propagation delay is given by:

$$d_{i,n}^{prop}(t) = \sum_{e_k^h \in \mathscr{E}_v^n(t)} D_k^h(t)\, y_{v,n}^f(t), \tag{6}$$

where $\mathscr{E}_v^n(t)$ represents the set of links that form the shortest path between the selected cache node n and the source VSAT node v at time t.

3.2. Transmission Delay

Transmission delay is the time required to send a packet over a link, depending on the packet size and link bandwidth. The transmission delay for content item f over a link e_k^h at t is $\frac{L^f}{\Omega_k^h(t)}$, where L^f is the packet size in $bits$ and $\Omega_k^h(t)$ is the link bandwidth in $\frac{bits}{second}$. The transmission delay for a content request $q_i \in \mathbb{Q}(t)$, corresponding to content $f \in \mathbb{F}$, originating from a user terminal $v \in \mathbb{V}$ and served by a cache node $n \in \mathbb{N}$, is denoted as $d_{i,n}^{txt}(t)$. It is determined by the available bandwidth of the bottleneck (i.e., the link with the minimum bandwidth) along the shortest path between n and v. The transmission delay is defined as:

$$d_{i,n}^{txt}(t) = \frac{L^f}{\min\limits_{e_k^h \in \mathscr{E}_v^n(t)} \left(\Omega_k^h(t)\right)} y_{v,n}^f(t), \tag{7}$$

where $\min\limits_{e_k^h \in \mathscr{E}_v^n(t)} \left(\Omega_k^h(t)\right)$ denotes the bottleneck bandwidth, *i.e.,* the minimum available bandwidth among all links along the shortest path between the designated cache node n and the requesting VSAT v at time t. This bottleneck bandwidth determines the maximum achievable transmission rate along the end-to-end delivery path.

3.3. Queuing Delay

We assume an *M/M/1* queuing model with a single server, where arrivals follow a Poisson process and service times are exponentially distributed. The queuing delay is the time a request waits at the cache node before being served. Let $\lambda_n(t)$ denote the arrival rate in requests per millisecond ($\frac{\text{requests}}{\text{ms}}$) and $\mu_n(t)$ the service rate in the same unit at the serving cache node $n \in \mathbb{N}$. To simplify the analysis, the queuing delay is modeled as the mean waiting time for a content request $q_i \in \mathbb{Q}(t)$ for content item $f \in \mathbb{F}$, issued from user terminal $v \in \mathbb{V}$ and served by cache node n at time t. This delay is denoted by $d_{i,n}^{qq}(t)$.

$$d_{i,n}^{qq}(t) = \frac{\rho_n(t)}{\mu_n(t)\big(1-\rho_n(t)\big)} y_{v,n}^f(t), \tag{8}$$

where $\rho_n(t) = \frac{\lambda_n(t)}{\mu_n(t)}$ denotes the utilization of cache n at t.

The overall delivery delay $d_{i,n}^{tot}(t)$ for a content request $q_i \in Q(t)$ served by a cache node $n \in \mathbb{N}$ at time slot t is defined as the cumulative sum of its constituent delay components along the delivery path:

$$d_{i,n}^{tot}(t) = d_{i,n}^{prop}(t) + d_{i,n}^{txt}(t) + d_{i,n}^{qq}(t). \tag{9}$$

4) Content Provider Revenue and Delivery Cost Model

The content provider's revenue from content delivery comprises two main components: service subscription and advertising revenues. Service subscription revenue comes from users paying for services like internet access via telecom providers, while ad revenue is earned through advertisement monetization. We consider the service subscription revenue as a usage-based, pay-as-you-go model [28], where pricing depends on data consumption. The service subscription revenue associated with a content request $q_i \in \mathbb{Q}(t)$, for content item $f \in \mathbb{F}$ requested through a VSAT terminal $v \in \mathbb{V}$, is denoted as $\Pi_i^{\text{sub}}(t)$. It quantifies the revenue earned by serving the request under a subscription-based model and is mathematically expressed as:

$$\Pi_i^{sub}(t) = \pi^f b^f y_{v,n}^f(t), \tag{10}$$

where π^f is the subscription price of content item f per bit. Additionally, we adopt the Cost-per-Impression (CPI) advertisement revenue model [29], where advertisers are charged based on the number of ad impressions delivered to end users. The advertising revenue associated with a content request $q_i \in \mathbb{Q}(t)$, requesting content $f \in \mathbb{F}$ via a VSAT terminal $v \in \mathbb{V}$, is denoted by $\Pi_i^{\text{adv}}(t)$ and is formulated as:

$$\Pi_i^{adv}(t) = \sum_{a \in \mathbb{A}} \hat{\pi}^a z_{v,a}^f(t), \tag{11}$$

where $\hat{\pi}^a$ is the price per impression for ad item a.

We model the probability of user disengagement using a combination of the Kaplan–Meier estimator [30] and an exponentially decaying survival function [31]. This hybrid modeling approach captures the likelihood that a user continues to engage with the content over time, where the survival probability decreases exponentially with increasing levels of engagement sensitivity. Key sensitivity factors include: Content type (e.g., live vs. on-demand), User demographics (e.g., age-based ad tolerance), Ad-content language misalignment (when the language of the ad differs from that of the primary content), Ad runtime relative to the total content duration. The model outputs higher disengagement probabilities when these factors are poorly aligned with user preferences. Consequently, lower survival values indicate a higher likelihood of early termination or content abandonment. The disengagement cost incurred by the content provider for serving a request $q_i \in \mathbb{Q}(t)$, corresponding to content $f \in \mathbb{F}$, requested via user terminal $v \in \mathbb{V}$ at time t, is denoted as $\Pi_i^{dis}(t)$, and is mathematically expressed as:

$$\Pi_i^{dis}(t) = \pi^f b^f \Big(1 - \sum_{a \in \mathbb{A}} e^{-\kappa_i^0 \left(\kappa_i^t + \kappa_i^d + \kappa_i^m + \frac{a^\sigma}{\Delta t}\right)} z_{v,a}^f(t)\Big) \tag{12}$$

where κ_i^0 is the baseline disengagement rate, capturing the general likelihood of disengagement in the absence of sensitivity factors, κ_i^t represents the content type sensitivity, where content such as live broadcasts incurs higher sensitivity than on-demand videos, κ_i^d denotes the demographic factor, reflecting variations in ad tolerance across user groups (e.g., younger users being more ad-averse), κ_i^m captures ad-content language misalignment, modeling the negative impact when the language of the ad differs from that of the primary content, κ_i^r accounts for ad runtime sensitivity, representing the impact of ad length relative to the overall content duration. Each of these κ-parameters contributes additively to the overall decay rate

in the exponential survival function, such that higher values indicate a faster drop in user engagement. This model quantifies the revenue loss for content providers due to user disengagement. We define $\eta^i(t)$ as a binary variable that indicates if a request $q_i \in \mathbb{Q}(t)$ is disengaged or not.

$$\eta^i(t) = \begin{cases} 1, & \text{if } \sum_{a\in\mathbb{A}} e^{-\kappa_i^0\left(\kappa_i^t+\kappa_i^d+\kappa_i^m+\frac{a^\sigma}{\Delta t}\right)} z_{v,a}^f(t) \geq 1, \\ 0, & \text{otherwise.} \end{cases} \tag{13}$$

The total net revenue for delivering a request $q_i \in \mathbb{Q}(t)$, which seeks content $f \in \mathbb{F}$ from user terminal $v \in \mathbb{V}$ at time t, can be derived from Equations (10), (11), and (12), and is given by:

$$\Pi_i^{tot}(t) = \Pi_i^{sub}(t) + \Pi_i^{adv}(t) - \Pi_i^{dis}(t) \tag{14}$$

The content delivery cost represents the revenue penalty for distributing content to the end user. For request $q_i \in \mathbb{Q}(t)$, requesting for content $f \in \mathbb{F}$ from user terminal $v \in \mathbb{V}$ served by cache node $n \in \mathbb{N}$ at t is:

$$\Pi_i^{cos}(t) = \frac{\Pi_i^{sub}(t)}{d_{i,c_p}^{tot}(t)} d_{i,n}^{tot}(t), \tag{15}$$

where $d_{s,c_p}^{tot}(t)$ is the content delivery delay from the cloud server to the end user during t (when $z_{v,n}^f(t) = 1 \big| n = c_p$).

B. Problem Formulation

In this paper, we propose a multi-level hierarchical optimization approach to address content cache placement and content distribution with DAI.

1) Content Cache Placement Problem

The objective is to maximize the system-wide cache hit rate at a given time slot by storing frequently requested content on cache nodes. This minimizes distribution delay and delivery costs by allowing users to retrieve content from nearby caches. Additionally, the goal is to ensure service continuity across time slots, reducing service disruption. The optimization problem is formulated as follows:

$$\mathbf{P}_1 : \underset{\mathbf{X}(t)}{\text{Maximize}} \sum_{f\in\mathbb{F}} \sum_{n\in\mathbb{N}} p_n^f(t) \cdot x_n^f(t), \tag{16a}$$

$$\text{s.t.} \quad \sum_{f\in\mathbb{F}} b^f x_n^f(t) \leq \beta_n, \quad \forall n \in \mathbb{N}, \tag{16b}$$

$$\sum_{n\in\mathbb{N}} x_n^f(t) \geq 1, \quad \forall f \in \mathbb{F}, \tag{16c}$$

$$x_n^f(t) \geq \sum_{q_i\in\mathbb{Q}(t-1)} \frac{y_{v,n}^f(t-1) \cdot \Upsilon^i(t-1)}{|\mathbb{Q}(t-1)|} \zeta_{v,i}^f(t-1), \quad \forall f \in \mathbb{F}, \forall n \in \mathbb{N}, \tag{16d}$$

$$\sum_{k\in\left\{\mathcal{N}_n(t)\cup n\right\}} x_k^f(t) \leq 1, \ \forall n \in \mathbb{N}, \ \forall f \in \mathbb{F}, \tag{16e}$$

$$x_n^f(t) \in \{0,1\}, \quad \forall n \in \mathbb{N}, \forall f \in \mathbb{F}, \tag{16f}$$

Constraint (16b) limits stored content to each cache node's capacity. Constraint (16c) ensures every required content item is available on at least one node, including at c_p. Constraint (16d) mandates that content items serving ongoing requests from the previous time slot remain cached until those requests are completed. Constraint (17e) ensures that only one type of content is cached on adjacent satellites, optimizing ISL and storage usage. Constraint (16f) enforces the binary nature of decision variables. The formulated optimization problem $\mathbf{P}_1$ is classified as an *ILP* problem.

2) Content Distribution Problem

This work aims to maximize content provider revenue while minimizing end-user content delivery cost in a given time slot. The goal is to maximize revenue while minimizing user dissatisfaction, influenced by transmission costs and delivery delays. Mathematically,

$$\mathbf{P}_2 : \underset{\mathbf{Y}(t),\mathbf{Z}(t)}{\text{Maximize}} \sum_{q_i\in\mathbb{Q}(t)} \Big(w_1 \Pi_i^{tot}(t) - w_2 \Pi_i^{cos}(t) \Big), \tag{17a}$$

$$\text{s.t.} \quad y_{v,n}^f(t) \leq x_n^f(t), \quad \forall f \in \mathbb{F}, \forall n \in \mathbb{N}, \forall v \in \mathbb{V}, \tag{17b}$$

$$z_{v,a}^f(t) \leq \sum_{n\in\mathbb{N}} y_{v,n}^f(t), \quad \forall a \in \mathbb{A}, \forall f \in \mathbb{F}, \forall v \in \mathbb{V}, \tag{17c}$$

$$\sum_{q_i\in\mathbb{Q}(t)} r^i \zeta_{v,i}^f(t) y_{v,n}^f(t) \leq \Omega_k^h(t), \quad \forall f \in \mathbb{F}, \forall n \in \mathbb{N}, \forall v \in \mathbb{V}, \forall e_k^h \in \mathscr{E}_v^n(t), \tag{17d}$$

$$\sum_{n\in\mathbb{N}} y_{v,n}^f(t) \leq 1, \quad \forall f \in \mathbb{F}, \forall n \in \mathbb{N}, \forall v \in \mathbb{V}, \tag{17e}$$

$$y_{v,n}^f(t) \leq \frac{1}{N_v^f(t)} \sum_{q_i\in\mathbb{Q}(t)} \frac{\gamma_k^h(t)\big(1+\gamma_k^h(t+1)\Upsilon^i(t)\big)}{1+\Upsilon^i(t)} \zeta_{v,i}^f(t), \quad \forall f \in \mathbb{F}, \forall n \in \mathbb{N}, \forall v \in \mathbb{V}, \tag{17f}$$

$$z_{v,a}^f(t) \leq 1 - \frac{1}{N_v^f(t)} \sum_{q_i\in\mathbb{Q}(t)} \zeta_{v,i}^f(t) \psi^i(t), \quad \forall a \in \mathbb{A}, \forall f \in \mathbb{F}, \forall v \in \mathbb{V}, \tag{17g}$$

$$\sum_{a\in\mathbb{A}} z_{v,a}^f(t) \leq \frac{\Delta t}{\mathscr{D}_{min}}, \quad \forall a \in \mathbb{A}, \forall f \in \mathbb{F}, \forall v \in \mathbb{V}, \tag{17h}$$

$$\mathscr{D}^i \zeta_{v,i}^f(t) \geq \mathscr{D}_{min} z_{v,a}^f(t), \quad \forall q_i \in \mathbb{Q}(t), \forall a \in \mathbb{A}, \forall f \in \mathbb{F}, \forall v \in \mathbb{V}, \tag{17i}$$

$$z_{v,a}^f(t) \mu^a \leq C_n y_{v,n}^f(t), \quad \forall f \in \mathbb{F}, \forall n \in \mathbb{N}, \forall v \in \mathbb{V}, \tag{17j}$$

$$\{y_{v,n}^f(t), z_{v,a}^f(t)\} \in \{0,1\}, \quad \forall f \in \mathbb{F}, \forall n \in \mathbb{N}, \forall v \in \mathbb{V} \tag{17k}$$

where w_1 and w_2 are the weighting factors for content provider revenue and content delivery delay cost, respectively, with $w_1 + w_2 = 1$. These weights balance the trade-off between revenue and delivery delay cost. $N_v^f(t) = \sum_{q_i\in\mathbb{Q}(t)} \zeta_{v,i}^f(t)$ is the number of requests for content $f \in \mathbb{F}$ from VSAT $v \in \mathbb{V}$ at t. Constraint (17b) ensures that a user request is served only if the content is cached on the node. Constraint (17c) ensures that ads are inserted only into actively served requests. Constraint (17d) ensures that requests are assigned to cache nodes having sufficient bandwidth on all links on the shortest path between the node and the user. Constraint (17e) guarantees that a request is associated with only one cache node per time slot. Constraint (17f) ensures that content requests spanning from the current time slot to the next are associated only with a node whose shortest-path links to the end user remain available in the subsequent time slot. Constraint (17g) prevents ads from displaying to premium users. Constraint (17h) ensures that ads per content item remain within a predefined threshold. Constraint (17j) ensures that ads are added only to content exceeding a minimum playback duration, $\mathscr{D}_{min}$. Constraint (17i) ensures that ads are inserted only if the node has sufficient processing capacity.

Constraint (17k) states the binary decision variables. The optimization problem in Equation (17) (*i.e.,* $\mathbf{P}_2$) belongs to the class of *ILP* problems. We propose metaheuristic-based schemes for efficient solutions since content caching and distribution problems are *NP-hard*.

IV. The Proposed Scheme

Greedy algorithms are commonly used in satellite-based content caching and distribution [32] due to their low computational complexity, fast convergence, and ease of implementation. These algorithms make locally optimal decisions based on immediate evaluations of the objective function, often achieving acceptable performance in small-scale or time-constrained scenarios. However, they frequently converge to local optima and are generally inadequate for solving large-scale or dynamic problems. On the other hand, Binary Particle Swarm Optimization (BPSO) is effective in solving large-scale problems and can provide near-optimal solutions [33], [34]. BPSO is a population-based metaheuristic inspired by the social behavior of flocks and swarms. In the binary variant, each particle encodes a solution as a binary vector, and its position is updated iteratively using probabilistic rules governed by both personal and global best positions. BPSO is particularly effective for high-dimensional combinatorial problems, such as the ILP-based content distribution with DAI, offering better solution quality compared to greedy algorithm counterparts. However, its convergence can be slow, particularly when initialized with a random population.

We address the hierarchical problem by first solving the content cache placement problem ($\mathbf{P}1$) using the Greedy Enhanced Content Caching (*GE_CC*) algorithm. The solution of $\mathbf{P}1$ is then used as input for the content distribution with the DAI ILP problem ($\mathbf{P}2$). To solve $\mathbf{P}2$, we propose a two-stage hybrid optimization framework that combines the fast initialization of greedy algorithms with the global search capabilities of BPSO. In the first stage, we proposed a Greedy-based Content Distribution with DAI (*G_DAI*) algorithm that generates a fast initial solution. In the second stage, we proposed a BPSO-based approach for Content Distribution with DAI ($BPSO\_DAI$) algorithm, which refines the solution, enhancing accuracy and overcoming local optima to achieve near-optimal performance. This allows the BPSO to start from a solution that is closer to the optimum, unlike the randomly initialized BPSO, which typically begins far from the optimal region of the search space. Although incorporating the greedy algorithm introduces a slight additional delay during the initialization phase, it accelerates the overall convergence of the BPSO by providing a more informed starting point.

A. Greedy Enhanced Content Caching (GE_CC)

The *GE_CC* algorithm optimizes service continuity-aware content cache placement for the ILP problem in Equation (16). Algorithm 1 outlines the workflow of *GE_CC*, which efficiently determines the optimal cache placement strategy $\mathbf{X}^*(t)$ at t. The algorithm takes as input the set of network nodes $\mathbb{N}$, content items $\mathbb{F}$, content requests $\mathbb{Q}(t)$ and $\mathbb{Q}(t-1)$ for the current and previous time slots t and $t-1$ prior optimal cache distribution $\mathbf{Y}^*(t-1)$, the current topology $G(t)$, and the maximum number of iterations $max\_Iteration$.

Algorithm 1 Proposed *GE_CC* Algorithm

Input: $\mathbb{N}$, $\mathbb{F}$, $\mathbf{Y}^*(t-1)$, $\mathbb{Q}(t)$, $\mathbb{Q}(t-1)$, $G(t)$, $max\_Iteration$
Output: $\mathbf{X}^*(t)$: Optimal cache placement strategy at time, t
1: $optimalFitness \leftarrow 0$ ▷ Initialize fitness value.
2: **while** *stopConditionNotMet()* **do**
3: $\quad\mathbf{X} \leftarrow \mathbf{0}^{F\times N}$ ▷ Initialize the cache placement strategy.
4: $\quad\mathbb{N} \leftarrow$ Extract all node sets and randomize their order.
5: $\quad$**for** each node, $n \in \mathbb{N}$ in the network **do**
6: $\quad\quad B_n^{av} \leftarrow \beta_n$ ▷ Set available storage.
7: $\quad\quad$**for** each content $f \in \mathbb{F}$ **do**
8: $\quad\quad\quad x_n^f \leftarrow 1$ ▷ Cache unfinished request content.
9: $\quad\quad\quad B_n^{av} \leftarrow B_n^{av} - b^f$ ▷ Update storage.
10: $\quad\quad$**end for**
11: $\quad\quad$**for** each content item $f \in sortIDdescend(n)$ **do**
12: $\quad\quad\quad$**if** $notInNieghbourNode(f,n)$ **and** $B_n^{av} \geq b^f$ **then**
13: $\quad\quad\quad\quad x_n^f \leftarrow 1$ ▷ Cache the content item.
14: $\quad\quad\quad\quad B_n^{av} \leftarrow B_n^{av} - b^f$ ▷ Update storage.
15: $\quad\quad\quad$**end if**
16: $\quad\quad\quad\mathbf{X} \leftarrow x_n^f$ ▷ Update cache placement strategy.
17: $\quad\quad$**end for**
18: $\quad$**end for**
19: $\quad currentFitness \leftarrow fitness(\mathbf{X})$ ▷ Compute current fitness using Equation (18).
20: $\quad$**if** $currentFitness \geq optimalFitness$ **then**
21: $\quad\quad optimalFitness \leftarrow currentFitness$ ▷ Update optimal fitness.
22: $\quad\quad\mathbf{X}^*(t) \leftarrow \mathbf{X}$ ▷ Update optimal strategy.
23: $\quad$**end if**
24: **end while**
25: $x_{CP}^* \leftarrow 1$ ▷ Cache all content items at cloud provider.
26: **return** $\mathbf{X}^*(t) \leftarrow x_{CP}^*$ ▷ Return optimal strategy.

The algorithm initializes the optimal fitness value to zero (*line 1*). Each iteration generates a potential placement strategy and updates the optimal strategy based on fitness values (*lines 2–24*). The current cache placement strategy is initialized as a zero matrix of size FN (*line 3*) with nodes sorted randomly (*line 4*). Here, F and N denote the total number of content items and network nodes, respectively. Each node's available storage capacity is initialized to its total capacity *(line 6)*. To ensure uninterrupted content delivery, as per Constraint (16d), any cached content still serving at least one request remains cached *(lines 7–10)*. Next, content items are sorted in descending order of popularity at the node. The algorithm prioritizes caching the most popular items, provided they are not cached on neighboring nodes (*Constraint* (16e)) and storage requirements are met (*Constraint* (16b), *lines 11–16*). At each iteration, the strategy with the highest fitness value, computed using Equation (16a), is selected (*lines 19–24*). To ensure content availability as per Constraint (16c), all content items are also cached on the cloud-based content provider server (*line 25*). Finally, the algorithm updates network resources and returns the optimal content placement strategy (*line 26*).

The *stopConditionNotMet()* method determines whether the iteration should continue. It returns *True* as long as either of the following conditions is met: (1) the iteration count has not reached the maximum allowable iterations (*i.e.,* $max\_Iteration$), or (2) the number of consecutive iterations where the fitness values remain nearly unchanged has not exceeded a predefined threshold. The $sortIDdescend(\cdot)$ method takes a node ID as input and returns a set of content items sorted in descending order of popularity at that node. The $notInNeighbourNode(\cdot)$ method accepts a content item and a cache node ID as inputs, returning *True* if the content item is already cached

in one of the neighboring nodes of the specified node and *False* otherwise. We define the fitness function as the objective function of the *ILP* problem, given in Equation (16a), which represents the system's cache hit rate and is expressed as follows:

$$fitness(\mathbf{X}) = \mathbf{P}^T\mathbf{X}, \tag{18}$$

where $\mathbf{P}$ is the content popularity matrix, and $\mathbf{X}$ is the cache placement matrix.

As shown in Algorithm 1, the *GE_CC* algorithm has a time complexity of $\mathcal{O}(max\_Iteration|\mathbb{N}||\mathbb{F}|)$. This indicates that its computational complexity depends on both the size of the network and the number of content items.

Algorithm 2 PROPOSED *G_DAI* ALGORITHM

Input: $\mathbb{N}$, $\mathbb{F}$, $\mathbf{Y}^*(t-1)$, $\mathbb{Q}(t)$, $\mathbb{Q}(t-1), G(t)$, $\mathbb{A}$
Output: $\{\mathbf{Y}^*(t), \mathbf{Z}^*(t)\}$ ▷ Optimal content request-cache association and ad insertion strategies at t.
1: $\mathbf{X} \leftarrow \mathbf{X}^*(t)$ ▷ Update the optimal cache placement strategy via Algorithm 1.
2: $\mathbf{Y} \leftarrow \mathbf{0}^{F\times V\times N}$, $\mathbf{Z} \leftarrow \mathbf{0}^{F\times V\times A}$ ▷ Initialize the user-cache association and ad insertion strategies as a zero matrix.
3: **for** each content request $q_i \in \mathbb{Q}(t)$ **do**
4: $n \leftarrow$ Select the nearest node to $v = n^{i,src}$ that has cached $f = \mathcal{F}^i$ and satisfies Constraints (17b) and (17d)–(17f).
5: **if** $(n \neq \emptyset)$ **then** $y^f_{v,n} \leftarrow 1$
6: $\mathscr{A} \leftarrow$ Select the sets of ad contents randomly that satisfies Constraints (17c) and (17g)–(17j).
7: **if** $(\mathscr{A} \neq \emptyset)$ **then** $z^f_{v,\mathscr{A}} \leftarrow 1$
8: $\mathbf{Y} \leftarrow y^f_{v,n}$, $\mathbf{Z} \leftarrow z^f_{v,\mathscr{A}}$
9: **end for**
10: **return** $\{\mathbf{Y}^*(t), \mathbf{Z}^*(t)\} \leftarrow \{\mathbf{Y}, \mathbf{Z}\}$ ▷ Updates the optimal strategies.

B. Greedy-based Content Distribution with DAI (G_DAI)

We propose the *G_DAI* algorithm to solve the ILP content distribution problem in Equation (17), as outlined in Algorithm 2. The algorithm takes as input set of network nodes $\mathbb{N}$, content items $\mathbb{F}$, content requests $\mathbb{Q}(t)$ and $\mathbb{Q}(t-1)$ for time slots t and $t-1$, the previous content cache distribution strategy $\mathbf{Y}^*(t-1)$, the current network topology $G(t)$, and the ad content items $\mathbb{A}$. It provides a fast initial solution for the *BPSO_DAI* algorithm discussed in subsection IV-C.

Using Algorithm 1, Algorithm 2 first determines the optimal cache placement strategy *(line 1)*. It then initializes the user-cache association and ad insertion strategies as zero matrices of size $F \times V \times N$ and $F \times V \times A$, respectively *(line 2)*. where $F = |\mathbb{F}|$, $V = |\mathbb{V}|$, $N = |\mathbb{N}|$, and $A = |\mathbb{A}|$ denote the total number of content items, user terminals, network nodes, and advertisement items, respectively. For each content request at t, the algorithm uses the *Dijkstra Shortest Path* algorithm to select the nearest node to the user terminal VSAT source address that has cached the requested content item, while satisfying Constraints (17b) and (17d)–(17f) *(line 4)*, and if a node is available, associates the request with the selected cache node *(line 5)*. The algorithm randomly selects a set of ad content items from all possible ads that satisfy Constraints (17c) and (17g)–(17j) *(line 6)* and updates the ad insertion strategy for the selected ad content for the request q_i *(line 7)*. The user cache association and ad insertion strategies are then updated *(line 8)*. Finally, the optimal strategy is updated *(line 10)*.

In Algorithm 2, the *Dijkstra* algorithm is used to compute the shortest paths and has a time complexity of $\mathcal{O}(|\mathbb{E}(t)| + |\mathbb{N}|\log|\mathbb{N}|)$ [35], where $|\mathbb{E}(t)|$ is the number of edges in the network at time slot t. Since *Dijkstra's* algorithm is executed $|\mathbb{Q}(t)|$ times—once for each content request—the overall time complexity of Algorithm 2 becomes $\mathcal{O}(|\mathbb{Q}(t)|(|\mathbb{E}(t)|+|\mathbb{N}|\log|\mathbb{N}|)+max\_Iteration|\mathbb{N}||\mathbb{F}|)$. This indicates that the computational cost scales with both the network size and the number of content requests.

C. Binary Particle Swarm Optimization for Content Distribution with DAI (BPSO_DAI)

BPSO_DAI is a BPSO-based approach for optimizing *ILP* content distribution, offering a more optimal solution than the Greedy-based approach. BPSO aligns with the binary nature of decision variables, making it well-suited for this problem. The *BPSO_DAI* algorithm generates suboptimal solutions for content distribution and ad insertion based on network resources, topology, and content requests for each t, as outlined in Algorithm 3. It takes the following inputs $\mathbb{N}$ (set of nodes), $\mathbb{F}$ (set of content items), $\mathbf{Y}^*(t-1)$ (content cache distribution in previous time slot), $\mathbb{Q}(t)$ and $\mathbb{Q}(t-1)$ (content requests for the current and previous time slots), $G(t)$ (current network topology), $\mathbb{A}$ (set of ad content items), w_1 and w_2 (weights for the objective function). It also requires PSO-specific parameters: w (inertia weight), c_1 and c_2 (cognitive and social learning factors), $\mathcal{N}_{pop}$ (swarm size), and $\mathcal{N}_{gen}$ (maximum iterations).

Algorithm 3 PROPOSED *BPSO_DAI* ALGORITHM

Input: $\mathbb{N}$, $\mathbb{F}$, $\mathbf{Y}^*(t-1)$, $\mathbb{Q}(t)$, $\mathbb{Q}(t-1), G(t)$, $\mathbb{A}$, w_1, w_2, w, c_1, c_2, $\mathcal{N}_{pop}$, $\mathcal{N}_{gen}$
Output: $\{\mathbf{Y}^*(t), \mathbf{Z}^*(t)\}$ ▷ Optimal user-cache association and DAI strategies at t.
1: $\mathbf{X} \leftarrow \mathbf{X}^*(t)$ ▷ Update the optimal cache placement strategy via Algorithm 1.
2: Set $g \leftarrow 0$ ▷ Initialize the current iteration with 0
3: **for** Each particle $\ell = \{1, \cdots, \mathcal{N}_{pop}\}$ **do**
4: **for** each content request $q_i \in \mathbb{Q}(t)$ **do**
5: Initialize particle's position of user-cache association by random selection of a node that has the requested content and satisfying Constraints (17b) and (17d)–(17f).
6: Initialize particle's position of ad insertion by random selection of sets of ads that meet Constraints (17c) and (17g)–(17j).
7: **end for**
8: Initialize the particle's velocity for user-cache association and ad insertion by generating a small random number from a uniform distribution $\mathscr{U}(0,1)$.
9: Update the personal best and save it as $P^\ell_{best}(g)$
10: **end for**
11: Replace the position of the worst performing particle by the optimal solution obtained from Algorithm 2.
12: Update the global best and save it as $G_{best}(g)$.
13: $Converged \leftarrow False$
14: $Count \leftarrow 0$
15: **while** $g \leq \mathcal{N}_g$ and not $Converged$ **do**
16: Set $g \leftarrow g+1$,
17: Update $G_{best}(g) \leftarrow G_{best}(g-1)$
18: **for** Each particle $\ell = \{1, \cdots, \mathcal{N}_{pop}\}$ **do**
19: Compute current velocity of the particle using Equations (20) and (21).
20: Update current position of the particle using Equations (24) and (25).
21: Update the personal best and save it as $P^\ell_{best}(g)$
22: **if** *fitness* of $P^\ell_{best}(g) \leq$ *fitness* of $P^\ell_{best}(g-1)$ **then** $P^\ell_{best}(g) \leftarrow P^\ell_{best}(g-1)$
23: **if** *fitness* of $P^\ell_{best}(g) \geq$ *fitness* of $G_{best}(g)$ **then** $G^\ell_{best} \leftarrow P^\ell_{best}(g)$
24: **end for**
25: **if** $(|$Fitness of $G_{best}(g)-$Fitness of $G_{best}(g-1)| \leq \epsilon_{PSO})$ **then** $Count \leftarrow Count+1$ **else** $Count \leftarrow 0$
26: **if** $(Count \leq C_m)$ **then** $Converged \leftarrow True$
27: **end while**
28: **for** Each unassociated requests $q_u \in \mathbb{Q}(t)$ **do**
29: Perform collaborative user-cache association and ad insertion based on $G_{best}(g)$ and update $G_{best}(g)$.
30: **end for**
31: **return** $(\mathbf{Y}^*(t), \mathbf{Z}^*(t)) \leftarrow G_{best}(g)$ ▷ Update the optimal solution with the global best position.

Algorithm 3 starts by determining the optimal cache placement strategy using Algorithm 1 *(line 1)*. *BPSO_DAI* algorithm initializes the user-cache association and DAI

strategy as zero matrices of size $F \times V \times N$ and $F \times V \times A$, respectively. where $F = |\mathbb{F}|$, $V = |\mathbb{V}|$, $N = |\mathbb{N}|$, and $A = |\mathbb{A}|$ denote the total number of content items, user terminals, network nodes, and advertisement items, respectively. For each content request, the algorithm randomly selects a node from the subset that has cached the requested content, ensuring compliance with Constraints (17b) and (17d)–(17f), and updates the user-cache association accordingly *(line 5)*. Similarly, the algorithm randomly selects a subset of ad content items that satisfy Constraints (17c) and (17g)–(17j), updating the DAI position accordingly *(line 6)*. Then, the algorithm initializes the corresponding velocity for the user-cache association and ad insertion strategies based on their sizes, assigning a very small initial velocity randomly generated from a uniform distribution $U(0,1)$ *(line 8)*. Each particle updates its corresponding personal best based on its respective position *(line 9)*. The algorithm then replaces the worst-performing particle's position with the optimal solution from the *G_DAI* algorithm *(lines 12–16)*, accelerating convergence for fast-response applications. Finally, the algorithm updates the global best position by selecting the best-performing particle in the swarm based on the fitness function. We define the fitness function of a particle, ℓ as the objective function obtained based on the positions of the user-cache, $\mathbf{Y}(t)$, and ad insertion, $\mathbf{Z}(t)$, at t, expressed as follows:

$$fitness(\mathbf{Y},\mathbf{Z}) = \sum_{q_i \in \mathbb{Q}(t)} \Big(w_1 \Pi_\ell^{tot}(t) - w_2 \Pi_\ell^{cos}(t) \Big), \tag{19}$$

While the stopping criterion is not met, the algorithm updates the velocity of each particle as (*line 19*):

$$V_Y^{(\ell)}(g+1) \leftarrow w V_Y^{(\ell)}(g) + c_1 r_1 \big(P_{best,\ell,Y} - \mathbf{Y}^{(\ell)}(g)\big) + c_2 r_2 \big(Gbest_Y - \mathbf{Y}^{(\ell)}(g)\big), \tag{20}$$

$$V_Z^{(\ell)}(g+1) \leftarrow w V_Z^{(\ell)}(g) + c_1 r_1 \big(P_{best,\ell,Z} - \mathbf{Z}^{(\ell)}(g)\big) + c_2 r_2 \big(Gbest_Z - \mathbf{Z}^{(\ell)}(g)\big), \tag{21}$$

where w is the inertia weight, c_1 and c_2 are the cognitive and social coefficients, and r_1 and r_2 are random numbers between 0 and 1. $P_{best,\ell},Y$ and $P_{best,\ell},Z$ are the personal best positions of particle ℓ for the user-cache association and DAI strategies, respectively, $Gbest_Y$ and $Gbest_Z$ are the global best positions for these strategies. $\mathbf{Y}^{(\ell)}$ and $\mathbf{Z}^{(\ell)}$ represent the current positions of particle ℓ. The position of each particle is updated based on its current velocity *(line 20)*. A Sigmoid transformation maps the velocities to probabilities for binary conversion as follows:

$$S_Y\big(V_Y^{(\ell)}(g+1)\big) = \frac{1}{1+e^{-V_Y^{(\ell)}(g+1)}}, \tag{22}$$

$$S_Z\big(V_Z^{(\ell)}(g+1)\big) = \frac{1}{1+e^{-V_Z^{(\ell)}(g+1)}}, \tag{23}$$

The algorithm updates the positions of the user-cache association and DAI strategies using Equations (24) and (25), respectively.

$$Y^{(\ell)}(g+1) = \begin{cases} 1, & \text{if } r \le S_Y\big(V_Y^{(\ell)}(g+1)\big), \\ 0, & \text{otherwise.} \end{cases} \tag{24}$$

$$Z^{(\ell)}(g+1) = \begin{cases} 1, & \text{if } r \le S_Z\big(V_Z^{(\ell)}(g+1)\big), \\ 0, & \text{otherwise.} \end{cases} \tag{25}$$

Next, the algorithm computes the fitness of each particle and updates the personal best position if the current fitness exceeds the particle's previous best *(lines 22)*. Similarly, the global best is updated if a particle outperforms the current global best position of the swarm *(lines 23)*. The global best from the previous iteration is retained if the global best of the current iteration performs worse than the last iteration's global best (*lines 17-24*). This process continues until the stopping criterion is met *(lines 25–26)*, either when the iteration counter reaches its maximum or the best solution remains unchanged for a predefined number of iterations. The system operates in a distributed manner to accommodate unassociated content requests and enable real-time user-cache association and DAI strategies for fast decision-making in dynamic environments. Cache nodes collaborate based on proximity, performing collaborative user-cache association and ad insertion based on $G_{best}(g)$ and updating $G_{best}(g)$ accordingly (*lines 28-30*).

We assume that at the start of each time slot, cache nodes share cache availability information with neighbors to ensure efficient content delivery for unassociated requests. Then, a progressive search is performed to associate a request if there is no existing strategy or locally cached content. Also, randomly selected ads are inserted into the associated request. This method emphasizes a distributed approach before resorting to the cloud provider, unlike existing works that directly associate unassigned requests with the cloud, which improves the system performance significantly. Finally, the optimal user-cache association and ad insertion strategy for t is updated based on the best global position *(line 31)*, and network resources are adjusted accordingly.

The time complexity of the BPSO_DAI algorithm is given by $\mathcal{O}(\mathcal{N}_{pop}\mathcal{N}_g|\mathbb{N}||\mathbb{Q}(t)|) + \mathcal{O}(|\mathbb{E}(t)| + |\mathbb{N}|\log|\mathbb{N}|)$. Therefore, the overall complexity depends on the swarm size, number of generations, network size, and the number of active content requests.

Figure 2 illustrates the overall framework for seamless content caching and distribution with DAI. The framework consists of two main building blocks: (i) content cache placement and (ii) content distribution with DAI. The content cache placement problem is solved using the proposed GE_CC algorithm, and its output serves as input to the second block. In the content distribution with DAI component, the G_DAI algorithm generates a fast initial solution, which is then refined by a BPSO-based optimization strategy to enhance solution quality and ensure efficient ad-aware content delivery.

V. Performance Evaluation

In this section, we evaluate the performance of the proposed solution.

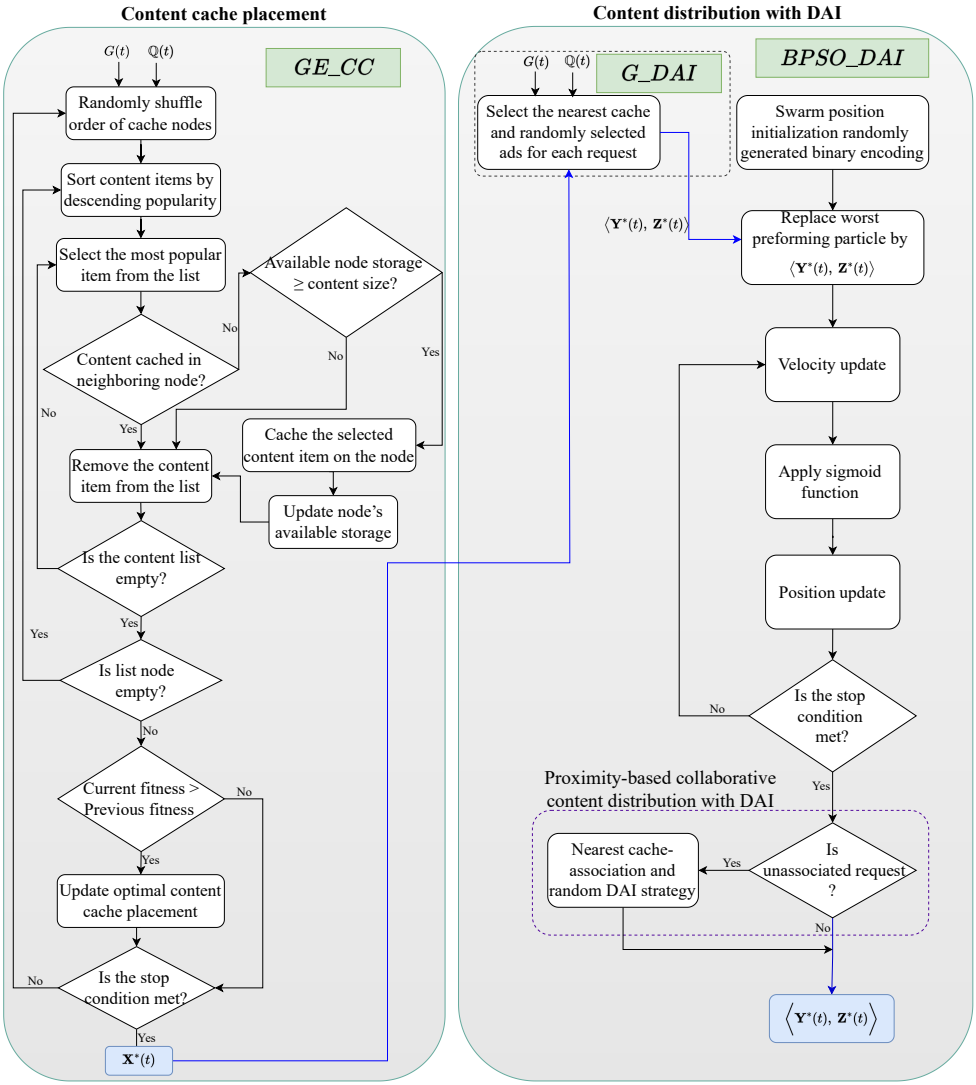


Figure 2: Proposed framework.

A. Simulation Settings

To evaluate the effectiveness of our proposed collaborative caching and distribution strategy with DAI, we developed a simulation framework that reflects the operational characteristics of STINs. We utilized realistic datasets to construct the satellite constellation, including the corresponding gateway locations and user terminal positions. These datasets reflect actual satellite deployments and ground infrastructure, ensuring that our simulation captures the spatial and operational characteristics of real-world STINs. We use the CelesTrak dataset [36] to develop a multi-layer satellite network comprising 3 GEO and 50 LEO satellites. We integrate Iridium NEXT LEO satellites with SES GEO satellites to establish a comprehensive multi-layer satellite network. The data set is a *two-line element (TLE)* file containing the orbital parameters and characteristics of each satellite. The network includes two SES ground stations for the GEO satellites, located in Brewster, Washington, USA, and Winnipeg, Manitoba, Canada [37], along with four Iridium ground stations for the LEO satellites situated in Tempe, Arizona, Fairbanks, Alaska, Svalbard, Norway, and Punta Arenas, Chile [38]. Additionally, we use the *2018 US ZIP Code Geolocation* dataset [39] to map VSAT (user terminals) locations generating content requests. This dataset comprises approximately 40,000 ZIP codes with corresponding geographical coordinates (latitude and longitude) across the United States. The dataset contains three columns: ZIP Code, Latitude, and Longitude. To select 100 user terminals from the 40,000 possible locations uniformly, we apply the *k-means clustering* algorithm [40]. The simulation models the satellite constellation, including the gateways and user terminals, over a 24-hour period (January 1, 2025) using the Satellite Communications Toolbox in MATLAB to capture the temporal dynamics of the STIN topology. The simulation results demonstrate the dynamic evolution of the network topology over time. In particular, a recurring pattern with a cycle of $T = 6$ hours was identified, driven by the periodic nature of satellite-gateway and satellite–user terminal connectivity. Each recurring topology remains stable for approximately 15 minutes, which defines

the duration of a single time slot. Consequently, the entire 24-hour period is divided into $T = 24$ distinct time slots, each corresponding to a unique and temporally stable network configuration.

To ensure the realism and validity of the simulation environment, parameter values and assumptions were adopted based on established findings in the existing literature. Specifically, the ad-stitching capabilities of VSATs and other ground nodes are modeled based on recent studies on MEC-enabled satellite networks, with computing capacities ranging from 120 to 240 vCPUs. In contrast, satellite nodes are assumed to have more constrained computing resources, typically ranging from 48 to 96 vCPUs, as reported in [41], [42]. Similarly, the storage capacities assigned to both ground and satellite nodes in our simulation range from 5 GB to 50 GB, consistent with the values reported in [6]. In our simulations, the content sizes are modeled after high definition (*HD*) video files, ranging between 1 GB and 4 GB [43]. We consider non-skippable advertisements with durations ranging from 6 to 30 seconds [44], and adopt a mid-roll ad insertion scheme. Ad stitching is modeled as an ad insertion Virtual Network Function (VNF), a capability commonly deployed in modern satellite communication networks. The processing requirement for executing this VNF is assumed to range between 1 and 4 vCPUs [42]. For cache eviction behavior, we assume a hierarchical proximity-based popularity-based update policy [45].

We model the content request arrival at each user terminal as an exponential distribution, characteristic of the Poisson arrival process in an $M/M/1$ queuing model, with a mean content request arrival rate in the range of $[0.05, 0.20]$ requests per millisecond [42]. Additionally, we model the popularity of generated content requests using a Zipf distribution with a variable exponent $[0.2, 0.8]$ [6] to capture dynamic content popularity, ensuring adaptability to changing request patterns over time. We consider 400 content items initially stored at a cloud-based content provider (c_p) and 100 ads hosted by a cloud-based ad provider. Content cache placement, user cache association, and ad insertion strategies are dynamically updated per time slot to adapt to network and user requirements. For PSO parameters, we set the inertia $w = 0.9$, cognitive learning factor $c_1 = 2.0$ social learning factors $c_2 = 2.0$, swarm size $\mathcal{N}_{pop} = 50$, and maximum iterations $\mathcal{N}_{gen} = 500$. Furthermore, we randomly generate the values of κ_1, the disengagement rate, and κ_c^i, κ_d^i, and κ_l^i, which represent factors related to content type, user demographics, and ad-content language misalignment, respectively. These values are drawn from a uniform distribution between 0 and 1, denoted as *U(0,1)*, to introduce random variation in the disengagement factors. We consider user playback devices supporting *HD* resolutions. The remaining simulation parameters are listed in Table III.

Table III: Simulation Parameters

Parameters	**Value**
Number of LEO satellites	50 [36]
Number of GEO satellites	3 [36]
User-LEO latency	[20 - 30] ms [46]
LEO-GEO latency	[110- 120] [46]
User-GEO	[120 - 140] ms [46]
Gateway-cloud latency	[100 - 150] ms [45]
Feeder uplink capacity	1 Gbps [47]
Data rate requirement	{3Mbps, 8Mbps, 25Mbps} [48]
Ad price per impression	[20 - 10]$ [49]
Content item pricing	[0.08 - 0.12]$ per GB [50]

B. Benchmark Scheme

To evaluate the proposed algorithm $BPSO\_DAI$, we compare it against three commonly used cache replacement policies [51]: (1) First-In-First-Out (*FIFO*), which evicts the oldest content first; (2) Least Recently Used (*LRU*), which removes the least recently accessed content; and (3) Least Frequently Used (*LFU*), which discards the content with the fewest accesses. We also consider state-of-the-art baseline solutions, including the Two-layer Iterative Content Caching and Distribution algorithm (TICCD) [5], the Density-based Content Distribution approach (DCD) [17], and the submodular optimization-based cooperative caching strategy (Submodular) [6], as benchmarks for performance comparison. All three benchmark algorithms—TICCD, DCD, and Submodular—are designed to minimize cache content delivery delays. TICCD employs dynamic programming techniques for both content cache placement and distribution. DCD utilizes an iterative algorithm based on Gibbs sampling for cache placement and applies the Kuhn-Munkres algorithm to solve the content distribution problem. Similarly, the Submodular method adopts a greedy algorithm based on submodular optimization to address dynamic content caching and distribution. While these approaches consider dynamic network topologies and resource constraints, they overlook several critical aspects that are central to our proposed BPSO_DAI algorithm. First, none of these benchmark methods consider service continuity across consecutive time slots. This omission may result in service disruptions and negatively impact user experience and overall system performance. Second, advertisement insertion as a monetization strategy is not addressed in any of the benchmark models, despite its growing relevance in content delivery networks. Third, the impact of user disengagement is completely neglected, even though it can significantly influence system efficiency and monetization. These missing components reduce the practical effectiveness and robustness of the benchmark algorithms compared to our proposed approach.

C. Key Performance Indicators (KPIs)

- ***Cache Hit Rate:*** This metric refers to the average ratio of content requests served by cache nodes, instead of the cloud-based content provider, to the total number of content requests in each time slot. The average of the cache hit rate at t is computed as:

$$CHR(t) = \frac{1}{|\mathbb{Q}(t)|} \sum_{i=1}^{|\mathbb{Q}(t)|} \sum_{f=1}^{|\mathbb{F}|} \sum_{n=1, n\neq c_p}^{|\mathbb{N}|} \sum_{v=1}^{|\mathbb{V}|} \zeta_{v,i}^{f}(t)\, y_{v,n}^{f}(t). \tag{26}$$

- ***Content Provider Revenue:*** This metric represents the total revenue generated by the content provider, calculated based on the number of successfully served requests and

the associated revenue from ad insertions or content delivery. The average revenue of the content provider at t is calculated as follows:

$$R(t) = \frac{1}{|\mathbb{Q}(t)|} \sum_{i=1}^{|\mathbb{Q}(t)|} \left(\Pi_i^s(t) + \Pi_i^a(t) - \Pi_i^d(t) \right). \quad (27)$$

• ***Content Delivery Delay:*** This metric refers to the average delay experienced by end users when retrieving content from cache nodes. The average content delivery delay per service request is computed as follows:

$$CDD(t) = \frac{1}{|\mathbb{Q}(t)|} \sum_{i=1}^{|\mathbb{Q}(t)|} \sum_{f=1}^{|\mathbb{F}|} \sum_{n=1}^{|\mathbb{N}|} \sum_{e_k^h \in \mathscr{E}_v^n(t)} D_k^h(t) \zeta_{v,i}^f(t) y_{v,n}^f(t). \quad (28)$$

• ***Service Disruption Ratio:*** This metric measures the incompleteness of user requests (QoE) due to two factors – (1) when a request continues into the next time slot, and the cached content is evicted due to a cache update and (2) when a user terminates a service because ad duration or frequency exceeds their tolerance threshold, leading to disengagement. Service disruption occurs when the probability of user disengagement exceeds 1. It is quantified as the ratio of disrupted services—due to either cache eviction or user disengagement—to the total number of service requests in a given time slot, including active requests:

$$SDR(t) = \frac{1}{|\mathbb{Q}(t)|} \sum_{i=1}^{|\mathbb{Q}(t)|} \sum_{n=1}^{|\mathbb{N}|} |x_n^f(t) - x_n^f(t-1)| \Upsilon^i(t-1) \zeta_{v,i}^f(t) + \eta^i(t). \quad (29)$$

• ***Content delivery cost:*** This metric refers to the revenue penalty for distributing content to the end user. The average content delivery cost per request at time slot t is calculated using Equation (15) as:

$$C(t) = \frac{1}{|\mathbb{Q}(t)|} \sum_{q_i \in \mathbb{Q}(t)} \Pi_i^{cos}(t)), \quad (30)$$

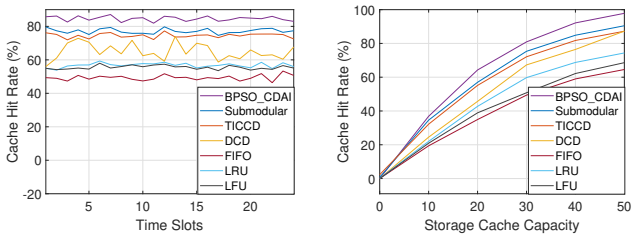
where $\Pi_i^{cos}(t)$ denotes the content delivery cost associated with request q_i at time slot t.

D. Results and Discussion

1) Cache Hit Rate

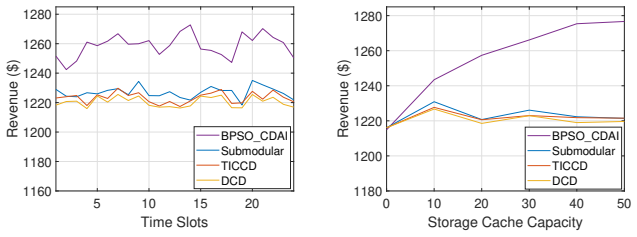
As illustrated in Figure 3a, the proposed solution, BPSO_DAI, consistently outperforms the benchmark algorithms in terms of cache hit rate, achieving average improvements of 9.7%, 12.8%, 17%, 30.1%, 32.4%, and 36.7% over Submodular, TICCD, DCD, LRU, LFU, and FIFO, respectively, across all time slots. Figure 3b shows that the cache hit rate increases with cache size across all algorithms. A larger cache allows storing more content items, thereby improving the cache hit rate. For instance, when the cache size per node is 40GB, the proposed solution outperforms the benchmark algorithms—Submodular, TICCD, DCD, LRU, LFU, and FIFO—by more than 6.7%, 10.4%, 16.5%, 24.8%, 31.3%, and 34.5%, respectively. As shown in Figure 3, the proposed solution, *BPSO_DAI*, consistently achieves a higher cache hit rate across all time slots and cache size configurations compared to benchmark algorithms. This superior performance stems from two primary factors. First, *BPSO_DAI* employs a proximity-based content caching strategy, which enables it to leverage a larger number of cache nodes throughout the network. This contrasts with the benchmarks, where content placement strategies are less efficient in utilizing available storage resources. Second, unlike the benchmarks, *BPSO_DAI* incorporates service continuity by accounting for content requests that span multiple time slots. Benchmark algorithms fail to consider such continuity, resulting in cache misses for recurring requests and thus lower hit rates. In contrast, *BPSO_DAI* effectively handles these requests, significantly boosting the cache hit rate and overall system efficiency.

Furthermore, the main reason the proposed solution outperforms the commonly used cache replacement algorithms—*LRU*, *LFU*, and *FIFO*—is that *BPSO_DAI* employs a content popularity-based caching scheme. Unlike these traditional methods, which rely solely on content access timing, *BPSO_DAI* prioritizes storing highly popular content, leading to a higher cache hit rate. In contrast, *LRU*, *LFU*, and *FIFO* do not directly consider content popularity but instead rely on content access timing. This approach often evicts popular content too soon, even if it remains in high demand, leading to a lower cache hit rate. For example, in *LRU*, a frequently accessed yet highly popular content item may be replaced by a newly arrived but less popular item simply because it was not accessed recently. Similarly, *LFU* may evict content with a temporary drop in request frequency, even if it is expected to be popular again. On the other hand, *FIFO* replaces content strictly based on arrival time, disregarding popularity altogether, which can lead to unnecessary cache misses.



(a) With varying time slots. (b) With varying cache capacity.

Figure 3: Average cache hit rate.



(a) With varying time slots. (b) With varying cache capacity.

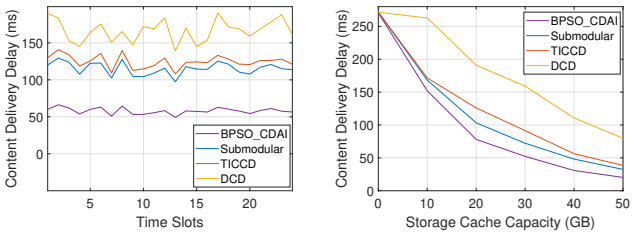
Figure 4: Average end user cost.

2) Content Provider Revenue

Figure 4 compares the average revenue generated by the content provider—through content delivery and monetized advertising—between the proposed solution and benchmark algorithms: Submodular, TICCD, and DCD. Figure 4a shows that *BPSO_DAI* consistently outperforms Submodular, TICCD, and DCD in terms of revenue, achieving

average gains of more than $40, $50, and $52, respectively, across all time intervals. As seen in the figure, the revenue fluctuates across time slots. This fluctuation in *BPSO_DAI* is primarily due to two factors: (1) variations in network topology, which impact content delivery delay and force the optimization process to balance revenue and delay trade-offs, and (2) statistical variations in user requests, which naturally lead to revenue changes. In contrast, the fluctuations observed in the benchmarks are attributed solely to statistical variations in content requests. As shown in Figure 4b, revenue in *BPSO_DAI* increases linearly with cache size capacity. This is because larger on-board caches store more content items, reducing delivery delay. Consequently, the optimization process has greater flexibility in inserting more ads, leading to higher revenue. In contrast, revenue in Submodular, TICCD, and DCD remains unaffected by variations in cache size, as it relies primarily on subscription fees, which remain constant regardless of whether requests are served from the cloud or on-board caches. However, at very low cache capacities, *BPSO_DAI* and the benchmarks exhibit similar performance since revenue is primarily dependent on subscription pricing. As the cache capacity increases, the proposed solution outperforms the benchmark. For example, as shown in Figure 4b, *BPSO_DAI* achieves an average revenue gain of more than $53, $55, and $56 compared to Submodular, TICCD, and DCD, respectively, when the cache capacity per node is 40 GB.

The improvement in our proposed solution is driven by three key factors. First, it integrates advertising monetization alongside subscription-based pricing, maximizing revenue for the content provider. Second, it employs a distributed, collaborative content cache distribution scheme, allowing unassociated service requests to be handled online through cooperation with neighboring cache nodes. Finally, its topology-aware design ensures seamless handling of content services that span multiple time slots, enabling more efficient accommodation of content requests.

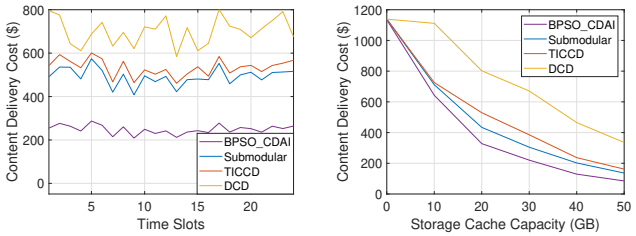


(a) With varying time slots. (b) With varying cache capacity.

Figure 5: Average content delivery delay.

3) Content Delivery Delay

Figure 5 presents the average content cache delivery delay, comparing the proposed solution with Submodular, TICCD, and DCD. As shown in Figure 5a, *BPSO_DAI* reduces the average content delivery delay by 60.4ms, 68.2ms, and 125.5ms, respectively, compared to Submodular, TICCD, and DCD across all time slots. Additionally, Figure 5b illustrates that the content delivery delay decreases as cache storage size increases. This is because more extensive cache storage enables more content items to be cached closer to users, reducing fetching time and improving content delivery performance. However, the proposed and benchmark solutions exhibit similar delivery delays at very low cache storage capacities since they rely primarily on remote cloud-based content providers. However, as the size of the cache increases, the proposed solution consistently outperforms the benchmark by maintaining lower content delivery delays across various cache capacities. For example, as shown in Figure 5b, *BPSO_DAI* achieves significantly lower content delivery delays than Submodular, TICCD, and DCD algorithms, reducing the delay by 17.4ms, 25.5ms, and 70ms, respectively, at a cache size of 50GB. As shown in Figure 5, the proposed solution consistently achieves reduced content delivery delay across all time slots and cache sizes. This performance improvement in content delivery delay is closely linked to the higher cache hit rate discussed earlier. The gains stem from the proposed solution's ability to utilize more caching resources through a content popularity-driven strategy, as well as its awareness of service continuity and network topology—factors not considered by the benchmark methods.



(a) With varying time slots. (b) With varying cache capacity.

Figure 6: Average content delivery cost.

4) Content delivery cost

Figure 6 illustrates the content delivery cost as a function of content delivery delay, which significantly affects the Quality of Experience (QoE) during the content distribution process. As shown in Figure 6a, the proposed solution achieves a lower average delivery cost per user request compared to the benchmark approaches—Submodular, TICCD, and DCD —by $224.2, $275.3, and $421.5, respectively, across all time slots. Similarly, Figure 6b compares the content delivery cost across different cache sizes. For instance, at a cache size of 40 GB per node, the proposed $BPSO_DAI$ reduces delivery costs by $74, $107.2, and $336 compared to Submodular, TICCD, and DCD, respectively. As evident from the figure, the delivery cost decreases with increasing cache size. This is because larger cache capacities allow more content to be stored locally, reducing delivery delays and thereby lowering overall delivery costs. The performance gains stem from the proposed algorithm's ability to efficiently utilize storage resources and ensure service continuity across time slots. By adapting to dynamic topology changes and supporting seamless content delivery, the proposed method enables caching and delivery of more

content items with reduced latency and cost. This metric is a critical QoE indicator, as it reflects how much a user pays in relation to the speed of content delivery—directly impacting user satisfaction.

5) Service Disruption Ratio

Figure 7 compares the service disruption ratio of the proposed $BPSO_DAI$ solution with the benchmark approaches—Submodular, TICCD, and DCD—over the first 10 time slots, assuming 50% of content requests continue into the next time slot. The proposed solution demonstrates a significantly lower disruption rate, with fewer than 2% of content requests affected. In contrast, the Submodular, TICCD, and DCD approaches exhibit disruption rates exceeding 12.6%, 11.3%, and 10.2%, respectively. This improvement is primarily attributed to the topology-aware design of $BPSO_DAI$, which maintains service continuity across dynamic, time-varying network topologies. Nevertheless, a small fraction of service disruptions still occur due to resource constraints and user disengagement caused by ad insertions. Service disruption is a critical metric, as it directly affects user satisfaction and, consequently, the overall Quality of Experience (QoE).

Figure 8 shows that the *BPSO_DAI* converges in under 200 iterations.

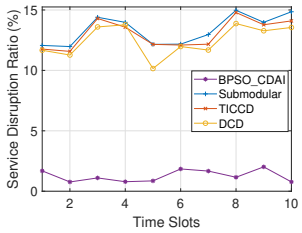


Figure 7: Service disruption ratio.

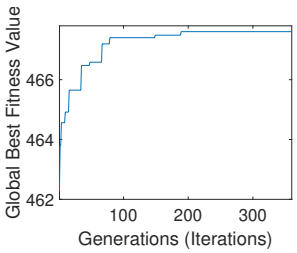
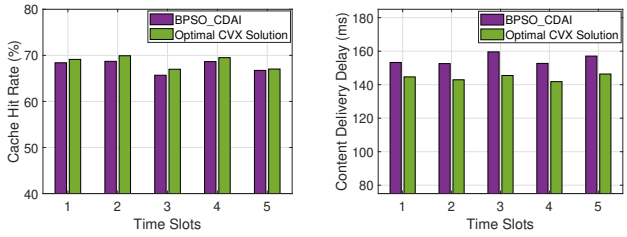


Figure 8: The convergence process of *BPSO_DAI*

.



(a) Cache hit rate. (b) Content delivery delay.

Figure 9: Performance comparison with *ILP*.

6) Comparison of BPSO_DAI with the Optimal ILP

This subsection evaluates the optimality gap of the proposed *BPSO_DAI* and the optimal *ILP* solution. The *ILP* model formulated in Equations (16) and (17) was solved using the *CVX MOSEK* optimization solver [52] with 60 content items and 30 ad content items, and a 5GB cache storage capacity. We set the Zipf exponent $\delta = 0.1$ to introduce diversity and consider only the first five time slots. Figure 9 illustrates the penalty gap between the proposed and the optimal *ILP* solutions. As shown in Figure 9a, *BPSO_DAI* achieves a maximum of 1.3% optimality gap in cache hit rate, demonstrating performance close to *ILP*. Similarly, Figure 9b shows a 14.32 ms optimality gap for content distribution, with a delivery delay gap of less than one hop between satellites and between satellite and VSATs. This small penalty gap arises from the enhancements introduced by the collaborative user-cache association for unassociated requests, which significantly improves the performance of the proposed *BPSO_DAI* algorithm.

VI. Conclusion

In conclusion, this paper presents a BPSO-based approach for efficient collaborative content distribution with DAI in multi-layer STINs. It focuses on topology-aware content caching, revenue-driven cache distribution, and the impact of service continuity and ad insertion on user engagement in time-varying networks. The proposed solution, *BPSO_DAI*, selects optimal cache nodes for storing popular content, handling requests, and performing ad stitching while considering node storage and processing capacities, link constraints, and network connectivity. It ensures seamless service continuity between time slots, preventing disruptions during handovers. Additionally, we introduce a collaborative distributed online strategy for efficient content cache distribution with DAI, specifically addressing unassociated requests in real-time. Simulation results show significant improvements in cache hit rates, reduced cache fetching durations, and increased content provider revenues. The proposed solution achieves more than a 4.5% increase in revenue and reduces cache retrieval duration by over 39% compared to the benchmark algorithms—Submodular, TICCD, and DCD. Future work will focus on integrating machine learning-based techniques to predict content popularity and user behavior, thereby further improving caching efficiency, enabling personalized ad insertion, and enhancing user engagement in Satellite-Terrestrial Integrated Networks (STINs). While federated learning offers privacy advantages, its communication overhead and deep reinforcement learning's computational demands make them less suitable for resource-constrained or latency-sensitive applications such as MEC-enabled STINs. We aim to investigate lightweight and efficient alternatives in our future research. Future work will also explore sensitivity analysis under MEC node failures and extend robustness evaluations to encompass more diverse mobility patterns and failure scenarios.

References

[1] Ericsson, "Ericsson Mobility Report June 2024," Accessed: Jan. 2025. [Online]. Available: https://www.ericsson.com/en/reports-and-papers/mobility-report/reports/june-2024

[2] X. Zhu and C. Jiang, "Integrated Satellite-Terrestrial Networks Toward 6G: Architectures, Applications, and Challenges," *IEEE Internet Things J*, vol. 9, no. 1, pp. 437–461, 2022.

[3] R. Speed, "Axiom Space and Red Hat to take edge computing into orbit," *The Register*, March 2025. [Online]. Available: https://www.theregister.com/2025/03/07/axiom_space_and_red_hat/

[4] Wei, Qing and Chen, Yingyang and Jia, Ziye and Bai, Wenle and Pei, Tingrui and Wu, Qihui, "Energy-Efficient Caching and User Selection for Resource-Limited SAGINs in Emergency Communications," *IEEE Trans. Commun.*, vol. 73, no. 6, pp. 4121–4136, 2025.

[5] Y. Chen, X. Ma, A. Zhou, and S. Wang, "Cooperative Content Caching and Distribution for Satellite CDNs," in *IEEE ICNP*, 2023, pp. 1–6.

[6] X. Zhu, C. Jiang, Z. Yang, and H. Wang, "Delay-Optimized Edge Caching in Integrated Satellite-Terrestrial Networks With Diverse Content Popularity Distribution and User Access Modes," *IEEE Internet Things J*, pp. 1–1, 2024.

[7] G. Megan and E. Jennifer, "How does Google make money?" 2021, Accessed: Nov. 2024. [Online]. Available: https://www.cnbc.com

[8] R. Dubin, A. Dvir, O. Hadar, T. Frid, and A. Vesker, "Novel ad insertion technique for MPEG-DASH," in *IEEE CCNC*, 2015, pp. 582–587.

[9] L. Bringuier, "Increasing ad personalization with server-side ad insertion," in *IBC Conference*. IET, 2016, p. 44.

[10] "Conviva's State of Streaming Advertising 2021," 2021. [Online]. Available: https://pages.conviva.com/rs/138-XJA-134/images/RPT_Conviva_State_of_Streaming_Advertising_2021.pdf

[11] GB Times, "Why is YouTube so sensitive?" *GB Times*, 2024. [Online]. Available: https://gbtimes.com/why-is-youtube-so-sensitive/

[12] J. Coppola, "The Psychology Behind Why People Dislike Ads," 2020. [Online]. Available: https://wistia.com/learn/marketing/the-psychology-behind-why-people-dislike-ads

[13] Parrot Analytics, "Investigating the reasons why americans stop watching tv shows," *Parrot Analytics*, 2018. [Online]. Available: https://www.parrotanalytics.com/insights/investigating-the-reasons-why-americans-stop-watching-tv-shows/

[14] H. G. Abreha, I. Maity, H. Chougrani, C. Politis, and S. Chatzinotas, "Resource-aware on-board content caching in multi-layer satellite edge networks," in *IEEE ICC*, 2024, pp. 3943–3949.

[15] S. Bhandari, T. X. Vu, S. Chatzinotas, and B. Ottersten, "Efficient content caching for delivery time minimization in the leo satellite networks," in *IEEE ICC Workshops*, 2023, pp. 1246–1252.

[16] J. Tang, J. Li, L. Zhang, K. Xue, Q. Sun, and J. Lu, "Content-Aware Routing based on Cached Content Prediction in Satellite Networks," in *IEEE GLOBECOM*, 2022, pp. 6541–6546.

[17] D. Jiang, F. Wang, Z. Lv, S. Mumtaz, S. Al-Rubaye, A. Tsourdos, and O. Dobre, "QoE-Aware Efficient Content Distribution Scheme For Satellite-Terrestrial Networks," *IEEE Trans. Mobile Comput.*, vol. 22, no. 1, pp. 443–458, 2023.

[18] G. Shushi, C. Zihan, W. Yaonan, Z. Qinyu, and W. Ye, "Layered coded cache placement and cooperative delivery with sharing links in satellite-terrestrial integrated networks," *China Communications*, vol. 21, no. 3, pp. 217–229, 2024.

[19] R. Mekuria, Y. Syed, and G. Hughes, "Robust SCTE 35 in the OTT workflow," in *Proceedings of the 2nd Mile-High Video Conference*, 2023, pp. 27–33.

[20] S. Pham, K. Hughes, and T. Lohmar, "Implementing dynamic ad insertion in HTML5 using MPEG DASH," in *IBC Technical Papers*. Fraunhofer FOKUS, Microsoft, Ericsson, 2017.

[21] R. Seeliger and L. Bassbouss, "Dynamic ad substitution in hybrid broadband broadcast environments," in *IEEE LCN*, Edmonton, AB, Canada, 2022, pp. 248–250.

[22] Jia, Ziye and Cao, Yilu and He, Lijun and Wu, Qihui and Zhu, Qiuming and Niyato, Dusit and Han, Zhu, "Service Function Chain Dynamic Scheduling in Space-Air-Ground Integrated Networks," *IEEE Trans. Veh. Technol.*, vol. 74, no. 7, pp. 11 235–11 248, 2025.

[23] He, Lijun and Li, Shiyin and Jia, Ziye and Wang, Juncheng and Han, Zhu, "Joint Data Compression and Task Scheduling for LEO Satellite Networks," *IEEE Trans. Veh. Technol.*, vol. , no. , pp. 1–6, 2025.

[24] Abreha, Haftay Gebreslasie and Chougrani, Houcine and Maity, Ilora and Nguyen, Van-Dinh and Chatzinotas, Symeon and Politis, Christos, "Fairness-Aware Dynamic VNF Mapping and Scheduling in SDN/NFV-Enabled Satellite Edge Networks," in *IEEE ICC*, vol. , no. , 2023, pp. 4892–4898.

[25] H. Cui, J. Zhang, Y. Geng, Z. Xiao, T. Sun, N. Zhang, J. Liu, Q. Wu, and X. Cao, "Space-air-ground integrated network (sagin) for 6g: Requirements, architecture and challenges," *China Communications*, vol. 19, no. 2, pp. 90–108, 2022.

[26] T. Darwish, T. A. Alhaj, and F. A. Elhaj, "Controller Placement in Software Defined Emerging Networks: A Review and Future Directions," *Telecommunication Systems*, vol. 88, p. 18, 2025.

[27] S. Rawat, A. Chopra, S. Singh, and S. Sinha, "Mid Roll Advertisement Placement Using Multi Modal Emotion Analysis," in *International Conference on Artificial Neural Networks*. Springer, 2019, pp. 159–171.

[28] A. Patil, S. Prabhu, and P. Ashok, "5G Monetization Strategies and Business Models: An insightful Analysis for Telecommunications Service Providers," in *ICDICI*, 2024, pp. 16–23.

[29] K. Fridgeirsdottir and S. Najafi-Asadolahi, "Cost-per-impression pricing for display advertising," *Operations Research*, vol. 66, no. 3, pp. 653–672, 2018.

[30] H. Abedi Firouzjaei, "Survival analysis for user disengagement prediction: question-and-answering communities' case," *Social Network Analysis and Mining*, vol. 12, no. 1, p. 86, 2022. [Online]. Available: https://link.springer.com/article/10.1007/s13278-022-00914-8

[31] L. Tian, "Survival Analysis: Unit 3 Lecture Notes," Accessed 2025, stanford University Lecture Notes. [Online]. Available: https://web.stanford.edu/~lutian/coursepdf/STAT331unit3.pdf

[32] Q. Liang, Y. Liu, and W. Tang, "Joint Cache Placement and Content Scheduling in Integrated LEO Satellite-Terrestrial Networks," in *IEEE ICCC*, 2022, pp. 642–648.

[33] A. Chakraborty and A. K. Kar, *"Swarm Intelligence: A Review of Algorithms"*. Springer International Publishing, 2017, pp. 475–494.

[34] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of IJCNN*, vol. 4, 1995, pp. 1942–1948 vol.4.

[35] Zu, Jiachen and Hu, Guyu and Wu, Yang and Shao, Dongsheng and Yan, Jiajie, "Resource Aware Chaining and Adaptive Capacity Scaling for Service Function Chains in Distributed Cloud Network," *IEEE Access*, vol. 7, pp. 157 707–157 723, 2019.

[36] T. Kelso. Celestrak - NORAD Two-Line Element Sets. Accessed: Nov. 2024. [Online]. Available: https://celestrak.org/NORAD/elements/

[37] SES S.A., "Ses teleport map - coverage overview," 2024, Accessed: Nov. 2024. [Online]. Available: https://www.ses.com/our-coverage/teleport-map

[38] Iridium Communications Inc., "Iridium communications inc. - official website," 2024, Accessed: Nov. 2024. [Online]. Available: https://www.iridium.com/company/

[39] A. Abatko. (2018) US Zip Code Geolocations from 2018 Government Data. [Online]. Available: Available:https://gist.github.com/abatko

[40] X. Jin and J. Han, *K-Means Clustering*. Boston, MA: Springer US, 2010, pp. 563–564. [Online]. Available: https://doi.org/10.1007/978-0-387-30164-8_425

[41] Zhang, Peiying and Wang, Chao and Kumar, Neeraj and Liu, Lei, "Space-Air-Ground Integrated Multi-Domain Network Resource Orchestration Based on Virtual Network Architecture: A DRL Method," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 3, pp. 2798–2808, 2022.

[42] H. G. Abreha, H. Chougrani, I. Maity, Y. Drif, C. Politis, and S. Chatzinotas, "Fairness-Aware VNF Mapping and Scheduling in Satellite Edge Networks for Mission-Critical Applications," *IEEE Trans. Netw. Service Manag*, vol. 21, no. 6, pp. 6716–6730, 2024.

[43] Overcast HQ, "How Big Are Video Files?" https://www.overcasthq.com/blog/how-big-are-video-files/, 2023, Accessed: 2025-08-04.

[44] Google Ads Support, "About ad insertion," 2025, Accessed: 2025-08-04. [Online]. Available: {https://support.google.com/google-ads/answer/11462260?hl=en}

[45] H. G. Abreha, I. Maity, H. Chougrani, C. Politis, and S. Chatzinotas, "On-Board Content Caching With Dynamic Cache Reconfiguration in Multi-Layer Satellite Edge Networks," *IEEE IEEE Open J. Commun. Soc.*, vol. 6, pp. 5727–5745, 2025.

[46] OmniAccess. (2023) OmniAccess LEO. Accessed: Nov. 2024. [Online]. Available: Available:https://www.omniaccess.com/leo/

[47] X. Gao, R. Liu, A. Kaushik, and H. Zhang, "Dynamic Resource Allocation for Virtual Network Function Placement in Satellite Edge Clouds," *IEEE Trans. Netw. Service Manag*, vol. 9, no. 4, pp. 2252–2265, 2022.

[48] Gee-Tech, "What Broadband Speed Do You Need to Stream in SD, HD, and 4K?" 2025. [Online]. Available: https://gee-tech.com/web/what-broadband-speed-do-you-need-to-stream-in-sd-hd-and-4k/

[49] G. Ads, "Actual cost-per-click (cpc): Definition," 2025. [Online]. Available: https://support.google.com/google-ads/answer/6297?hl=en&ref_topic=24937

[50] G. Cloud, "Cloud storage pricing," 2025, Accessed: Dec. 2025. [Online]. Available: https://cloud.google.com/storage/pricing

[51] R. Zhao, Y. Ran, J. Luo, and S. Chen, "Towards Coverage-Aware Cooperative Video Caching in LEO Satellite Networks," in *IEEE GLOBECOM*, 2022, pp. 1893–1898.

[52] I. CVX Research, "Mosek solver," https://cvxr.com/cvx/doc/mosek.html, Accessed: Jan. 2025.

This article has been accepted for publication in IEEE Transactions on Network and Service Management. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2025.3629810

Haftay Gebreslasie Abreha (Student Member, IEEE) received his M.Sc. degrees in Telecommunications Engineering and Communication Network Engineering jointly from the University of Trento and Scuola Superiore Sant'Anna, Italy, in 2015 and PhD in Informatics from the University of Luxembourg in 2025. He is currently a Research Associate at the Interdisciplinary Centre for Security, Reliability and Trust (SnT) at the University of Luxembourg. His research focuses on 6G network security and edge computing architectures for integrated satellite–terrestrial networks, with a particular emphasis on QKD integration into 6G networks by leveraging software-defined networking (SDN) and network function virtualization (NFV) for efficient service provisioning. Before joining SnT, Haftay was a research assistant at renowned institutions, including IMDEA Networks Institute in Spain, the University of Trento in Italy, and the United Arab Emirates University (UAEU). He also has industrial experience as a network administrator with the Immigration and Citizenship Service of Ethiopia.

Ilora Maity (Senior Member, IEEE) received the B.Tech. degree in Computer Science and Engineering from the West Bengal University of Technology, India, in 2008, the M.E. degree from the Indian Institute of Engineering Science and Technology, Shibpur, India, in 2011, and the Ph.D. degree from the Indian Institute of Technology Kharagpur, India, in 2021. She worked as a Technical Analyst in Banking and Financial Services at Cognizant Technology Solutions India Pvt. Ltd., Kolkata, from 2011 to 2015, and was a Postdoctoral Researcher at Aalto University, Finland, in 2021. She is currently a Research Associate with the Interdisciplinary Centre for Security, Reliability and Trust (SnT), University of Luxembourg. Dr. Maity has authored over 40 papers in refereed international journals and conferences. Her research interests include software-defined networking, network function virtualization, the Internet of Things, satellite communications, and quantum communications. She has served as a member of the Technical Program Committee for several IEEE conferences and is a member of the editorial board of the International Journal of Wireless Communications and Mobile Computing.

Youssouf DRIF received his PhD degree from the Federal University of Toulouse Midi-Pyrénées (France), in 2022. His research interests are SDN and NFV for Satcom and, more precisely, in using these technologies for satellite integration into mobile networks. Youssouf joined the Signal Processing & Satellite Communications research group, SIGCOM, headed by Prof. Symeon Chatzinotas, and has since worked on 5G-NTN, UAVs, and Quantum technologies in multiple ESA and European projects.

Christos Politis (Fellow, IEEE) is Senior Communications Systems Engineer within SES TechCom. Prior to joining SES in January 2018, he was PhD Researcher at Interdisciplinary Centre for Security, Reliability and Trust (SnT) within University of Luxembourg in 2014-2017, and Satellite Communications Engineer at Centre National d' Etudes Spatiales (CNES), France in 2012. He has technical experience in national and international R&D and Innovation projects on ICT with major focus on Satellite Communications and he has authored several journals, conference and book chapter publications. In particular, he has been contributing to the 5G related projects H2020 Work Program 2018-2020 "iNGENIOUS", H2020 5G PPP Phase III "5G-VINNI", H2020 5G PPP Phase II "SaT5G", ESA ARTES "SATis5" and ESA ARTES "EdgeSAT", as well as to the phase 1 of ESA funded "5G-EMERGE". He obtained his PhD degree in Spectrum Monitoring Algorithms for Wireless and Satellite Communications from the University of Luxembourg, Luxembourg in 2018, his MSc. degree in Communications and Signal Processing from the University of Newcastle, UK in 2014, his MSc. degree in Space Communication Systems from ISAE-SUPAERO, France in 2012, and the Dipl.-Eng. degree in Electrical and Computer Engineering from the University of Patras, Greece in 2011.

Symeon Chatzinotas (S'06, M'09, SM'13, F'23) is Full Professor and Head of the SIGCOM Research Group at SnT, University of Luxembourg. He is coordinating the research activities on communications and networking across a group of 80 researchers, acting as a PI for more than 40 projects and main representative for 3GPP, ETSI, DVB. He is currently serving in the editorial board of the IEEE Transactions on Communications, IEEE Open Journal of Vehicular Technology and the International Journal of Satellite Communications and Networking. In the past, he has been a Visiting Professor at the University of Parma, Italy and was involved in numerous RD projects for NCSR Demokritos, CERTH Hellas and CCSR, University of Surrey. He was the co-recipient of the 2014 IEEE Distinguished Contributions to Satellite Communications Award and Best Paper Awards at WCNC, 5GWF, EURASIP JWCN, CROWNCOM, ICSSC. He has (co-)authored more than 700 technical papers in refereed international journals, conferences and scientific books.