PhD-FSTM-2025-070

The Faculty of Science, Technology and Medicine

DISSERTATION

Defence held on 24/06/2025 in Luxembourg

to obtain the degree of

DOCTEUR DE LUNIVERSITE DU LUXEMBOURG
EN INFORMATIQUE

by

Haftay Gebreslasie Abreha
Born on 05 June 1987 in Tigray, Ethiopia

OPTIMIZED SERVICE PROVISIONING FOR NEXT-GENERATION APPLICATIONS WITH SATELLITE EDGE COMPUTING

Dissertation defense committee

Dr. Symeon Chatzinotas, Dissertation Supervisor
Professor, Universit du Luxembourg

Dr. Bhavani Shankar Mysore Rama Rao, Chairman
Professor, Universit du Luxembourg

Dr. Fabrizio Granelli, Member
Professor, Universit di Trento, Italy

Dr. Thang X. Vu, Member
Research scientist, Universit du Luxembourg

Dr. Christos Politis, Member
Senior Engineer, SES S.A., Luxembourg

Optimized Service Provisioning for Next-Generation Applications with Satellite Edge Computing

Haftay Gebreslasie Abreha

Abstract

Satellite Edge Computing (SEC) extends the Mobile Edge Computing (MEC) paradigm by deploying cloud computing resource pools on in-orbit satellite nodes within a constellation. By enabling computational offloading to space, SEC empowers the on-board execution and seamless completion of service. At the same time, its in-orbit storage capacity facilitates proactive content caching and efficient content delivery to end users. The distributed and localized processing and storage capabilities of satellites in the SEC paradigm, unlike the traditional bent-pipe satellite architecture, ensure high service availability and coverage, particularly for users in remote, underserved, and disaster-stricken areas where terrestrial networks are unavailable. The ubiquitous service coverage, low latency, and high bandwidth enhance scalability and meet the stringent requirements of next-generation applications such as mission-critical services, real-time analytics, and high-quality multimedia delivery, making the SEC paradigm a promising solution for service provisioning in future networks. In this context, service provisioning refers to allocating and managing resources to deliver services to end users based on application-specific requirements.

Although SEC-enabled platforms provide tremendous advantages, service provisioning in these networks faces several significant challenges. Firstly, on-board satellite resources are limited, so they cannot process or store all the tasks and data that users request. This necessitates advanced optimization algorithms to manage and allocate these scarce resources efficiently. Furthermore, the requirements of emerging next-generation applications and services are stringent (*e.g.*, low latency, high reliability, and high bandwidth) and often complex to achieve, particularly in the satellite environment where resource constraints and long propagation delays are prevalent. The dynamic nature of service requests, with their heterogeneous and sometimes intricate requirements, adds another layer of difficulty in meeting these demands. Additionally, the mobility of satellites results in dynamic coverage areas, time-varying topologies, and fluctuations in link characteristics. These factors hinder seamless service provisioning and complicate the design of effective service provisioning strategies. An advanced service provisioning scheme is required to alleviate these challenges and fully leverage the benefits of the SEC. This scheme must comprehensively accommodate heterogeneous requirements, dynamic traffic requests, and evolving topologies while meeting stringent service requirements. Such a scheme is crucial in next-generation networks like 5G/6G, where Non-Terrestrial Networks (NTN) are considered a key enabling technology. NTNs improve network resilience by integrating terrestrial and satellite networks, providing ubiquitous connectivity, and improving overall network performance.

Motivated by these challenges, in this thesis, we explore service provisioning schemes for next-generation applications by leveraging the in-orbit computational and storage power of SEC. In the first contribution, we proposed a Virtual Network Function (VNF) mapping and scheduling scheme for mission-critical applications, which enables Network Functions (NFs) execution on-board in the constellation to maximize fairness in terms of End-to-End (E2E) service delay margin and reduce the E2E service delay among competing services. In the second contribution, we propose an efficient storage resource utilization method for on-board content caching by designing a novel satellite proximity-based content popularity scheme. Furthermore, we proposed an optimization framework for cache reconfiguration overhead-aware on-board content caching. We also study on-board cache-to-cache updates to reduce reconfiguration overhead, followed by an Age of Information (AoI)-aware content caching scheme to ensure cached content freshness. Additionally, in the third contribution, we focus on service provider revenue generation through ad monetization while enhancing efficient content distribution schemes. We study the impact of excessive ad insertion, which leads to user disengagement and indirectly reduces the content provider's revenue. We model and propose a joint optimization algorithm that balances revenue and end-user delivery delay while considering the constraints. We also propose seamless content delivery by allowing distributed Server-Side Ad Insertion (SSAI) and service continuity-aware content distribution strategies to mitigate user disengagement and network fluctuations.

Preface

This Ph.D. thesis was conducted from December 2021 to May 2025, primarily within the SIGCOM research group at the Interdisciplinary Centre for Security, Reliability, and Trust (SnT), University of Luxembourg, under the supervision of Prof. Symeon Chatzinotas, Chief Scientist I and Head of the SIGCOM. The thesis was co-supervised by Dr. Christos Politis, Communications Systems Engineer at Socit Europenne des Satellites (SES), Luxembourg; Dr. Ilora Maity, Research Associate at SIGCOM; and Dr. Houcine Chougrani, Research Scientist at King Abdullah University of Science and Technology (KAUST), Saudi Arabia. The Comit d'valuation de Thse (CET) members, comprising the supervisors, co-supervisors, and Prof. Fabrizio Granelli, Full Professor at the University of Trento, Italy, periodically evaluated the progress and quality of the thesis throughout its development.

Support of the Thesis

This work was supported by the Luxembourg National Research Fund (FNR) in collaboration with the industrial partner SES S.A., under the project titled "Satellite-assisted edge/fog processing for latency reduction and enhanced QoS in mission-critical IoT applications," grant IPBG19/14016225/INSTRUCT. The substantial support provided by the SIGCOM research group is also gratefully acknowledged.

Acknowledgments

First and foremost, I express my deepest gratitude to all my supervisors and colleagues, whose integrity and dedication have enriched my knowledge and inspired me throughout this journey. I am especially thankful to Dr. Ilora Maity for her invaluable guidance, steadfast support, and the numerous opportunities she provided for my professional and personal growth. I am also grateful to Dr. Houcine Chougrani from KAUST for his invaluable technical guidance during the early stages of my research. Furthermore, I am deeply thankful to Prof. Symeon Chatzinotas for his outstanding technical guidance and timely feedback throughout my PhD journey. In addition, I extend my gratitude to Dr. Christos Politis from SES for his insightful technical guidance and for helping me grasp key practical concepts of satellite communication systems. I also thank Prof. Fabrizio Granelli from the University of Trento for his valuable comments and suggestions during each CET meeting, which significantly improved my technical skills and produced high-quality research papers. I also thank the Luxembourg National Research Fund (FNR) and SES S.A. for funding my work. Last but not least, I am deeply grateful to my family and friends for their unwavering support throughout this journey.

Haftay Gebreslaise Abreha

Luxembourg, July 2, 2025

Contents

List of Figures

List of Tables

List of Symbols

Symbols	Description
$\mathbb{N}$	Sets of nodes
$\mathbb{E}$	Set of links
$\mathbb{S}$	Set of satellite nodes
$\mathbb{G}$	Sets of GEO satellites
$\mathbb{M}$	Sets of MEO satellites
$\mathbb{L}$	Sets of LEO satellites
$\mathbb{U}$	Sets of user terminals
$\mathbb{J}$	Set of satellite gateways
cps	Cloud-based content provider server
aps	Cloud-based ad provider server
$\mathbb{E}$	Sets of links between nodes
t	The t^{th} time slot
G_t	Graph at snapshot (time slot) t^{th}
Δt	Each time slot duration
k^{μ}	Processing capacity of node $k \in \mathbb{N}$
k^{β}	Memory space (storage) capacity of node $k \in \mathbb{N}$
$k^{\mathcal{N}(t)}$	Set of neighboring nodes to node $k \in \approx$ at time slot t
$\mathcal{V}_k^h(t)$	Visibility indicator between nodes $h \in \mathbb{N}$ and $k \in \mathbb{N}$ at time slot t
$e_{h,k} \in \mathbb{E}$	Link between nodes $h \in \mathbb{N}$ and $k \in \mathbb{N}$
$e_{h,k}^{\Omega}(t)$	Bandwidth capacity of link $e_{h,k} \in \mathbb{E}$ at time slot t
$e_{h,k}^{D}(t)$	Propagation delay of $e_{h,k} \in \mathbb{E}$ at time slot t
$e_{h,k}^{\gamma}(t)$	Availability of $e_{h,k} \in \mathbb{E}$ at time slot t

$\mathcal{E}_{h,k}(t)$	Set of links in shortest path between $k \in \mathbb{N}$ and $h \in \mathbb{N}$ at time slot t
$e^{\rho}_{h,k}(t)$	$e_{h,k} \in \mathbb{E}$ link state comparison between current, G_t and next G_{t+1} snapshot
$\boldsymbol{S}_{sfc}(t)$	Sets of SFC service requests at time slot t
$\boldsymbol{S}_{cnt}(t)$	Sets of content service requests at time slot t
s	The s^{th} service request
s^{src}	Source node where service request $s \in \boldsymbol{S}_{cnt}(t)$ starts
s^{dst}	Destination node where service request $s \in \boldsymbol{S}_{cnt}(t)$ ends
s^{vnf}	Sets of VNF components of service request $s \in \boldsymbol{S}_{cnt}(t)$
$f_{s,j}$	The j^{th} *VNF* component of service request $s \in \boldsymbol{S}_{cnt}(t)$
$f^{c,k}_{s,j}(t)$	If node $k \in \mathbb{N}$ supports processing $f_{s,j} \in s^{vnf}$ indicator
s^{cnt}	Content item requested in $s \in \boldsymbol{S}_{cnt}(t)$
s^{B}	Minimum data rate required to deliver the content s^{cnt}
$s^{\mathcal{D}}$	Estimated display duration of content s^{cnt}
s^{ψ}	User-content subscription status of service request $s \in \boldsymbol{S}_{cnt}(t)$(1 for premium and 0 for non-premium)
$s^{t,arr}$	Arrival time of the request $s \in \boldsymbol{S}_{cnt}(t)$
$s^{\Upsilon}(t)$	Service request $s \in \boldsymbol{S}_{cnt}(t)$ continues from time slot t to the subsequent time slot $t+1$
$f^{A,k}_{s,j}(t)$	Arrival time of VNF $f_{s,j} \in s^{vnf}$ at node $k \in \mathbb{N}$ in snapshot, t
$f^{\tau,k}_{s,j}$	Queuing delay of $f_{s,j} \in s^{vnf}$ at $k \in \mathbb{N}$ in snapshot, t
$f^{\ell,k}_{s,j}$	Processing delay of $f_{s,j} \in s^{vnf}$ at node $k \in \mathbb{N}$
$d^{tot}_{s}(t)$	E2E delay of service request, $s \in \boldsymbol{S}_{sfc}(t)$
$\eta^{vnf}_{s,d}(t)$	Service Delay margin of $s \in \boldsymbol{S}_{sfc}(t)$ at snapshot, t
$\eta^{vm}_{s,d}(t)$	VNF mapping component of $\eta^{vnf}_{s,d}(t)$
$\eta^{vm}_{s,d}(t)$	VNF scheduling component of $\eta^{vnf}_{s,d}(t)$
$\mathbb{C}$	Set of all content items
c^{β}	The size of content item $c \in \mathbb{C}$
c^{L}	The packet size of content item $c \in \mathbb{C}$
$p^{1,c}_{k}(t)$	Local popularity of content item $c \in \mathbb{C}$ at $k \in \mathbb{S}$ at time slot t
$p^{2,c}_{k}(t)$	Proximity-based popularity of content item $c \in \mathbb{C}$ at $k \in \mathbb{S}$ at time slot t
$p^{c}_{k}(t)$	Hierarchical proximity-based of content item $c \in \mathbb{C}$ at $k \in \mathbb{S}$ at time slot t

$a_k^c(t)$	AoI of content item $c \in \mathbb{C}$ stored at $k \in \mathbb{N}$ at time slot t
a_{max}^c	Maximum tolerable AoI of content item $c \in \mathbb{C}$
$c^{\gamma,k}(t)$	Binary content freshness of content item $c \in \mathbb{C}$ stored at $k \in \mathbb{N}$ at time slot t
$\eta_k^{cnt,H}(t)$	Average cache hit rate per content item at $k \in \mathbb{N}$ at time slot t
$d_{k,h}^{cnt,c}(t)$	Placement delay to cache content $c \in \mathbb{C}$ at $k \in \mathbb{S}$ from $h \in \mathbb{N}$ at time slot t
$d_{k,h}^{txt,c}(t)$	Transmission delay of content item $c \in \mathbb{C}$ to $k \in \mathbb{S}$ from $h \in \mathbb{N}$ at time slot t
$\eta_k^{cnt,D}(t)$	Average cache placement delay per content at $k \in \mathbb{S}$ at time slot t
$\zeta_k^j(t)$	Binary variable if gateway $j \in \mathbb{J}$ is selected for satellite $k \in \mathbb{S}$
$\omega_{k,j}^{\mathbf{1},c}(t)$	Average feeder link bandwidth utilization due to status update traffic to cache content item $c \in \mathbb{C}$ on-board via $j \in \mathbb{J}$ to $k \in \mathbb{S}$
$\omega_{k,j}^{\mathbf{2},c}(t)$	Average feeder link bandwidth utilization due to content cache placement traffic to cache $c \in \mathbb{C}$ on-board via $j \in \mathbb{J}$ to $k \in \mathbb{S}$
$\eta_k^{cnt,L}(t)$	Total feeder link load per content item measured at $k \in \mathbb{S}$
$\mathbb{A}$	Set of ad content items
a^σ	Display duration of ad $a \in \mathbb{A}$
a^μ	Processing capacity requirement of ad $a \in \mathbb{A}$
c^R	Service subscription price of content item $c \in \mathbb{C}$ per bit
$\mathcal{R}_s^{sub}(t)$	Service subscription revenue to complete $s \in \mathcal{S}_{cnt}(t)$
a^R	Price per impression for ad $a \in \mathbb{A}$
$\mathcal{R}_s^{adv}$	Advertising monetized revenue from $s \in \mathcal{S}_{cnt}(t)$
$\mathcal{R}_s^{dis}(t)$	User disengagement cost of $s \in \mathcal{S}_{cnt}(t)$
$\mathcal{R}_s^{cos}(t)$	End user content delivery delay cost for $s \in \mathcal{S}_{cnt}(t)$
$\mathcal{D}_{min}$	Minimum allowable gap between two ad break points in a request
$x_k^{s,j}(t)$	Binary variable indicating if the VNF $f_{s,j} \in s^{vnf}$ is mapped on SEC node $k \in \mathbb{N}$ at time slot t.
$y_{h,k}^{s,j}(t)$	Binary variable indicating if the virtual link between the VNFs $f_{s,j-1} \in s^{vnf}$ and $f_{s,j} \in s^{vnf}$ mapping includes physical link between the nodes of $h \in \mathbb{N}$ and $k \in \mathbb{N}$
$\chi_{sj,gq}^k(t)$	Binary variable indicating if the VNF $f_{s,j} \in s^{vnf}$ has to wait compared to the corresponding VNF $f_{g,q} \in s^{vnf}$ to complete processing at node $k \in \mathbb{S}$ at time slot t

$x'^{c}_{k}(t)$	Binary variable indicating if content item $c \in \mathbb{C}$ is cached at node $k \in \mathbb{N}$ at time slot t.
$y'^{c}_{k,h}(t)$	Binary variable indicating if node $h \in \mathbb{N}$ is selected as content server to cache content item $c \in \mathbb{C}$ at node $k \in \mathbb{S}$ at time slot t.
$z^{c}_{u,k}(t)$	Binary variable indicating whether node $k \in \mathbb{N}$ serves requests for content item $c \in \mathbb{C}$ from user terminal $u \in \mathbb{U}$ at time slot t.
$z'^{c}_{u,a}(t)$	Binary variable indicating if ad $a \in \mathbb{A}$ is inserted into requests for content item $c \in \mathbb{C}$ from user terminal $u \in \mathbb{U}$ at time slot t.

List of Abbreviations

Abbreviation	Description
5G	Fifth Generation
6G	Sixth Generation
AP	Ad Provider
AoI	Age of Information
AR	Augmented Reality
AWS	Amazon Web Services
BPSO	Binary Particle Swarm Optimization
CAPEX	Capital Expenditure
CC	Cloud Computing
CCP	Convex-Concave Procedure
CDN	Content Delivery Network
COTS	Commercial Off-The-Shelf
CPU	Central Processing Unit
CSP	Cloud Service Provider (CSP)
DAI	Dynamic Ad Insertion
E2E	End-to-End
GA	Genetic Algorithm
GDPR	General Data Protection Regulation
GCP	Google Cloud Platform
GEO	Geostationary Earth Orbit
ILP	Integer Linear Programming Problem
INLP	Integer Nonlinear Programming Problem
IOL	Inter Orbit Link
IoT	Internet of Things

IoV	Internet of Vehicles
ISL	Inter Satellite Link
ISS	International Space Station
LEO	Low Earth Orbit
LXC	Linux Containers
MEC	Mobile Edge Computing
MILP	Mixed-Integer Linear Programming
MEO	Medium Earth Orbit
ML	Machine Learning
NTN	Non-Terrestrial Networks
NGSO	Non-Geostationary Orbit
NF	Network Functions
NFV	Network Function Virtualization
ONF	Open Networking Foundation
OPEX	Operational Expenditure
PaaS	Platform as a Service
PSO	Particle Swarm Optimization
QoE	Quality of Experience
QoS	Quality of Service
RQ	Research Question
SaaS	Software as a Service
SAGIN	Space-Air-Ground Integrated Network
SFC	Service Function Chain
SCC	Satellite-Backhauled Cloud Computing
SEC	Satellite Edge Computing
SatCom	Satellite Communication
s.t.	Subject to
STIN	Satellite-Terrestrial Integrated Network
STK	System Toolkit
SV	Server Virtualization
VSATs	Very Small Aperture Terminals
VM	Virtual Machine

VR	Virtual Reality
XR	Extended Reality

Notations

$\log(z)$	The natural logarithm of z.
e^z	Exponential function of z.
$\max$	Maximum operation.
$\min$	Minimum operation.
$z \in \mathcal{Z}$	Element z belongs to set $\mathcal{Z}$.
$z \notin \mathcal{Z}$	Element z does not belong to set $\mathcal{Z}$.
$\mathcal{Z} \cup \mathcal{Q}$	Union of set $\mathcal{Z}$ and set $\mathcal{Q}$.
$E[\cdot]$	Expected value.
max	Maximum operation.
min	Minimum operation.
$(\cdot)^T$	Transpose of $(\cdot)$.
$\lvert\mathbb{C}\rvert$	The size of vector $\mathbb{C}$.
$A \vert B$	Event A given that event B has occurred.

Chapter 1

Introduction

Cloud Computing (CC) revolutionizes the way we store, manage, and access data, enabling unprecedented scalability, flexibility, and cost-effective solutions. CC resources include physical or virtual servers for computing, storing, and networking capabilities [1]. This approach allows businesses and individuals to access powerful computing and storage capacities hosted at distant data centers managed by Cloud Service Providers (CSPs) [2], such as Amazon Web Services (AWS) [3], Google Cloud Platform (GCP) [4], IBM Cloud [5], and Microsoft Azure [6], without the need for excessive local IT infrastructure to effectively facilitate service provisioning through efficient allocation, configuration and management of the cloud resource pools. Satellite networks are widely regarded as a viable solution for addressing connectivity challenges in regions where the deployment of terrestrial networks is not feasible. Satellite-backhauled Cloud Computing (SCC) empowers remote, underserved, or disaster-stricken areas to access CC resources over satellite networks [7, 8]. The services are centralized in traditional SCC architecture, where all service requests need to reach the centralized cloud server for processing, storage, and access. Traditional satellite networks often use a bent-pipe architecture, where the satellite receives raw data and sends it back to the ground without processing it. While this centralized architecture approach may be beneficial for latency-tolerant services under low-scale traffic demands, it may not give real-time solutions to high-scale traffic demand conditions and latency-sensitive applications, such as mission-critical operations, Virtual Reality (VR), and Augmented Reality (AR), which require low latency and high reliability. This is because the link between the satellite and the cloud computing service provider is long, which results in a high propagation delay. Furthermore, service requests from remote users must be transmitted through limited-capacity feeder links to reach cloud servers, which can result in link congestion and service disruptions, especially under high traffic demand, thereby limiting the overall scalability of the system.

Motivated by the challenges of traditional SCC, a groundbreaking paradigm called Satellite Edge Computing (SEC) has emerged [9–11]. Inspired by the Mobile Edge Computing (MEC) framework [12], SEC revolutionizes the landscape by relocating computational and storage power and networking capabilities to space, creating in-orbit edge computing within satellite constellations [10, 11, 13–15]. This innovative approach addresses key challenges in traditional SCC by providing space-based computing and storage capabilities within the constellation that reduce reliance on remotely placed cloud servers, enhance scalability, minimize latency and bandwidth usage, and alleviate central server load. This framework offers ubiquitous service coverage, independent of terrestrial infrastructure, enabling reliable, low-latency service delivery anytime and anywhere. Compared to traditional centralized cloud models, this distributed architecture offers significantly improved service availability, resilience, and efficiency, making it a more suitable solution to meet the stringent requirements of next-generation applications.

The following section provides an overview of an SCC and an SEC. In addition, the motivation, methodology, scope, and contribution of the thesis are discussed.

1.1 SCC

SCC enables satellite operators to leverage CC resources such as computing, storage, and networking capabilities provided by cloud service vendors, eliminating the need to build and maintain costly private data center infrastructures. This paradigm allows users in rural, remote, and underserved areas and disaster-stricken regions to access cloud services via satellite constellations when there is no or limited ground-based infrastructure availability. This is especially crucial for disaster recovery, remote healthcare, and Internet of Things (IoTs) data transmission applications [16]. Furthermore, this paradigm enhances the performance, manageability, flexibility, and scalability of Satellite Communication (SatCom) operators. It improves operational efficiency, reduces Capital Expenditures (CAPEX), creates new business opportunities, and provides unprecedented agility for ground station operations [7, 8].

As shown in Figure 1.1, in SCC, when a remote user requests a cloud service, the request is transmitted via the user terminal to a satellite through the user uplink. The satellite then relays the request via a feeder downlink to the ground station gateway. From there, the gateway forwards the request to remotely located cloud servers for processing, retrieval, or content access, depending on the request type. Cloud access follows one of the three commonly used cloud computing service models: Infrastructure as a Service (IaaS), Platform as a Service (PaaS), or Software as a Service (SaaS) [17–19]. The IaaS model provides on-demand access to fundamental infrastructure resources, such as computing power, storage, and networking, over the Internet. This eliminates the need for users to invest in or maintain local data center infrastructure. Instead, users can provision and manage resources like Virtual Machines (VMs), containers, and virtualized network components directly from the cloud, enabling scalable and flexible infrastructure management [17]. The PaaS model offers a cloud-based platform with tools, middleware, and databases, allowing users to develop and manage applications without handling underlying infrastructure or local resource maintenance [19]. In the SaaS cloud service model, users are provided with a complete application stack that can be accessed and used over the internet without the need to install, manage, or deploy the application on local machines [18]. Once processed, the response is sent from the cloud computing servers through the terrestrial infrastructure to the ground station gateway. The gateway then transmits the response to the satellite via the feeder uplink, and finally, the satellite beams it down to the user terminal. The network management and orchestration are performed in the Central Network Hub (CNH). Multiple gateways connect to this hub to receive services. The CNH determines how to execute, store, and deliver these services based on an optimization strategy for each gateway.

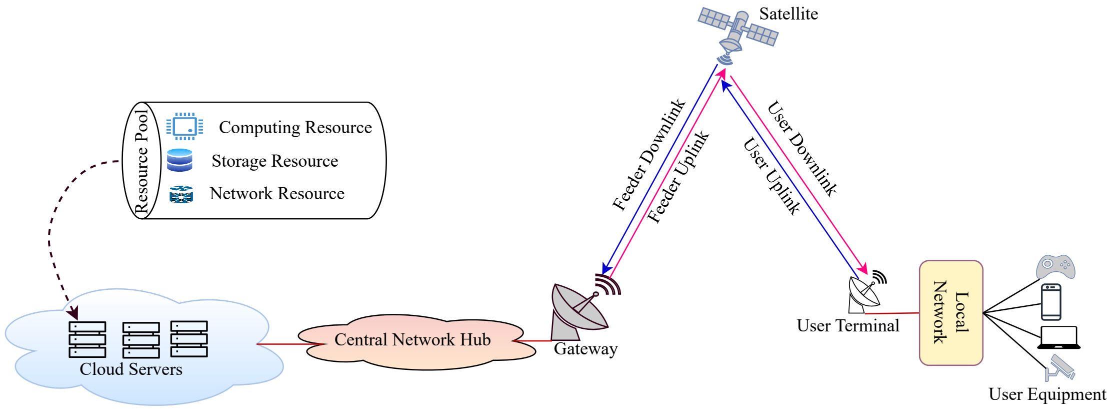


Figure 1.1: Traditional satellite network architecture for cloud computing service

In traditional satellite networks for cloud computing, service requests must be transmitted to centralized cloud servers for processing, as satellites operate under a bent-pipe model, merely relaying data without on-board processing or storage. Furthermore, current regenerative satellites [20] mainly handle signal processing and lack full MEC capabilities for efficient service execution. As a result, both bent-pipe and regenerative architectures depend heavily on ground cloud infrastructure. This architecture introduces high service delays due to long transmission distances and leads to excessive bandwidth consumption, particularly on the feeder links. On the other hand, as data volumes and user demands grow, this centralized approach struggles to meet the stringent requirements of next-generation applications, causing inefficiencies and bottlenecks. To address these challenges, a new paradigm, SEC, has emerged, inspired by terrestrial MEC [9, 13, 21, 22].

1.2 SEC

Advancements in satellite processing capabilities have progressed rapidly, with major suppliers such as Airbus and Honeywell now offering on-board processors [23, 24]. These high-performance processors enable satellites to perform complex data processing tasks directly on-board, significantly enhancing operational efficiency and reducing dependency on ground stations. Similarly, satellite storage capabilities have advanced substantially, with modern satellites now managing terabytes of data storage [25]. This increased storage capacity enables satellites to handle and analyze large volumes of data locally, reducing reliance on continuous communication with ground stations. It also supports real-time decision-making, critical for latency-sensitive and mission-critical applications. These technological advancements in processing and storage capabilities collectively empower more autonomous and efficient satellite operations, laying a strong foundation for the effective implementation of SEC.

Conventional centralized cloud computing may not be efficient for latency-sensitive services (*e.g.,* real-time gaming or video conferencing), and it can also become a bottleneck in meeting stringent requirements as data and service demands grow. SEC architecture is a new paradigm designed to overcome the limitations of traditional satellite networks in cloud computing services by placing cloud computing resources in a satellite constellation [13]. In the SEC, computational and storage resources are deployed in space, enabling satellite nodes to perform cloud computing tasks in orbit instead of relying solely on terrestrial data centers. Furthermore, this architecture facilitates task execution through collaboration among satellites via Inter-Satellite Links (ISLs) and Inter-Orbit Links (IOLs) (*i.e.,* Free Space Optical Links (FSO) or Radio Frequency (RF) wireless links), enhancing performance by efficiently solving complex tasks in a distributed manner [11]. The adoption of SEC provides two key advantages: (1) ***computation offloading***, enabling users to access extensive computational resources in space, and (2) ***content caching/storage***, allowing satellites to cache content and store data files, thereby reducing redundant transmissions from remote cloud data centers [9, 13, 21, 22].

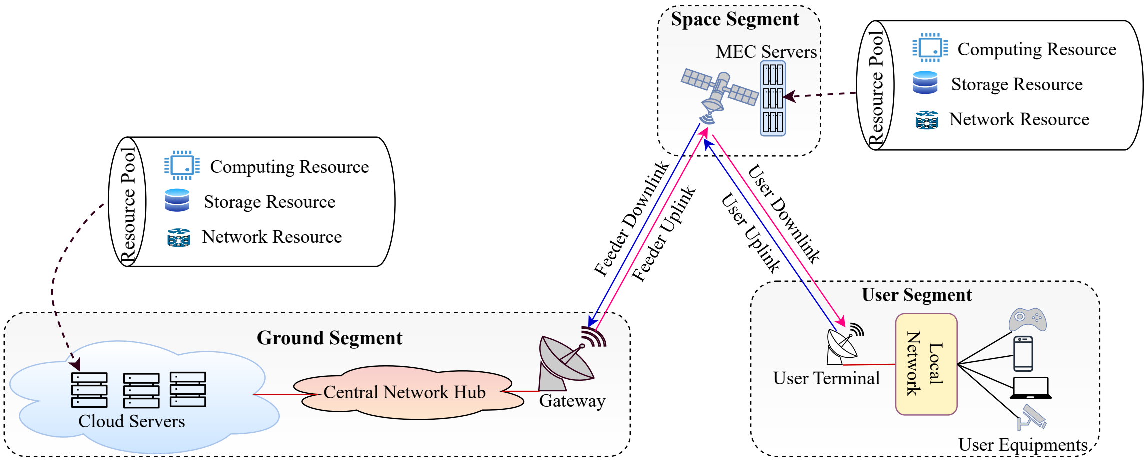


Figure 1.2: SEC network architecture for cloud computing service

Figure 1.2 illustrates an end-to-end SEC-enabled satellite network architecture for cloud computing service provisioning. The architecture comprises three main components: the ground, space, and user segments. The detailed functional components of the SEC-enabled satellite network are discussed as follows:

1. **Ground Segment**: The Ground Segment manages, controls, and orchestrates the cloud service provisioning process. It includes several cloud servers with vast cloud resources, encompassing computing, storage, and networking. The computing resources are used for processing data requests, the storage resources are responsible for storing content items, and the networking resources provide connectivity, such as routing. Furthermore, the ground segment includes the CNH. The CNH manages, controls, and orchestrates resources to provide seamless cloud service. It also integrates cloud servers and satellite resources to ensure seamless content delivery and efficient service provisioning. The CNH in a network can be centralized or distributed, depending on the system design. This configuration is selected to enhance system performance by improving the

responsiveness and scalability of the CNH. In addition, the ground station gateways act as communication bridges between ground-based systems and satellites, receiving and transmitting service requests via feeder downlinks and uplinks, respectively, to ensure smooth communication between segments.

2. **Space Segment**: The space segment includes MEC servers within satellite nodes, providing computing, storage, and networking resource pools that enable distributed computing, storage, and networking capabilities, allowing efficient content caching, processing, and dynamic decision-making. The Space Segment of the network architecture includes satellites classified into three categories based on their orbital positions: Geostationary Earth Orbit (GEO), Medium Earth Orbit (MEO), and Low Earth Orbit (LEO). Each type of satellite serves a distinct role within the overall system, with its specific characteristics influencing data transmission, content caching, and resource management [26]. GEO satellites are positioned approximately 35,786 km above the Earth and remain stationary relative to the Earth's surface, providing broad coverage and continuous connectivity for large areas. While they have higher latency, GEO satellites are ideal for global content distribution and backhaul communication. MEO satellites orbit approximately 20,200 km, serving as intermediaries between GEO and LEO satellites. They balance coverage and latency, making them particularly useful for data aggregation, content prefetching, and load balancing across different network segments. LEO satellites are positioned at altitudes between 300 km and 2,000 km, offering low latency due to their proximity to Earth. This makes them well-suited for real-time applications such as live streaming, dynamic ad insertion, and mission-critical services. Equipped with MEC servers, LEO satellites enable edge computing capabilities, local data processing, and efficient content caching, which reduces dependency on ground-based servers. Therefore, a multi-layer satellite network that includes SEC-enabled LEO, MEO, and GEO satellites offers comprehensive service provisioning that can meet the diverse requirements of various applications. This architecture ensures high reliability, resilience, and the ability to satisfy stringent user demands.

3. **User Segment**: The User Segment includes the user terminals, local networks, and User Equipment (UE). The user terminal serves as the data reception and transmission gateway, connecting UEs (e.g., smartphones, laptops, and IoTs) to the satellite network

via the user downlink and uplink. This segment is where cloud service requests are generated. Users submit service requests to the satellite through their local network user terminal, and the request is processed based on the service type and the policies obtained from the CNH. Actions are then taken to fulfill the service request according to user requirements. For example, services may be processed locally on the satellite or cloud servers, depending on user requirements and resource availability in each network segment.

Advantages of SEC

SEC offers several advantages over traditional SCC by processing data closer to end users rather than relying solely on centralized cloud servers. The key benefits include:

- **Reduced Latency:** The computational and storage capabilities in space enable users in remote areas to access and complete service requests with minimal delay, rather than relying on centralized cloud servers. This approach ensures ubiquitous service coverage and significantly reduces latency, making it well-suited for time-sensitive applications. These include mission-critical services, disaster recovery, real-time services, live video streaming, IoT, and sensor networks, where immediate data processing is essential.

- **Reduced Bandwidth Consumption:** Service provisioning on-board satellites in an SEC-enabled satellite network reduces the data traffic transmitted to centralized cloud servers, minimizing bandwidth consumption, particularly on feeder links. This approach alleviates feeder link load, reduces traffic congestion, and lowers communication costs by processing and completing services on-board.

- **Improved Security and Privacy:** Completing service requests closer to the user terminal in an SEC-enabled network reduces the need to traverse multiple network hops compared to traditional SCC. This minimizes the risk of cyber threats and data breaches, enhancing data security and privacy. Additionally, it improves data confidentiality and ensures better compliance with data privacy regulations such as the General Data Protection Regulation (GDPR) [27].

- **Improved Scalability** Distributed decision-making and the ubiquitous availability of cloud resources near end users enhance service accessibility, supporting a larger number of users and accommodating large-scale data volumes.

Key Enabling Technologies of SEC

SEC leverages various key enabling technologies to achieve these advantages, ensuring high performance, scalable, and efficient services for remote users. These technologies enable efficient computational offloading and content caching/storage, including:

- **Network Virtualization**: Network Virtualization (NV) is a crucial technology that enhances the flexibility and scalability of SEC by abstracting and decoupling network functions from hardware. It creates multiple virtual network environments while sharing the same physical infrastructure [28]. This enables dynamic management and configuration of network resources based on user demand, allowing service provisioning tailored to their requirements. NV leverages two key emerging technologies: Software Defined Networking (SDN) [29] and Network Function Virtualization (NFV) [30], which enable multiple virtual networks to operate over a shared physical infrastructure. This allows for network segmentation, isolation, and resource allocation tailored to different services and users based on predefined configurations, plans, or designs.

  - *SDN*: SDN is an emerging network architecture advanced by the Open Networking Foundation (ONF) where network control is decoupled from forwarding and is directly programmable [29, 31]. SDN decouples the control plane from the data plane, allowing for flexible and efficient network management and operation via software programs. This agility enhances innovation, ensuring that service requests meet their requirements. An SDN controller has a global view of network elements and can improve network performance through programmability. It dynamically optimizes network operations based on real-time status, enhancing data traffic scheduling, end-to-end congestion control, load balanced packet routing, and Quality of Service (QoS) support. Additionally, SDN reduces costs by simplifying forwarding devices and streamlining network operations.

  - *NFV*: NFV is an emerging technology that decouples network functions from dedicated hardware, enhancing flexibility and scalability while meeting user requirements [30, 31]. NFV enables network functions to run on general purpose Commercial Off-The-Shelf (COTS) computing hardware, storage, and networking infrastructure. Compared to traditional proprietary network function devices, this significantly reduces CAPEX and Operational Expenditure (OPEX). With NFV, a service can be decomposed into Virtual Network Functions (VNFs) that operate as software instances running on industry standard physical servers. This allows consolidating multiple network equipment types onto high-volume servers, switches, and storage in data centers, distributed network nodes, or end user premises. NFV employs virtual resources, abstracting computing, storage, and network resources through a hypervisor, effectively decoupling virtual resources from their underlying physical infrastructure. Virtual networks in NFV consist of: 1). Virtual Nodes Software components with either hosting or routing functionality (*e.g.*, an OS running in a virtual machine). 2). Virtual Links Logical interconnections between virtual nodes, appearing as direct physical links. A service in NFV comprises one or more NFs, which must be executed sequentially to complete a service request. The ordered sequence of VNF components that make up a service is called Service Function Chaining (SFC). SFC ensures that network services are dynamically orchestrated to meet user demands efficiently.

- **Server Virtualization**: Server Virtualization (SV) is an emerging technology that abstracts physical server resources, such as Central Processing Unit (CPU), memory, and storage, into multiple virtual nodes, each with its own virtualized resources (e.g., virtual CPU, memory, and storage) [32]. These virtual nodes operate independently, accommodating diverse service requests while enhancing flexibility and scalability to meet user demands. This technology efficiently utilizes Satellite-backhauled server nodes, maximizing resource allocation and flexibility. Hypervisors, such as VMware [33], KVM [34], and Xen [35], facilitate SV by managing multiple VMs on a single physical server. There are two primary types of SV technologies: 1). VMs provide complete isolation by virtualizing hardware resources. Each VM runs its complete operating system (including its kernel) on top of a hypervisor [36]. 2). Containers are lightweight and provide isolation at the OS level [37]. They share the host OS kernel and run as isolated processes in user space. Popular containerization technologies include Docker [38] and Linux Containers (LXC) [39], which allow multiple containers to run on a single physical server node. The choice between VMs and containers depends on the level of virtualization required. VMs offer more substantial isolation and are suitable for applications needing separate OS environments, while containers provide greater efficiency and speed due to their lightweight nature.

- **Content Delivery Network (CDN)**: A CDN is an emerging technology that utilizes a distributed network of servers to deliver content faster and more efficiently to end users [40]. In a CDN, content is cached in distributed nodes closer to users, reducing delivery delay by offloading requests from the original servers, primarily cloud servers. This approach enhances system performance by reducing latency, as content is served from nearby nodes instead of distant cloud servers. It also improves reliability by ensuring content availability through redundant caching across multiple nodes, even in the case of failures. Additionally, scalability is enhanced as distributing content across multiple nodes allows the system to handle increased user demand more efficiently. CDN is crucial in SEC-enabled satellite networks, enabling content caching and distribution for remote users [41]. It supports various services by optimizing content delivery based on service requests and user requirements.

1.3 Motivation

With the rapid proliferation of internet-connected devices and continuous advancements in data quality, the demand for data-intensive applications is growing at an unprecedented rate. According to the Ericsson Mobility Report (2024), global mobile data traffic is expected to grow more than 2.85 times by 2030 compared to 2024 [42]. Furthermore, this significant growth is driven by the increasing adoption of 5G technology and the proliferation of high-bandwidth applications such as ultra-high-definition video streaming (4K, 8K, and 16K), as well as extended reality (XR) services, including augmented reality (AR), virtual reality (VR), and mixed reality (MR). This surge in data consumption presents significant challenges for traditional cloud-based service provisioning, particularly in ensuring low latency, high reliability, and efficient bandwidth utilization. On the other hand, according to a report by Nutanix, nearly 75% of companies have migrated their mission-critical applications to the cloud [43]. The need for greater flexibility, adaptability, and resilience in business operations drives this shift. These applications impose stringent requirements, as their failure can result in significant losses, including financial damage, operational disruptions, and even threats to human life, such as in disaster recovery, military operations, and economic analysis. A comprehensive service provisioning scheme is required to meet the stringent requirements of modern applications, particularly in terms of bandwidth and delay. However, according to the ITU Report (2023), approximately 33% of the global population lacks internet access, primarily due to the absence of terrestrial network coverage [44], which extends to only 20% of the globe [45]. Moreover, the terrestrial network is vulnerable to natural disasters like floods, earthquakes, and tsunamis. This digital divide underscores the need for alternative network infrastructures, such as satellite-based solutions, to ensure seamless connectivity and service accessibility in underserved regions, leveraging their global coverage [46].

The traditional bent-pipe satellite network architecture [9] is inadequate for meeting the stringent requirements of diverse and heterogeneous applications that demand high quality and low latency. In contrast, the SEC-enabled satellite network is better suited for such applications due to its ubiquitous coverage, high availability, low latency, and high bandwidth, effectively addressing the growing demand for service provisioning. SEC-enabled satellite networks are crucial in addressing growing demands and complementing terrestrial infrastructure. However, conventional networks and standalone SEC-enabled satellite deployments are insufficient to meet the increasing user demand due to several key factors:

- **Limited Computing and Storage On-Board Resources**: As mentioned, SEC-enabled networks facilitate computational offloading and content caching/storage [15, 47] to enhance service provisioning by offering further service innovation and business agility. However, satellite nodes have significant computational limitations [11], making them unsuitable for processing highly compute-intensive services. In SDN/NFV-enabled networks, services are represented as sequentially ordered VNF components in an SFC request. The service is completed when all its VNF components are executed in the specified order [48]. Due to limited onboard computational capacity, a single satellite node is often unable to execute all the VNF components of an SFC request, especially when dealing with computation-intensive services or a high number of concurrent requests. To address this limitation, the VNF components within an SFC request need to be executed separately across multiple satellite nodes within the constellation, depending on their computational requirements [49–51]. Similarly, the limited storage capacity of satellite nodes [14, 51] makes storing all requested content on a single node infeasible. As a result, content items must be stored across multiple satellite nodes. While this distributed approach to both VNF execution and content storage enables the fulfillment of complex service requests, it introduces additional complexity and risks violating the required service performance guarantees.

- **Dynamic Channel Conditions and Limited Link Capacity:** The channel conditions between satellites and ground users, as well as between satellites themselves, vary dynamically due to weather conditions and mobility. Not only do channel characteristics such as bandwidth capacity and propagation delay fluctuate due to satellite mobility and environmental factors, but link availability is also affected [49,50,52]. Consequently, satellite communication links exhibit high variability in bandwidth capacity and propagation delay due to the movement of Non-Geostationary Orbit (NGSO) satellites, atmospheric interference, and line-of-sight obstructions. This results in unpredictable QoS, particularly for real-time applications, and adversely affects computational offloading, routing, and content distribution. Additionally, the links in the network have limited bandwidth capacity, which may lead to congestion during high-demand scenarios, thereby degrading system performance.

- **Highly Dynamic Network Topology:** Satellites in NGSO constellations, such as LEO and MEO satellites, move around the Earth at very high speeds. For instance, LEO satellites span approximately 7.8 km per second relative to Earth, completing an orbit in about 90 minutes. As a result, the International Space Station (ISS), which also operates in LEO, orbits the Earth roughly 16 times per day [53]. In an NGSO constellation, the network topology constantly changes as satellites move in orbit. The dynamic nature of satellite networks presents significant challenges in service provisioning, including computational offloading, content caching, and distribution. Satellite mobility can lead to improper placement of VNFs, content, and routing paths, causing service disruptions and violations of service requirements, resulting in overall system performance degradation.

- **Dynamic and Diverse Service Request Requirements:** Different applications have varying demand requirements [54], such as E2E service delay and bandwidth. Meeting these diverse requirements necessitates adaptive and dynamic service provisioning, which can allocate resources efficiently in real-time based on user needs and network conditions. Furthermore, the dynamic spatial and temporal variations in service requests pose a significant challenge in efficiently responding to user demands while meeting user requirements [54,55]. For example, users in densely populated urban areas generate more frequent and concentrated service requests compared to those in remote or rural regions. Additionally, service requests and requirements vary widely, ranging from time-sensitive, mission-critical, and IoT applications to data-intensive services such as 4K/8K videos, VR, and AR. Meeting these heterogeneous and often complex requirements is particularly challenging, especially in large-scale systems. The dynamic and diverse nature of these requests makes designing an optimal service provisioning scheme increasingly complex.

- **Scalability and Service Orchestration:** The massive and complex multi-orbit constellations present another challenge for service provisioning and resource orchestration. For instance, according to Orbiting Now, as of March 5th, 2025, there are 11,833 active satellites across various Earth orbits [56, 57]. The Starlink satellite megaconstellation currently consists of 7,086 satellites, 7,052 of which are operational. SpaceX ultimately aims to deploy up to 42,000 satellites in this megaconstellation [58, 59]. Managing resources in this vast, heterogeneous, and dynamic satellite network remains a significant challenge, particularly in meeting user requirements. As the satellite network scales with an increasing number of nodes and user terminals, effective resource orchestration becomes more complex. This underscores the need for intelligent orchestration frameworks that span multiple satellite layers (LEO, MEO, GEO) to ensure consistent service quality and performance.

The SEC-enabled network can operate more efficiently by addressing specific challenges. However, additional issues must be tackled during service provisioning to further enhance its performance. The key challenges include:

- **Strategic Storage and Computing Placement**: Along with the inherent resource constraints of on-board computing and storage, strategic resource placement is crucial for efficient service provisioning in SEC-enabled satellite networks. For instance, in SFC requests, system performance depends on where the VNF components of each service are executed [60, 61]. Similarly, for content retrieval or distribution requests, such as multimedia applications, performance is influenced by the CDN node from which the content is fetched [62, 63]. Therefore, strategic resource placement is vital in meeting user requirements and must be carefully considered during service provisioning in SEC-enabled satellite networks.

- **On-board Resource Optimization**: Beyond the challenges of on-board resource placement, efficiently managing available communication, computational, and storage resources is essential for improving overall system performance. For instance, executing all VNF components of an SFC request within a single satellite node is often infeasible due to computational limitations for processing and communication constraints for forwarding while meeting user requirements [60,61]. Similarly, storing all content items on a single satellite node is impractical due to storage limitations and communication constraints that affect content distribution [62, 63]. Therefore, an optimal on-board resource optimization strategy is crucial to effectively meet end-user requirements.

1.4 Research Questions

Following a detailed understanding of the limitations of traditional SCC systems and the challenges in service provisioning within SEC-enabled satellite networks, this thesis investigates the following research questions to develop novel methods for optimizing service provisioning in next-generation networks.

Research Question 1 (RQ1)

How can the VNF components of a mission-critical application service request be efficiently deployed on resource-constrained satellite nodes within a time-varying SEC-enabled satellite network topology, while accounting for dynamic service request patterns across time and space to meet their diverse and stringent performance requirements?

In this work, we focus on optimizing VNF mapping and scheduling to meet the stringent requirements of mission-critical applications while considering the computational constraints of on-board resources and bandwidth capacity. This is achieved by enhancing fairness in E2E service delay margins among competing service requests. Additionally, we examine the impact of service continuity and the challenges posed by time-varying network topologies. Finally, we analyze the temporal variability of link quality regarding propagation delay, bandwidth capacity, and availability to assess their impact on overall system performance.

Research Question 2 (RQ2)

How can on-board resources be efficiently utilized for content caching to improve system performance while minimizing cache reconfiguration overhead in a dynamic-topology, resource-constrained, SEC-enabled multi-layer satellite network, while ensuring content freshness and accommodating spatiotemporal service requests with heterogeneous requirements?

This work presents on-board content cache placement strategy focusing on the trade-off between system performance and content cache reconfiguration overhead, specifically cache update delay and feeder link load. Furthermore, we study and model the impact of content freshness and relevance on system performance. Additionally, we model content caching strategy for efficient on-board resource utilization to enhance performance in multi-layer SEC-enabled satellite networks while accounting for on-board resource limitations, dynamic topology, and the massive, heterogeneous, and dynamic traffic demands of non-uniformly distributed users.

Research Question 3 (RQ3)

How can a joint ad insertion, content caching, and distribution method be developed to maximize content provider revenue in time-varying and resource-constrained MEC-enabled Satellite-Terrestrial Integrated Networks (STINs), while accommodating dynamic service requests with heterogeneous requirements, reducing service disruption, and ensuring end-user satisfaction?

In this work, we study an efficient joint ad insertion, content caching, and distribution scheme while incorporating a service continuity mechanism in SEC-enabled multi-layer STINs. This work addresses the challenge of content delivery disruptions caused by service continuity constraints, dynamic satellite mobility, and link availability. Furthermore, we investigate the trade-off between content provider revenue generation through ad monetization and the potential costs associated with user disengagement, while accounting for dynamically distributed content requests and heterogeneous service requirements.

1.5 Scope of the Thesis and Research Methodology

This section presents the methodology for designing optimized service provisioning in SEC-enabled satellite networks. Figure 1.3 illustrates the research methodology adopted in this thesis. First, a comprehensive review of the state-of-the-art (SOTA) in conventional and SEC-enabled satellite networks was conducted to identify potential research gaps. Based on these gaps, an SEC-enabled satellite network system model was designed and formulated for the research gap problem. In the problem formulation stage, the objective function(s) were strategically selected among the potential key performance metrics, accompanied by all relevant constraints, to represent the corresponding research gaps as an optimization problem. Next, appropriate solution methods were proposed based on the class of the formulated problem. The proposed solution was then compared to benchmark approaches and the optimal solution to evaluate the optimality gap. Finally, the conclusions were outlined.

In the system model, the time-varying topology of the SEC-enabled network is considered, while user distribution is assumed to be randomly distributed geographically. Furthermore, service requests are modeled using a Poisson distribution. Before formulating the optimization problem, the objective function is identified based on the objective of the research idea, which may be either a single-objective or multi-objective function. The problem is then formulated by incorporating the objective function and relevant constraints. The formulated optimization problem can be classified as either convex or non-convex, depending on the nature of its objective function and constraints.

- **Convex Optimization Problem**: An optimization problem is convex if its objective function is convex, and the feasible region defined by the constraints is also convex. A function $f(x)$ is convex if:

  $$f(\lambda x_1 + (1-\lambda)x_2) \leq \lambda f(x_1) + (1-\lambda)f(x_2), \quad \forall x_1, x_2 \in \mathbb{R}^n, \lambda \in [0,1]$$

  Convex problems guarantee global optimality, as any local minimum is also a global minimum, making them easier to solve and highly practical for various applications.

- **Non-Convex Optimization Problem**: An optimization problem is non-convex if the objective function or any constraint is non-convex, meaning the feasible region may have multiple local optima instead of a single global optimum. The local minimum of the problem may not be the same as the global minimum of the problem. Mathematically, a function is non-convex if it fails the convexity condition (*i.e.,* $f(\lambda x_1 + (1-\lambda)x_2) \leq \lambda f(x_1) + (1-\lambda)f(x_2), \quad \forall x_1, x_2 \in \mathbb{R}^n, \lambda \in [0,1]$). Unfortunately, many service provisioning schemes are classified as non-convex optimization since they

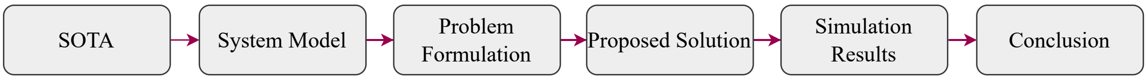


Figure 1.3: Research methodology.

involve non-linear functions as well as integer variables. This thesis typically considers service provisioning methods, non-linear, and combinatorial optimization. Hence, convexification and reformulation techniques are required for this optimization to obtain sub-optimal solutions. In this thesis, Success Convex Approximation (SCA) [64], heuristic, metaheuristic and combinatorial methods are used separately or together to solve the formulated problems. Furthermore, the tools used to solve the optimization problems are MATLAB with CVX solver [65]. However, in this thesis, we employ heuristic and metaheuristic approaches, as the computational complexity of optimal solution methods is very high, making them infeasible for resource-limited environments such as SEC-enabled on-board systems.

Furthermore, in MATLAB, extensive simulations are conducted using realistic datasets, including satellite constellation data and user distributions, to evaluate the effectiveness of each method. This thesis focuses on service provisioning schemes where SDN controllers orchestrate the strategy process. However, the placement of SDN controllers is beyond the scope of this thesis. Additionally, the channel model is not considered; instead, channels are assumed to be error-free and ideal.

1.6 Contributions and Related Publications

This section presents the contribution of the thesis and the related publication as the result of the research, peer-reviewed journal, and conference papers that have already been published and are currently under review. These papers are listed below in the text under $J \equiv$ Journals, $C \equiv$ Conferences.

Chapter 2

In this chapter, we develop an optimization framework for VNF mapping and scheduling in SDN/NFV-enabled SEC-enabled satellite networks, addressing **RQ1**. Our study presents a novel VNF mapping and scheduling strategy for mission-critical applications, advancing SEC for timely and reliable communication. The approach holds potential for applications in disaster response, remote medical care, and other time-sensitive operations. In this framework, services are represented as sequentially ordered VNF components, and a service request is considered complete if all its VNF components are executed on-board satellite nodes while meeting the required constraints. To reduce computational complexity, we employ a decomposition scheme that divides it into two subproblems: VNF mapping and VNF scheduling. These subproblems are transformed using the Convex-Concave Procedure (CCP) to facilitate linearization and are then solved efficiently using CVX. Additionally, we employ SCA to find an optimal solution. Although this approach is optimal, it is computationally intensive. Given the resource limitations in SEC-enabled satellite networks, we propose a metaheuristic approach to solve the problem efficiently. Specifically, we use a Greedy-based solution for fast computations on small-scale problems. For larger-scale scenarios, we employ Simulated Annealing (SA), using the Greedy solution as an initial input, to refine the solution and achieve near-optimal results. This hybrid approach balances computational efficiency with solution quality, as the analytical decomposition scheme is computationally intensive. This approach selects and compares a state-of-the-art benchmark with our approach. Furthermore, our approach is compared to the optimal solution to evaluate the optimality gap or penalty, assessing how far our proposed solution is from the optimal solution. We also evaluate the proposed solution's time complexity, both for the optimal solution and the benchmark. The simulation results validate that the proposed solution significantly improves system performance with low time complexity.

Related Publications

[J1]. **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,"in IEEE Transactions on Network and Service Management, vol. 21, no. 6, pp. 6716-6730, Dec. 2024, doi: 10.1109/TNSM.2024.3452031.

[C1]. **H. G. Abreha**, H. Chougrani, I. Maity, V. -D. Nguyen, S. Chatzinotas and C. Politis, "Fairness-Aware Dynamic VNF Mapping and Scheduling in SDN/NFV-Enabled Satellite Edge Networks,"ICC 2023 - IEEE International Conference on Communications, Rome, Italy, 2023, pp. 4892-4898, doi: 10.1109/ICC45041.2023.10279545.

Chapter 3

In this chapter, we present an on-board content cache placement strategy in a multi-layer SEC-enabled satellite network, which includes satellites in LEO, MEO, and GEO orbital constellations to address **RQ2**. We primarily focus on content requests that demand stored content. This research involves three main subproblems: *(i)* efficient on-board resource utilization, *(ii)* maintaining a trade-off between system performance and cache reconfiguration overhead, and *(iii)* ensuring cached content relevance, timeliness, or freshness. To tackle these challenges, we formulate a joint optimization problem with a multi-objective function. To solve the problem, we employ a Greedy algorithm for fast, low-scale solutions and a Genetic Algorithm (GA)-based approach, using the Greedy solution as an initial point to refine and achieve a near-optimal solution. Our approach is evaluated against selected benchmarks. We design the satellite network using a realistic satellite constellation dataset and model user distribution based on the geographical population distribution in the U.S.. Additionally, a Machine Learning (ML) approach is utilized for randomized user selection. Furthermore, our solution is compared to benchmarks to assess its performance and evaluated against the optimal solution, solved using CVX, to measure the optimality gap, quantifying how close our solution is to the optimal. Extensive simulations demonstrate that our proposed approach outperforms all benchmarks and achieves near-optimal performance.

Related Publications

[J2]. **H. G. Abreha**, I. Maity, H. Chougrani, C. Politis, and S. Chatzinotas, “ On-board content caching in multi-layer satellite edge networks with dynamic cache reconfiguration,”IEEE Open Journal of the Communications Society, (Under Review)

[C2]. **H. G. Abreha**, I. Maity, H. Chougrani, C. Politis and S. Chatzinotas, “Resource-Aware On-board Content Caching in Multi-Layer Satellite Edge Networks,” ICC 2024 - IEEE International Conference on Communications, Denver, CO, USA, 2024, pp. 3943-3949, doi: 10.1109/ICC51166.2024.10622513.

Chapter 4

In this chapter, we present a joint on-board content caching and seamless content distribution strategy with Dynamic Ad Insertion (DAI) in MEC-enabled multi-layer STINs, integrating LEO, GEO, and ground terrestrial networks. In this research, we focus on seamless content distribution, which can only be achieved through service continuity-aware content caching and distribution, considering service request continuity and topology variability. Furthermore, we incorporate advertisement monetization to generate content provider revenue while modeling its impact on user engagement costs. We model the service disruption cost arising from early content cache dispatch due to the time-varying topology and user disengagement caused by excessive, unplanned, and unoptimized ad insertions in content requests. Finally, we formulate the problem as a joint optimization problem, where the objective function captures the trade-off between content provider revenue and end-user requirements. We solve this problem using metaheuristic algorithms, initially employing a Greedy-based solution for fast computation, suitable for small-scale problems with low computational complexity. The choice of a metaheuristic approach is driven by its simplicity and efficiency in handling complex optimization scenarios. Furthermore, considering the Greedy-based solution as an initial step, we propose a Binary Particle Swarm Optimization (BPSO)-based content distribution strategy as an enhanced solution. This approach incorporates a collaborative mechanism to handle unassociated requests, ensuring a more refined solution for large-scale problems while maintaining reasonable time complexity and achieving near-optimal performance. We compare the proposed solution with selected benchmarks to evaluate its performance. Additionally, we solve the problem using the CVX optimization solver to determine how close our solution is to the optimal one, measure the penalty deviation from optimality (optimality gap), and compare its time complexity with the optimal solution to quantify the computational efficiency gain. The results demonstrate that our proposed solution outperforms the benchmarks and achieves near-optimal performance with significantly lower computational complexity.

Related Publications

[J3]. **H. G. Abreha**, I. Maity, Y. Drif, C. Politis, and S. Chatzinotas, "Revenue-Aware Seamless Content Distribution In Satellite-Terrestrial Integrated Networks,"IEEE Transactions on Network and Service Management, (Under Review)

Chapter 5

In this chapter, we present the main conclusions of the thesis and discuss potential directions for future research.

Chapter 2

VNF Mapping and Scheduling in Satellite Edge Networks

SEC is seen as a promising solution for deploying NFs in orbit to provide ubiquitous services with low latency and bandwidth. SDN and NFV enable SEC to manage and deploy services more flexibly. In this chapter, we study a dynamic and topology-aware VNF mapping and scheduling strategy within an SDN/NFV-enabled SEC infrastructure. Our focus is on meeting the stringent requirements of mission-critical applications, recognizing their significance in both satellite-to-satellite and edge-to-satellite communications while ensuring service delay margin fairness across various time-sensitive service requests. We formulate the VNF mapping and scheduling problem as an Integer Nonlinear Programming (`INLP`), with the objective of *minimax* fairness among specified requests while considering dynamic satellite network topology, traffic, and resource constraints. We then propose two algorithms for solving the `INLP` problem: Fairness-Aware Greedy Algorithm for Dynamic VNF Mapping and Scheduling (`FAGD_MASC`) and Fairness-Aware SA-Based Algorithm for Dynamic VNF Mapping and Scheduling (`FASD_MASC`), which are suitable for low and high service arrival rates, respectively. Extensive simulations demonstrate that both `FAGD_MASC` and `FASD_MASC` approaches are very close to the optimization-based solution and outperform the Tabu search VNF remapping and rescheduling (`TS_MAPSCH`) benchmark. In particular, `FASD_MASC` achieves over 10% and 15% improvements in service acceptance rate and fairness, respectively.

2.1 Introduction

The ever-increasing demand for mission-critical applications in 6G has attracted the attention of researchers [66]. Mission-critical applications typically deal with very serious situations, such as disaster management, rescue, and military operations, with stringent requirements such as latency and reliability, as their failure can result in the loss of lives and property [67]. Therefore, network architectures that provide high Quality Of Service (QoS) and network coverage are crucial to meet the requirements. Terrestrial networks are vulnerable to natural disasters, such as earthquakes and hurricanes. In this context, satellite networks can be an alternative solution to overcome the challenges of network disruption.

Mission critic applications rely on satellite networks during terrestrial communication failures, particularly in scenarios requiring low latency and real-time decision-making supported by satellite-to-satellite links, such as emergency response operations [68]. The urgency of the services varies: for example, early warnings require rapid responses [69], while relief logistics can tolerate some delays [70]. Therefore, effective demand-driven service provisioning is crucial in satellite networks to accommodate these diverse requirements [71]. However, the traditional *bent-pipe* satellite architecture, which involves raw data transmission from satellites to remote terrestrial data centers without on-board processing, is not suitable for mission-critical applications. This architecture suffers from limitations in latency and bandwidth consumption, hindering the timely delivery and processing of critical information and impeding swift decision-making and response. SEC addresses this challenge by providing computing resources on-board in a *compute-as-a-service* model [13,15,72]. In this paradigm, satellites are equipped with microservers, enabling in-orbit data processing without relying on remote terrestrial data centers [72]. An SEC-enabled network facilitates ubiquitous, low latency, and bandwidth efficient service provisioning, making it particularly suitable for mission-critical applications.

Despite the advantages of SEC in the delivery of services for mission-critical applications, effectively managing the diverse and dynamic nature of network resources, such as computing, storage, and bandwidth, and traffic demands poses a challenge [15]. NFV [30] and SDN [29] are two networking virtualization technologies that address this challenge by enabling agile and flexible network management and orchestrating service delivery. In an SDN/NFV-enabled environment, network services are represented as sequentially ordered VNFs known as SFC [60]. In an SEC-enabled network, an SFC request is completed by executing its VNF components on the SEC nodes on-board[1], and selecting the actual path for SFC traffic to traverse between the source and destination nodes. The challenge of deploying VNF components for a service request in an SEC-enabled network with limited resources to meet the predefined requirements of mission-critical applications is a significant research question. Addressing this challenge involves designing an appropriate *VNF mapping* and *scheduling* strategy [60]. The *VNF mapping* identifies optimal nodes for executing VNF components and the sets of physical links connecting these nodes, while *VNF scheduling* selects the most suitable time slots to process each VNF component on designated nodes.

Extensive research on the SEC [10, 13, 15, 21, 22, 72] has primarily focused on computational offloading strategies. However, optimal offloading performance relies on precise service provisioning, emphasizing the need for effective VNF mapping and scheduling strategies [50, 60, 73–78]. Furthermore, except [60], the existing works emphasize VNF mapping, neglecting VNF scheduling, potentially leading to request rejections for violating service requirements, especially with multiple VNFs mapped to an SEC node. Current solutions use static approaches, causing inefficiencies and service blockage in dynamic traffic. Dynamic VNF mapping and scheduling improve resource use for better request acceptance, but satellite mobility hampers system performance in NGSO systems. Certain studies [74, 76, 78, 79] tackle the issue by discretizing the topology into quasi-static snapshots. They focus on link availability at a specific snapshot but overlook crucial details such as data rate and delay in link states while employing VNF mapping strategies. This approach may lead to performance degradation due to potential changes in link states between snapshots. To manage service requests spanning across multiple snapshots, some work [75] applies VNF re-mapping at the onset of each snapshot. However, this method encounters challenges in the resource-limited SEC environment due to significant energy and computational costs [74, 80] associated with frequent reconfiguration.

Fairness is a crucial metric for fairly allocating resources like virtual machines, CPUs, and bandwidth in CC systems [81, 82]. Although fairness can encompass various dimensions, our emphasis lies on ensuring fairness in terms of the service delay margin. This choice is motivated by the unique demands of mission-critical applications, which demand stringent latency requirements to ensure timely and effective responses to critical situations [83, 84].

[1] On-board or in-orbit computing is a future data center in orbit, where satellites act as small data centers and offer cloud-like services in-space.

Maintaining fairness of E2E service delays among requests can ensure service sensitivity and improve overall system QoS by treating time-critical and time-tolerant services differently. Furthermore, mission-critical applications benefit from fairness by treating requests for services with the same delay requirements equally to ensure the QoS of the system. Existing research on SEC and VNF mapping has often focused on computational offloading, network integration, and dynamic resource allocation. However, there is a gap in integrating mission-critical requirements and fairness considerations into VNF mapping and scheduling in SEC-enabled networks.

2.1.1 Contributions

This chapter investigates VNF mapping and scheduling strategies in the context of mission-critical applications, to maximize E2E service delay margin[2] fairness among competing service requests. Through this work, we aim to enhance the reliability and efficiency of mission-critical operations in SEC-enabled networks by incorporating both link availability and link state information. Focusing on dynamic network architecture, our study facilitates interactions between ground-based users and satellites. This architecture integrates *satellite-to-satellite* collaborations and *edge-to-satellite* interactions for efficient service provisioning. Mathematically, we formulate a dynamic and topology-aware VNF mapping and scheduling as an `INLP` problem to maximize the fairness of the service delay margin among competing services, with specified *minimax* fairness. We initially address the problem using an optimization-based solution formulated with the SCA method; however, due to its high time complexity, we subsequently propose two computationally efficient algorithms: `FAGD_MASC` and `FASD_MASC`. Both `FAGD_MASC` and `FASD_MASC` perform similarly and effectively under low service arrival rates. However, their design objectives target distinct operating conditions. `FAGD_MASC`, based on a greedy approach, achieves faster convergence and is tailored for scenarios with low arrival rates and stringent responsiveness requirements. In contrast, `FASD_MASC` employs an iterative and exhaustive search process, leading to higher fairness and service acceptance rates, making it more suitable for high arrival rate scenarios where maximizing resource allocation fairness and service acceptance is critical. The proposed solutions are evaluated through extensive simulations. The main contributions of this chapter are summarized

[2] An E2E service delay margin of a service is defined as the safe margin to reach the upper bound of the delay required by the request.

as follows.

- While existing works handle time-varying network topology by using sequential quasi-static topology snapshots, our approach goes further by conducting a comprehensive evaluation of link quality, considering both propagation delay and availability.

- We propose a novel integration of mission-critical requirements and fairness constraints into dynamic VNF mapping and scheduling strategies within SEC-enabled networks. Our approach addresses the challenges posed by time-varying traffic workloads and changing network topologies, striving to achieve E2E service delay margin fairness. We first model the E2E service delay taking into account the time-varying traffic requests and network topology challenges. We then introduce a dynamic VNF mapping and scheduling strategy in which the strategies are readjusted in response to traffic demand and network topology. Furthermore, we describe the network topology as a sequence of snapshots, where each snapshot shows a quasi-static network topology. We then formulate a dynamic and topology-aware VNF mapping and scheduling strategy to maximize the service delay margin fairness between competing services as an `INLP` problem with specified *minimax*.

- We solve the problem `INLP` using an optimization-based approach by applying the SCA method. Due to the time complexity of the optimization-based solution, we proposed two lower complex algorithms to solve the `INLP` problem: `FAGD_MASC` and `FASD_MASC`. We also evaluate the proposed algorithms compared to the optimization-based approach.

- We perform extensive simulations to evaluate the proposed solutions. The results show that the proposed strategies are close enough to the optimization-based solution and outperform the benchmark.

This chapter is organized into five sections. Section 2.2 provides a literature review of related works and state-of-the-art approaches. Section 2.3 presents the system model and problem formulation for fairness-aware dynamic VNF mapping and scheduling. The proposed solutions are described in Section 2.4. Section 2.5 presents the performance evaluation of the proposed solutions and approach. Finally, Section 2.6 concludes the chapter and outlines directions for future work.

2.2 Related Works

In this subsection, we review recent advances in SEC and VNF mapping and scheduling schemes developed for SEC-enabled networks.

2.2.1 Satellite Edge Computing

Research on computing services in space has become an increasingly hot topic in recent years [10, 13, 15, 21, 22, 72, 85]. To overcome the drawbacks of the bent pipe satellite network architecture, Denby *et al.* [13] proposed an orbital edge computing architecture that supports edge computing in nanosatellites for sensing, processing, and communication to maintain the required latency and energy. In [72], the *in-orbit computing as a service* model was presented to provide ubiquitous computing services to users. However, the above works [13, 72] focus on the placement of MEC servers on satellite nodes. Yuxuan *et al.* [21] proposed a space edge computing architecture to deploy software in orbit and coordinate with cloud data centers. Furthermore, the authors of [15] discussed a satellite-terrestrial collaboration model to offload computations. However, none of them addressed the service delivery schemes, unlike our work.

Integrating network virtualization technologies like SDN/NFV with SEC-enabled networks enhances network management flexibility. In [10], the authors proposed an SDN/NFV-enabled virtualized resource and a multilayer architecture for integrated satellite-terrestrial edge computing networks. Lei *et al.* [22] introduced *5GsatEC*, an SEC framework, enhancing coverage and reducing service delay through embedded hardware and microservices on satellites. In [15] and [85], the authors discussed cooperative computing in a Space-Air-Ground Integrated Network (SAGIN) architecture. Similarly, [67] studied integrating ultra-reliable, low-latency edge intelligence into the 6G cellular network for mission-critical service support. The existing literature focused on computational offloading and integration with terrestrial networks. However, in the absence of an appropriate service provisioning scheme, offloading performance cannot be guaranteed. In contrast, our work emphasizes VNF mapping and scheduling strategies for service provisioning in an SEC-enabled network for mission-critical applications.

2.2.2 VNF Mapping and Scheduling in SEC Environment

Wang *et al.* [73] demonstrated a reconfigurable SFC-based service provisioning framework in a SAGIN environment. The authors defined the VNF mapping problem as `INLP` to maximize service acceptance rates while minimizing computational resource consumption. The work primarily explores VNF mapping in static scenarios. In this chapter, our focus is on dynamic VNF mapping, enabling the readjustment of the VNF mapping strategy for existing service requests in an SEC-enabled network. In [60], the authors introduced a dynamic VNF mapping and scheduling scheme within a SAGIN architecture for Internet of Vehicles (IoV) applications, formulated as a Mixed-Integer Linear Programming (MILP) problem. Unlike our work, they overlooked the impact of the time-varying satellite network topology, potentially causing variations in QoS that may not meet the stringent requirements for services in mission-critical applications.

Several service provisioning schemes [50, 74–78], aimed to address challenges arising from the time-varying network topology inherent in NGSO constellations. Current methodologies often segment the NGSO topology into snapshots, with each snapshot portraying a quasi-static topology, primarily emphasizing link availability to distinguish between snapshots. However, knowing link availability alone is inadequate. Overlooking variations in link states (*e.g.*, bandwidth, propagation delay) across snapshots may lead to E2E service delay requirement violations. Some literature [50, 75] addresses the impact of time-varying satellite network topology on system performance during VNF mapping, particularly for services spanning multiple snapshots. They proposed triggering a VNF migration and reconfiguration scheme in response to topology changes. In [75], the authors introduced a dynamic service migration and reconfiguration scheme based on satellite and ground user mobility for ensuring service continuity and meeting QoS requirements. Excluding [60], all existing work [50, 74–78] on service provisioning in satellite networks manage the VNF mapping strategy independently, neglecting the VNF scheduling strategy. In contrast, our work concentrates on topology-aware dynamic VNF mapping and scheduling schemes, discretizing the topology into snapshots while considering both link state and availability with prior knowledge of temporal variations.

While existing work on SEC and VNF mapping largely addresses computational offloading, network convergence, and dynamic resource allocation, the integration of mission-critical demands and fairness into VNF mapping remains underexplored. While resource allocation is essential, maintaining fairness in service delay margins for mission-critical applications has been overlooked. Furthermore, existing solutions seldom address the time-varying topology of satellite networks, impacting the quality of service. Our work addresses these gaps by proposing dynamic, topology-aware optimizations for VNF mapping and scheduling. We aim to maximize fairness in service delay margins by considering satellite network dynamics and mission-critical requirements.

In summary, although there is some research on VNF mapping and scheduling in NTNs, more research is needed. Existing approaches in VNF mapping and scheduling often fall short in addressing the stringent requirements of mission-critical applications. Our work aims to fill this gap by integrating service delay margin fairness with the specific demands of mission-critical communication. Therefore, in this study, we focus mainly on the optimal selection of VNF mapping and scheduling strategies for dynamic traffic demands and topologies to meet the critical service requirements of mission-critical applications in an MEC-enabled satellite network.

2.3 System Model and Problem Formulation

This section presents the system model, encompassing the substrate network and service requests. Figure 2.1 depicts the general SDN/NFV-enabled SEC network architecture for an NGSO satellite constellation, as elaborated in Section 2.3.1. The NGSO architecture comprises MEC-enabled satellite nodes, capable of hosting and processing VNFs to fulfill services with critical requirements in orbit. This allows users to connect to SEC nodes via a wireless link and complete service requests in orbit. We assume two adjacent SEC nodes are connected via a wireless RF or FSO ISL. An SDN controller manages the VNF mapping and scheduling strategy, orchestrating VNF placement, network topology, and resource availability. The controller is assumed to be located on one of the SEC-enabled satellite nodes in the constellation. However, the specific placement of an SDN controller for VNF mapping and scheduling in a satellite network requires further research and is beyond the scope of this study.

We assume a service request comprises the service type, source, and destination addresses, detailed in Section 2.3.2. The sources and destinations of service requests are users without access to the terrestrial network. These users communicate directly with satellites via Very

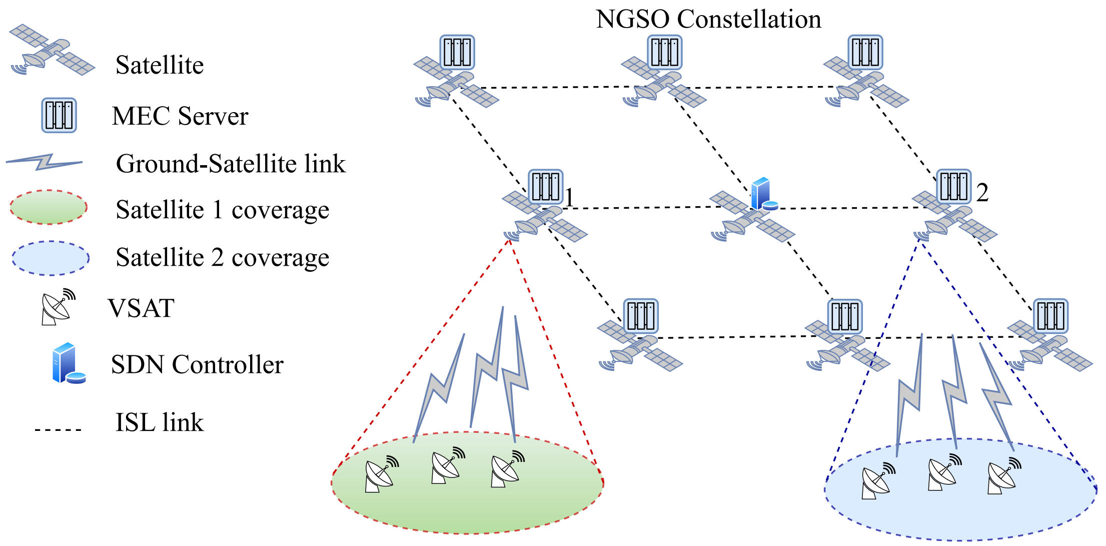


Figure 2.1: SDN/NFV-enabled SEC network architecture.

Small Aperture Terminals (VSATs) over wireless links (e.g., satellites 1 and 2 in Figure 2.1).

2.3.1 Substrate Network

We define the SEC-enabled network as an undirected graph $\mathcal{G} = (\mathbb{N}, \mathbb{E})$, where the set $\mathbb{N}$ contains physical nodes and $\mathbb{E}$ contains all possible links among the nodes. Additionally, $\mathbb{N}$ is subdivided into two subsets, namely, $\mathbb{S}$ and $\mathbb{U}$, representing the SEC nodes and end user terminals, respectively. Importantly, it holds that $\mathbb{N} = \mathbb{S} \cup \mathbb{U}$. Each SEC node $k \in \mathbb{S}$ is characterized by its processing capacity (*i.e., CPU*) k^{μ} and storage space (*i.e., buffer size*) k^{β} that can process VNFs and store the forwarding table for routing SFC traffic. End users act as both the source and destination devices for service requests, and it is assumed that they cannot process or store VNFs (*i.e.,* $k^{\beta} = 0$ and $k^{\mu} = 0$, $\forall k \in \mathbb{U}$). The link $e_{h,k} \in \mathbb{E}$ represents the connection between nodes $h \in \mathbb{N}$ and $k \in \mathbb{N}$. The satellite network topology in the NGSO constellation is time-varying due to satellite mobility. Consequently, link availability, delay, and rate vary, posing challenges for service provisioning. Fortunately, satellite mobility is predictable and periodic. To address this challenge, we subdivide the network topology into $\mathcal{T}$ sets of periodic quasi-static snapshots, represented as a set of time sequences of sub-topologies $\mathcal{G} = \{G_1, ..., G_t, ..., G_{\mathcal{T}}\}$, where $G_t \in \mathcal{G}$ denotes the t^{th} snapshot. The number of snapshots, $\mathcal{T}$, within a recurrence period relies on the satellite mobility rate. Notably, these snapshots can be pre-calculated from the prior knowledge of the satellite ephemeris and user locations. The quality and availability of each link are time-varying. Therefore, each link $e_{h,k} \in \mathbb{E}$ is characterized by its time-varying propagation delay $e^{D}_{h,k}(t)$ and data rate capacity $e^{\Omega}_{h,k}(t)$ at snapshot t. In addition, we define the availability of the link $e_{h,k} \in \mathbb{E}$ at the snapshot time t, $e^{\gamma}_{h,k}(t)$, as:

$$e^{\gamma}_{h,k}(t) = \begin{cases} 1, & \text{if link} \quad e_{h,k} \in \mathbb{E} \text{ is available at snapshot } t, \\ 0, & \text{Otherwise.} \end{cases} \tag{2.1}$$

Link availability in snapshots alone does not ensure QoS requirements due to variations in link states, impacting network performance and potentially causing incomplete or blocked service requests. To address this, we introduce a binary variable $e^{\rho}_{h,k}(t)$ to compare the propagation delay of the $e_{h,k} \in \mathbb{E}$ link between the snapshots $G_t \in \mathcal{G}$ and the subsequent snapshot $G_{t+1} \in \mathcal{G}$, defined as:

$$e^{\rho}_{h,k}(t) = \begin{cases} 1, & \text{if } e^{D}_{h,k}(t) \geq e^{D}_{h,k}(t+1), \\ 0, & \text{Otherwise.} \end{cases} \tag{2.2}$$

where, $e^{D}_{h,k}(t)$ and $e^{D}_{h,k}(t+1)$ represent the propagation delay of link $e_{h,k} \in \mathbb{E}$ in snapshot $G_t \in \mathcal{G}$ and subsequent snapshot $G_{t+1} \in \mathcal{G}$, respectively.

2.3.2 Service Request Model

We consider scenarios where terrestrial networks are unavailable; we rely exclusively on satellite networks as the primary infrastructure to provide mission-critical services. This approach guarantees reliable connectivity and QoS communication for remote users, meeting the stringent requirements of mission-critical services. We assume that service requests arrive randomly, each with unique requirements, service type, source, and destination address, following a Poisson distribution. Each service type comprises a sequentially ordered set of VNF components. Due to limited resources on SEC nodes, deploying all VNF types on a single node poses a significant challenge. Consequently, different VNF components are distributed across various nodes based on resource availability. However, this distribution of VNF deployment introduces additional propagation delays, potentially violating the service requirements. To address this challenge, careful VNF mapping is required.

We consider $\boldsymbol{S}_{sfc}(t)$ as a set of active mission critic service requests in the system (*i.e.,* newly arrived or currently in-process) at the snapshot t. As mentioned earlier, we assume that users generate and receive SFC request traffic. A service request comprises the source and destination addresses, representing the locations where the service originates and finishes. Additionally, it includes the service type, which defines the nature of the service request and its requirements. Each service request $s \in \boldsymbol{S}_{sfc}(t)$, is represented as $s = \langle s^{src}, s^{vnf}, s^{B}, s^{D}, s^{dst}\rangle$, where $s^{src} \in \mathbb{U}$, $s^{dst} \in \mathbb{U}$ are the source and destination user addresses, respectively. s^{vnf} is the ordered set of VNF components of service request s, with $f_{s,j} \in s^{vnf}$ representing the j^{th} VNF component of the service request. The notation $f_{s,j}^{cpu}$ and $f_{s,j}^{stor}$ denote the *computing* and *storage space* resources required to execute $f_{s,j}$. s^B and s^D are the minimum E2E bandwidth and maximum E2E delay tolerated by the service request s, respectively. These parameters ensure that mission-critical service requests exhibit diverse requirements in terms of bandwidth and delay, with specific services demanding high E2E bandwidth for rapid data transmission and others requiring stringent delay constraints for real-time communication.

The decision of mapping the VNF $f_{s,j}$ to the physical node k at snapshot t is represented by a binary variable, defined as:

$$x_k^{s,j}(t) = \begin{cases} 1, & f_{s,j} \text{ is mapped onto node } k \text{ at } t, \\ 0, & \text{Otherwise.} \end{cases} \tag{2.3}$$

Furthermore, we assume each SEC node can support specific VNF types due to the diverse resource requirements and operational characteristics of the VNFs. We describe a binary variable $f_{s,j}^{c,k}(t) \in \{0, 1\}$ to describe the VNF supporting node indicator as follows:

$$f_{s,j}^{c,k}(t) = \begin{cases} 1, & \text{if } f_{s,j} \text{ is supported by node } k \text{ at } t, \\ 0, & \text{Otherwise} \end{cases} \tag{2.4}$$

We consider that a virtual link between consecutive VNF components of a service request can be mapped onto a path with multiple physical links in the absence of direct connections between the nodes hosting the VNFs. Therefore, we define the decision to map the virtual link connecting VNFs $f_{s,j} \in s^{vnf}$ and $f_{s,j-1} \in s^{vnf}$ to the physical path that comprises link

$e_{h,k} \in \mathbb{E}$ in $G_t \in \mathcal{G}$ as:

$$y_{h,k}^{s,j}(t) = \begin{cases} 1, & \text{If virtual link between } f_{s,j} \text{ and} f_{s,j-1} \text{ is mapped to a physical path} \\ & \text{that includes the link } e_{h,k} \in \mathbb{E} \text{ at snapshot } t, \\ 0, & \text{Otherwise.} \end{cases} \quad (2.5)$$

We assume the sets of physical links $\mathcal{E}_{h,k}(t) \in \mathbb{E}$, which form the shortest physical path connecting the nodes $h \in \mathbb{N}$ and $k \in \mathbb{N}$. This path can be computed using a shortest path algorithm, such as the Dijkstra algorithm.

An efficient VNF scheduling scheme is necessary when the resources on a node are insufficient to process concurrent service requests. The decision of processing order for concurrent VNF components of competing service requests $s \in \boldsymbol{S}_{sfc}(t)$ and $g \in \boldsymbol{S}_{sfc}(t)$, represented by $f_{s,j} \in s^{vnf}$ and $f_{g,q} \in g^{vnf}$ respectively, at node k and snapshot t is defined as:

$$\chi_{sj,gq}^{k}(t) = \begin{cases} 1, & \text{if } f_{s,j} \text{ processes at } k \text{ after } f_{g,q} \text{ at t,} \\ 0, & \text{Otherwise.} \end{cases} \quad (2.6)$$

Service requests may start and end at a specific snapshot, t, or extend to the next snapshot, $t+1$. The service continuity indicator, $s^{\Upsilon}(t)$, indicates if service request s spans from the t^{th} snapshot to the subsequent $(t+1)^{th}$ snapshot and is defined as:

$$s^{\Upsilon}(t) = \begin{cases} 1, & \text{if } s \text{ starts at } t \text{ and continues in } t+1, \\ 0, & \text{Otherwise.} \end{cases} \quad (2.7)$$

2.3.3 E2E Service Delay and Service Delay Margin Model

In our work, we consider the E2E service delay to be composed of three fundamental components: 1) propagation delay, 2) processing delay, and 3) queuing delay. The propagation delay accounts for the cumulative propagation delay of the selected links to fulfill a given request. Similarly, the processing delay is the total delay required to execute the VNF components of the service request. Additionally, the queuing delay represents the overall delay for the VNF components of a given service request to wait before starting execution due to resource limitations for concurrent VNF processing. Please refer to Appendix 5.2 for a detailed explanation of delay components. Notably, our analysis excludes the deployment time required for the SDN/NFV orchestrator, as it falls beyond the scope of this study.

Within this context, the service delay comprises two primary components: the VNF mapping delay, encompassing processing and propagation delays, and the VNF scheduling delay, involving queuing delay. The VNF scheduling delay for a service request is determined by its processing delay and the arrival time of competing VNFs at the selected nodes for processing their VNF components. Thus, $f_{s,j}^{A,k}(t)$, denoting the arrival time of VNF $f_{s,j}$ at node k, is considered and its successive VNF $f_{s,j-1}$ is mapped to node h at snapshot t (*i.e.,* $x_h^{s,j-1}(t) = 1$). This is defined as:

$$f_{s,j}^{A,k}(t) = \left(f_{s,j-1}^{A,h}(t) + f_{s,j-1}^{\ell,h}\right)x_h^{s,j-1}(t) \;+ d_{h,k}^{prop}(t), \tag{2.8}$$

where $f_{s,j-1}^{A,h}(t)$ and $f_{s,j-1}^{\ell,h}$ represent the arrival time and processing delay of the preceding VNF component, $f_{s,j-1}$, in the service s at its designated node, h, respectively. $d_{h,k}^{prop}(t)$ represents the total propagation delay between the nodes $h \in \mathbb{N}$ and $k \in \mathbb{N}$.

We accordingly model the E2E service delay of a service request s at a given snapshot t, $d_s^{tot}(t)$ as the sum of the VNF mapping delay $d_s^{vm}(t)$ and VNF scheduling delay $d_s^{vs}(t)$.

$$d_s^{tot}(t) = d_s^{vm}(t) + d_s^{vs}(t), \tag{2.9}$$

See appendix A 5.2 for details of $d_s^{tot}(t)$, $d_s^{vm}(t)$, and $d_s^{vs}(t)$.

Remark 1. Propagation delay usually dominates the service delay, but when multiple VNFs are assigned to limited nodes, the queuing delay can become comparable.

Definition 1. *The service delay margin of a service request s is defined as the difference between its maximum tolerable delay s^D and its actual E2E service delay $d_s^{tot}(t)$ at a given snapshot t.*

$$\eta_{s,d}^{vnf}(t) = \frac{d_s^{tot}(t)}{s^D}, \tag{2.10}$$

Using Equation (2.9), we redefine (2.10) as the sum of service delay margins related to VNF mapping and scheduling as:

$$\eta_{s,d}^{vnf}(t) = \eta_{s,d}^{vm}(t) + \eta_{s,d}^{vs}(t), \tag{2.11}$$

where $\eta_{s,d}^{vm}(t)$ and $\eta_{s,d}^{vs}(t)$ are the service delay margins related to VNF mapping and scheduling, calculated as $\eta^{vm}_{s,d}(t) = \frac{d^{vm}_s(t)}{s^D}$ and $\eta^{vs}_{s,d}(t) = \frac{d^{vs}_s(t)}{s^D}$. The set of service delay margins for competing services is denoted as $\boldsymbol{\eta}^{vnf}_d(t) = \{\eta_1(t), \cdots, \eta_i(t), \cdots, \eta_{|\boldsymbol{S}_{sfc}(t)|}\}$. Using Equation (2.11), it can be further written as:

$$\boldsymbol{\eta}^{vnf}_d(t) = \boldsymbol{\eta}^{vm}_d(t) + \boldsymbol{\eta}^{vs}_d(t), \tag{2.12}$$

where $\boldsymbol{\eta}^{vm}_d(t)$ and $\boldsymbol{\eta}^{vs}_d(t)$ are sets of service delay margins for VNF mapping and scheduling, respectively.

2.3.4 Problem Formulation

We formulate the optimization problem to maximize fairness in the service delay margin among competing mission-critical service requests. Decision variables $x(t)$, $y(t)$, and $\chi(t)$, representing VNF node mapping, virtual link mapping, and VNF scheduling in snapshot t, are involved. The goal is to minimize $\eta^{vnf}_{s,d}(t)$, the service delay margin for the most adversely treated service request, s (*i.e.,* minimize the maximum), among competing services. This improves overall system performance and ensures a balanced distribution of service delay margins among the competing requests. The objective function can be defined as *Min-Max* *fairness* of the service delay margin, $\boldsymbol{\eta}_d^{vnf}(t)$ at snapshot t.

$$\underset{x(t),y(t),\chi(t)}{\text{minimize}}\,\{\max_{s}\ \boldsymbol{\eta}_d^{vnf}(t)\} \tag{2.13a}$$

$$\text{s.t.}\quad x_k^{s,j}(t) \in \{0,1\},\ \forall f_{s,j} \in s^{vnf},\ \forall s \in \boldsymbol{S}_{sfc}(t),\ \forall k \in \mathbb{N}, \tag{2.13b}$$

$$y_{h,k}^{s,j}(t) \in \{0,1\},\ \forall f_{s,j} \in s^{vnf},\ \forall s \in \boldsymbol{S}_{sfc}(t),\ \forall e_{h,k} \in \mathbb{E}, \tag{2.13c}$$

$$\chi_{sj,gq}^{k}(t) \in \{0,1\},\ \forall f_{s,j} \in s^{vnf}, f_{g,q} \in g^{vnf}, \forall k \in \mathbb{N},\ \forall s \in \boldsymbol{S}_{sfc}(t),\ \forall g \in \boldsymbol{S}_{sfc}(t), \tag{2.13d}$$

$$\sum_{k\in\mathbb{N}} x_k^{s,j}(t) = 1,\ \forall f_{s,j} \in s^{vnf},\ \forall s \in \boldsymbol{S}_{sfc}(t), \tag{2.13e}$$

$$x_k^{s,j}(t) \leq f_{s,j}^{c,k}(t),\quad \forall f_{s,j} \in s^{vnf},\ \forall k \in \mathbb{N},\ \forall s \in \boldsymbol{S}_{sfc}(t), \tag{2.13f}$$

$$\sum_{s\in\boldsymbol{S}_{sfc}(t)}\ \sum_{f_{s,j}\in s^{vnf}} x_k^{s,j}(t) f_{s,j}^{cpu} \leq k^{\mu},\ \forall k \in \mathbb{N}, \tag{2.13g}$$

$$\sum_{s\in\boldsymbol{S}_{sfc}(t)}\ \sum_{f_{s,j}\in s^{vnf}} x_k^{s,j}(t) f_{s,j}^{stor} \leq k^{\beta},\ \forall k \in \mathbb{N}, \tag{2.13h}$$

$$\sum_{e_{h,k}\in\mathbb{E}} y_{h,k}^{s,j}(t) e_{h,k}^{\gamma}(t) \geq 1,\ \forall f_{s,j} \in s^{vnf},\ \forall s \in \boldsymbol{S}_{sfc}(t), \tag{2.13i}$$

$$\sum_{s\in\boldsymbol{S}_{sfc}(t)}\ \sum_{f_{s,j}\in s^{vnf}} y_{h,k}^{s,j}(t) s^{B} e_{h,k}^{\gamma}(t) \leq e_{h,k}^{\Omega}(t),\ \forall e_{h,k} \in \mathbb{E}, \tag{2.13j}$$

$$f_{s,j}^{A,k}(t) x_k^{s,j}(t) \geq f_{s,j}^{A,k}(t) x_h^{s,j-1}(t),\ \forall x_h^{s,j-1}(t) = 1,\ \forall f_{s,j} \in s^{vnf},\ \forall s \in \boldsymbol{S}_{sfc}(t), \tag{2.13k}$$

$$y_{h,k}^{s,j}(t) \leq e_{h,k}^{\gamma}(t)\Big(1 - s^{\Upsilon}(t) + e_{h,k}^{\gamma}(t+1) e_{h,k}^{\rho}(t)\Big),\ \forall e_{h,k} \in \mathbb{E},\ \forall s \in \boldsymbol{S}_{sfc}(t), \tag{2.13l}$$

$$\chi_{sj,gq}^{k}(t) + \chi_{sj,gq}^{k}(t) = 1,\ \forall \{g \neq s\ \ OR\ q \neq j, k \in \mathbb{N}\},\ \forall s \in \boldsymbol{S}_{sfc}(t),\ \forall g \in \boldsymbol{S}_{sfc}(t), \tag{2.13m}$$

$$y_{m,n}^{s,j}(t) \geq x_h^{s,j-1}(t) + x_k^{s,j}(t) - 1,\ \forall e_{m,n} \in \mathcal{E}_{h,k}(t),\ k \in \mathbb{N},\ h \in \mathbb{N},\ \forall s \in \boldsymbol{S}_{sfc}(t), \tag{2.13n}$$

$$y_{m,n}^{s,j}(t) \leq \sum_{k\in\mathbb{N}} (x_h^{s,j-1}(t) + x_k^{s,j}(t)) - 1, \forall e_{m,n} \in \mathcal{E}_{h,k}(t),\ h \in \mathbb{N},\ \forall s \in \boldsymbol{S}_{sfc}(t), \tag{2.13o}$$

$$y_{m,n}^{s,j}(t) = 0, \forall e_{m,n} \notin \mathcal{E}_{h,k}(t), x_h^{s,j-1}(t) = 1, x_k^{s,j}(t) = 1,\ \forall f_{s,j} \in s^{vnf},\ \forall s \in \boldsymbol{S}_{sfc}(t), \tag{2.13p}$$

Constraints (2.13b), (2.13c), and (2.13d) indicate binary decision variables. Constraint (2.13e) ensures that a VNF component of a service request is processed by only one node. Constraint (2.13f) specifies that a VNF can only be mapped to a node that has the ability to execute it. Constraints (2.13g) and (2.13h) assert that a VNF can only be mapped to a node with sufficient storage and processing capacity, respectively. Constraint (2.13i) conveys that every virtual link must be mapped to at least one physical link that exists in a given snapshot. Constraint (2.13j) maintains that a virtual link can only be mapped to links with sufficient bandwidth for the VNF to transmit. Constraint (2.13k) requires the sequential processing of VNF components within a service request, ensuring adherence to their specified order for completion. Constraint (2.13l) allows assigning a service from snapshot $G_t \in \mathcal{G}$ to $G_{t+1} \in \mathcal{G}$ to a link if two conditions are met: the link must exist in both snapshots, and its propagation delay in $G_{t+1} \in \mathcal{G}$ should not exceed that in $G_t \in \mathcal{G}$. Constraint (2.13m) guarantees that a node processes only one VNF at a time. In case of two simultaneous requests, it dictates processing one while the other awaits execution, maintaining a specific order. Constraints (2.13n)-(2.13p) ensure that the virtual link between consecutive VNF components in a service request is assigned to all physical links along the shortest path connecting their hosting nodes. The Constraint (2.13o) affirms the precise mapping of the virtual link, accommodating scenarios in which a physical link is part of multiple shortest paths or a single shortest path between SEC nodes. Constraint (2.13p) ensures that a virtual link cannot be mapped to a physical link that is not part of the shortest path between the nodes hosting two consecutive VNF components of a service request.

The optimization problem formulated in Equation (2.13) falls within the class of `INLP` problems. The nonlinearity arises from the product of decision variables $x(t)$ and $y(t)$ in Equation (4) from Appendix A 5.2. Furthermore, the node mapping, $x_k^{s,j}$, in Equations (2.8) and (4), depends on the preceding VNF mapping $x_h^{s,j-1}$. This complexity makes the optimization problem computationally expensive to solve. Consequently, we decompose the problem into two stages: the VNF mapping optimization problem ($\mathbf{P}1$) and the VNF scheduling optimization problem ($\mathbf{P}2$). This decomposition linearizes the problem, allowing for the analysis of problem (2.13) as two linear subproblems, $\mathbf{P}1$ and $\mathbf{P}2$, described in Equations (2.14) and (2.15), respectively. $\mathbf{P}1$ represents the VNF mapping optimization problem with decision variables $x(t)$ and $y(t)$ as defined below.

$$\begin{aligned} \mathbf{P}1: \quad & \underset{x(t),y(t)}{\text{minimize}} \{ \max_{s} \ \boldsymbol{\eta}_d^{vm}(t) \} \\ \text{s.t.}, (2.13\text{b}), (2.13\text{c}), (2.13\text{e}) - (2.13\text{l}), & (2.13\text{n}) - (2.13\text{p}), \end{aligned} \tag{2.14}$$

Furthermore, $\mathbf{P}2$ is the VNF scheduling optimization problem with the decision variable $\chi(t)$ as defined below.

$$\begin{aligned} \mathbf{P}2: \quad & \underset{\chi(t)}{\text{minimize}} \left\{ \max_{s} \ \boldsymbol{\eta}_d^{vs}(t) \right\} \\ & \text{s.t.}, (2.13\text{d}), (2.13\text{m}), \end{aligned} \tag{2.15}$$

We address the challenge of the sequential dependence of the variable $x(t)$ initially by solving $\mathbf{P}1$ (*i.e.,* $x^*(t), y^*(t)$) and then solving $\mathbf{P}2$ (*i.e.,* $\chi^*(t)$). In this case, the solution to problem

P1 is the input to problem **P**2. However, due to the binary variables of $x(t)$, $y(t)$, and $\chi(t)$; **P**1 and **P**2 are combinatorial problems. Inspired by [86], we apply a binary relaxation technique to solve the subproblems efficiently where the binary variables are relaxed to satisfy $0 \leq x(t) \leq 1$, $0 \leq y(t) \leq 1$, and $0 \leq \chi(t) \leq 1$. Moreover, it is essential to employ an efficient and suitable relaxation method to ensure that the relaxed formulations of **P**1 and **P**2 produce output variables $x^*(t), y^*(t), \chi^*(t)$ constrained to binary values of either 1 or 0, as stated in Equations (2.16) and (2.17). We transform the problems **P**1 and **P**2 into the CCP **P**3 and **P**4, respectively [87] by carefully selecting the penalty function and parameters so that can be solved directly using mathematical optimization solvers such as *CVX* [65] iteratively.

$$\begin{aligned} &\mathbf{P}3 \quad \underset{x(t),y(t)}{\text{minimize}} \{ \max_{s} [\boldsymbol{\eta}_d^{vm}(t) - (-\psi_{xy})] \} \\ &\text{s.t.}, 0 \leq x(t) \leq 1, 0 \leq y(t) \leq 1, (2.13e) - (2.13l), (2.13n) - (2.13p), \end{aligned} \tag{2.16}$$

$$\begin{aligned} &\mathbf{P}4 \quad \underset{\chi(t)}{\text{minimize}} \left\{ \max_{s} [\boldsymbol{\eta}_d^{vs}(t) - (-\psi_{\chi})] \right\} \\ &\text{s.t.}, 0 \leq \chi(t) \leq 1, (2.13m), \end{aligned} \tag{2.17}$$

Here, ψ_{xy} and ψ_{χ} are the penalty functions of the original VNF mapping and scheduling optimization problems, respectively. The details of the penalty functions and parameters for binary relaxation can be seen in Appendix B 3. The subproblems **P**3 and **P**4 are convex problems that can be effectively solved using standard optimization software tools. Furthermore, the problems **P**3 and **P**4 are classified as Difference of Convex (DC) programming problems. Fortunately, these problems can be effectively solved using an iterative algorithm based on the CCP [87]. The CCP algorithm provides a systematic approach to iteratively handle the convex and concave components of the objective function, leading to convergence to optimal solutions for the DC programming problems. Therefore, we apply the decomposition and relaxation methods to problem (2.13) and an optimization-based approach based on SCA to solve them iteratively. We solve **P**3 to obtain the optimal solution $x^*(t)$ and $y^*(t)$ and update x^i and y^i by using *CVX*, where x^i and y^i are the VNF node mapping and virtual link mapping solutions in the i^{th} iteration, respectively. Then we solve **P**4 for a given x^i and y^i to obtain the optimal solution $\chi^*(t)$ using *CVX* and update χ^i by $\chi^*(t)$, where χ^i is the VNF scheduling solution in the i^{th} iteration. This process repeats until the solution converges by updating $i = i + 1$ and finally, we obtain the optimal solution $x^*(t), y^*(t)$, and $\chi^*(t)$. However, the high complexity of the optimization-based solution makes it unsuitable for mission-critical applications. Typically, VNF mapping and scheduling problems are NP-hard, inspiring the common use of heuristic algorithms for resolution [76, 77]. Consequently, we present two less complex algorithms suited for different traffic arrival rates to address the `INLP` problem.

2.4 Fairness-aware dynamic VNF mapping and scheduling for time-varying satellite network

In this section, we introduce two heuristic algorithms, namely `FAGD_MASC` and `FASD_MASC`, to efficiently obtain near-optimal solutions for the `INLP` problem in low and high traffic demand scenarios, respectively. Initially, the algorithms configure VNF mapping and then initiate a VNF scheduling strategy based on service delay margin values. They dynamically adapt to changing traffic demands through VNF re-mapping and re-scheduling, triggered by new service requests and E2E service delay violations. Further details are provided in the next subsections.

2.4.1 Feasible VNF Mapping Strategy Generator Algorithm

Algorithm 1 initializes the VNF mapping strategy $\boldsymbol{\pi}_{x,y}^{vnf} = \{\boldsymbol{x}(t), \boldsymbol{y}(t)\}$ for the proposed solutions `FAGD_MASC` and `FASD_MASC` as described in subsections 2.4.2 and 2.4.3, respectively. Here, $\boldsymbol{x}(t)$ and $\boldsymbol{y}(t)$ are the set of nodes and virtual link mapping strategies, respectively, of competing services in the system $\boldsymbol{S}_{sfc}(t)$. Given a service request $s \in \boldsymbol{S}_{sfc}(t)$ as an input parameter in the current snapshot t, the algorithm returns the VNF mapping strategy $\boldsymbol{\pi}_{x,y}^{vnf}$.

First, the algorithm extracts the VNF component s^{vnf} of the service request s *(line 1)*. The next step is to identify and group the specific nodes that support the VNF. For example, $f_{s,j}^{\mathcal{N}}$ denotes the sets of all nodes that can support and process the VNF $f_{s,j}$ *(line 3)*. For each VNF component of the service request s, an SEC node is randomly selected from the corresponding VNF supporting nodes that satisfy the Constraints (2.13f), (2.13g) and (2.13h), *(line 4)*. The VNF mapping strategy is updated for each VNF component in the service request. For example, for VNF $f_{s,j} \in s^{vnf}$ on an SEC node $k \in f_{s,j}^{\mathcal{N}}$ in $G_t \in \mathcal{G}$, it is denoted as $x_k^{s,j}(t) = 1$ *(line 5)*. The algorithm selects the shortest path between nodes hosting consecutive VNF components in the service request, adhering to Constraints (2.13i) and (2.13j). Moreover, to account for services continuing from the current snapshot t to the next snapshot $t+1$ (*i.e.*, the snapshot in which the VNF mapping decision is made), the selected links must satisfy the constraint specified in Equation (2.13l). Therefore, the Dijkstra algorithm is used to determine the shortest path, adhering to Constraints (2.13n)-(2.13o) *(line 7)*. The algorithm updates the corresponding node and link of the VNF mapping strategies $x(t)$ and $y(t)$ *(lines 8 and 9)* and returns the updated $\boldsymbol{\pi}_{x,y}^{vnf}$ *(line 12)*.

Algorithm 1: Feasible VNF Mapping Strategy Generator Algorithm

```
Input: SFC request $s$ in snapshot $G_t \in \mathcal{G}$
Output: $\boldsymbol{\pi}_{x,y}^{vnf} \leftarrow \{\boldsymbol{x}(t), \boldsymbol{y}(t)\}$ VNF Mapping strategy
 1: Extract the VNF components $s^{vnf}$ of the service request $s$
 2: for each $f_{s,j} \in s^{vnf}$ do
 3:     $f_{s,j}^{\mathcal{N}} \leftarrow$ Sets of SEC nodes that can process the VNF, $f_{s,j}$
 4:     Select a node $k \in f_{s,j}^{\mathcal{N}}$ randomly satisfying Constraints (2.13g) and (2.13h)
 5:     $x_k^{s,j}(t) \leftarrow 1$ ▷ Update the VNF node mapping strategy
 6:     if $x_h^{s,j-1}(t) == 1$ and $k \in \mathcal{N}(f_{s,j-1})$ then
 7:         Find the shortest path between $h$ and $k$ satisfying Constraints (2.13i), (2.13j),
            (2.13l), (2.13n)- (2.13p) using Dijkstra's algorithm.
 8:         $y_{h,k}^{s,j}(t) \leftarrow 1$ ▷ Update the Virtual link mapping strategy
 9:         $\boldsymbol{\pi}_{x,y}^{vnf} \leftarrow \{x_k^{s,j}(t), y_{h,k}^{s,j}(t)\}$ ▷ Update the VNF mapping for $f_{s,j}$
10:     end if
11: end for
12: return $\boldsymbol{\pi}_{x,y}^{vnf}$
```

2.4.2 Fairness-Aware Greedy Algorithm for Dynamic VNF Mapping and Scheduling

In this subsection, we propose `FAGD_MASC` to solve the `INLP` problem, as outlined in Algorithm 2. The algorithm takes the set of competing mission-critical service requests $\boldsymbol{S}_{sfc}(t)$ in snapshot t as input and outputs a sub-optimal VNF mapping and scheduling strategy $\boldsymbol{\pi}_{x,y,\chi}^{vnf}(t)$. First, the VNF mapping strategy $\boldsymbol{\pi}_{x,y}^{vnf}(t)$ is initialized using Algorithm 1 for the sets of service requests $\boldsymbol{S}_{sfc}(t)$ *(line 1)*. Then, the corresponding VNF scheduling in the selected VNF mapping and scheduling strategy, $\boldsymbol{\pi}_{\chi}^{vnf}(t)$, is calculated *(line 2)*. Furthermore, the optimal VNF mapping and scheduling strategy $\boldsymbol{\pi}_{x,y,\chi}^{vnf}(t)$ is updated by the initial strategy $\{\boldsymbol{\pi}_{x,y}^{vnf}(t), \boldsymbol{\pi}_{\chi}^{vnf}(t)\}$ *(line 3)*. Similarly, the maximum service delay margin $\eta_{s,d}^{vnf,*}(t)$ is also updated from the optimum strategy obtained *(line 4)*.

For each service request $s \in \boldsymbol{S}_{sfc}(t)$, the algorithm first checks if there exists a path $path_s(t) \in \mathbb{E}$ with a set of physical links that meet the Constraints (2.13i), (2.13j), and

(2.13l) *(line 7)*. When there is no possible path to accommodate the service request s (*i.e.*, $path_s(t) = \emptyset$), the algorithm invokes the VNF re-mapping and re-scheduling strategy *(line 8)*. Otherwise, if there are feasible paths ($path_s(t) \neq \emptyset$), the algorithm calculates the VNF mapping for the service request s using Algorithm 1 *(line 10)*. The algorithm `FAGD_MASC` computes the VNF scheduling for each VNF component s^{vnf} based on the service delay margin $\eta(t)$ *(line 11)*. The algorithm then updates the VNF mapping and scheduling $\boldsymbol{\pi}^{vnf}_{x,y,\chi}(t)$ *(line 12)*. In addition, it updates the maximum service delay margin among service requests ($\eta^{vnf}_{s,d}(t) = \max_i \{\boldsymbol{\eta}^{vnf}_d(t)\}$) *(line 13)*. `FAGD_MASC` updates the optimal VNF mapping and scheduling, $\boldsymbol{\pi}^{vnf}_{x,y,\chi}(t)$ with the current, $\boldsymbol{\pi}(t)$ if and only if the current maximum service delay margin is less than or equal to that of the previous iteration (*i.e.*, $\eta^{vnf}_{s,d}(t) \leq \eta^{vnf,*}_{s,d}(t)$), where $\eta^{vnf,*}_{s,d}(t)$ is the maximum optimal service delay margin obtained in the previous *(lines 14-17)*.

The process continues until the stop condition is reached, determined by the *StopConditionNotMet()* function *(line 5)*. The iteration stops if: 1) the maximum number of iterations is reached, 2) the objective function value ($\eta^{vnf,*}_{s,d}(t)$) remains unchanged, or 3) no candidate path in $path_s(t)$ satisfies the constraints after VNF re-mapping and re-scheduling *(lines 5-19)*. Finally, the algorithm returns the optimal VNF mapping and scheduling strategy $\boldsymbol{\pi}^{vnf}_{x,y,\chi}(t)$ at snapshot time, t, updating the network resources *(lines 20-21)*.

2.4.3 Fairness-Aware SA-based Algorithm for Dynamic VNF Mapping and Scheduling

`FASD_MASC` algorithm is an SA-based approach [88] proposed to solve the `INLP` problem. SA is inspired by metallurgical annealing, where controlled heating and cooling create a crystalline structure with minimal errors. Our choice of SA is driven by its simplicity and maturity as a metaheuristic algorithm, with theoretical convergence proofs [88].

`FASD_MASC` algorithm generates sub-optimal VNF mapping and scheduling decisions based on SFC requests, available network resources, and network topology over the snapshots. Given the set of mission-critical service requests $\boldsymbol{S}_{sfc}(t)$ and the initial state of the VNF mapping and scheduling strategy state $\boldsymbol{\pi}^{vnf,0}_{x,y,\chi}(t)$ in snapshot $G_t \in \mathcal{G}$, it returns the optimum VNF mapping and scheduling strategy $\boldsymbol{\pi}^{vnf}_{x,y,\chi}(t)$ as the final state, considering the current state. Algorithm 3 outlines the steps of the proposed `FASD_MASC` algorithm. Furthermore, the SA parameters, including the initial temperature $Temp_0$, cooling rate r_{cool}, Markov chain length L, and acceptance probability p, are also input parameters for the `FASD_MASC` algo-

```
Algorithm 2: PROPOSED FAGD_MASC ALGORITHM
  Input: Sets of competing mission-critical service requests, $\boldsymbol{S}_{sfc}(t)$
  Output: $\boldsymbol{\pi}^{vnf,*}_{x,y,\chi}(t) \leftarrow \{x^*(t), y^*(t), \chi^*(t)\}$: ▷ Optimal VNF Mapping and Scheduling
   1: $\boldsymbol{\pi}^{vnf}_{x,y}(t)$ ▷ Initialize the VNF mapping strategy using Algorithm 1
   2: $\boldsymbol{\pi}^{vnf}_{\chi}(t)$ ▷ Compute the corresponding VNF scheduling on $\boldsymbol{\pi}^{vnf}_{x,y}(t)$
   3: $\boldsymbol{\pi}^{vnf}_{x,y,\chi}(t) \leftarrow \{\boldsymbol{\pi}^{vnf}_{x,y}(t), \boldsymbol{\pi}^{vnf}_{\chi}(t)\}$ ▷ Update the optimal VNF mapping and scheduling
   4: $\eta^{vnf,*}_{s,d}(t) \leftarrow \max_{s} \{\boldsymbol{\eta}^{vnf}_{d}(t)\}$
   5: while StopConditionNotMet() do
   6:   for each competing service request $s \in \boldsymbol{S}_{sfc}(t)$ do
   7:     if $path_s(t) == \emptyset$ then
   8:       Trigger VNF Re-mapping and Re-scheduling function
   9:     end if
  10:     $\boldsymbol{\pi}^{vnf}_{x,y}(t)$ ▷ Generate a VNF mapping using Algorithm 1
  11:     $\boldsymbol{\pi}^{vnf}_{\chi}(t)$ ▷ Compute the corresponding VNF scheduling from $\boldsymbol{\pi}^{vnf}_{x,y}(t)$
  12:     $\boldsymbol{\pi}^{vnf}_{x,y,\chi}(t) \leftarrow \{\boldsymbol{\pi}^{vnf}_{x,y}(t), \boldsymbol{\pi}^{vnf}_{\chi}(t)\}$
  13:     $\eta^{vnf}_{s,d}(t) \leftarrow \max_{s} \{\boldsymbol{\eta}^{vnf}_{d}(t)\}$
  14:     if $\eta^{vnf}_{s,d}(t) \leq \eta^{vnf,*}_{s,d}(t)$ then
  15:       $\boldsymbol{\pi}^{vnf,*}_{x,y,\chi}(t) \leftarrow \boldsymbol{\pi}^{vnf}_{x,y,\chi}(t)$
  16:       $\eta^{vnf,*}_{s,d}(t) \leftarrow \eta^{vnf}_{s,d}(t)$ ▷ Update the optimal service delay margin
  17:     end if
  18:   end for
  19: end while
  20: Update the available resources of the network
  21: return $\boldsymbol{\pi}^{vnf,*}_{x,y,\chi}(t)$
```

rithm. These parameters significantly impact the algorithm's performance, necessitating careful selection [88]. $Temp_0$ determines the speed of convergence to the global optimum: higher values lead to slower convergence, while lower values risk premature convergence to a local optimum. The cooling rate r_{cool} determines the rate of temperature reduction at each iteration until $Temp$ reaches zero, indicating the end of the iterations. Similarly, the Markov chain length L specifies the number of iterations before a temperature decreases. Furthermore, the acceptance probability p determines the acceptability of new solutions.

The FASD_MASC algorithm starts by initializing the current temperature $Temp$ to the initial temperature $Temp_0$ *(line 1)*. It uses Algorithm 1 to create the initial VNF mapping strategy. A demand-driven approach based on the service delay margin $\boldsymbol{\eta}^{vnf}_{d}(t)$ generates a VNF scheduling strategy for active service requests $\boldsymbol{S}_{sfc}(t)$, updating the initial VNF mapping and scheduling strategy $\boldsymbol{\pi}^{vnf,0}_{x,y,\chi}(t)$ at snapshot t. This strategy serves as the initial state of the process. Then, the current VNF mapping and scheduling strategy $\boldsymbol{\pi}^{vnf}_{x,y,\chi}(t)$

Algorithm 3: Proposed `FASD_MASC` Algorithm

```
Input: $\boldsymbol{S}_{sfc}(t), \boldsymbol{\pi}^{vnf,0}_{x,y,\chi}(t), Temp_0, r_{cool}, L, p$
Output: $\boldsymbol{\pi}^{vnf,*}_{x,y,\chi}(t)$    ▷ Final VNF Mapping and Scheduling strategy
$Temp \leftarrow Temp_0$    ▷ Initialize current temperature
$\boldsymbol{\pi} \leftarrow \boldsymbol{\pi}^{vnf,0}_{x,y,\chi}(t)$    ▷ Initiate the current VNF mapping strategy state
while Temp $> 0$ do
    while  L $> 0$ do
        $\boldsymbol{\pi}^{vnf,Next}_{x,y,\chi}(t) \leftarrow \mathit{GenerateNextStrategy}(\boldsymbol{\pi}^{vnf}_{x,y,\chi}(t))$    ▷ Next VNF mapping and scheduling
        if $\exp(\frac{fitness(\boldsymbol{\pi}^{vnf}_{x,y,\chi}(t))-fitness(\boldsymbol{\pi}^{vnf,Next}_{x,y,\chi}(t))}{\mathcal{B}*Temp}) >$ p then
            $\boldsymbol{\pi}^{vnf}_{x,y,\chi}(t) \leftarrow \boldsymbol{\pi}^{vnf,Next}_{x,y,\chi}(t)$    ▷ Update the current VNF mapping and scheduling
        end if
        $L \leftarrow L - 1$
    end while
    $Temp \leftarrow r_{cool} * Temp$
end while
Update the available resources of the network
return  $\boldsymbol{\pi}^{vnf}_{x,y,\chi}(t)^{vnf,*}_{x,y,\chi}(t) \leftarrow \boldsymbol{\pi}^{vnf}_{x,y,\chi}(t)$
```

Algorithm 4: GenerateNextStrategy

```
Input: $\boldsymbol{\pi}^{vnf}_{x,y,\chi}(t)$    ▷ Current VNF Mapping and Scheduling strategy
Output: $\boldsymbol{\pi}^{vnf,Next}_{x,y,\chi}(t)$    ▷ Next VNF Mapping and Scheduling strategy
Randomly select a service request $s$ from the set of active mission-critical service
    requests $\boldsymbol{S}_{sfc}(t)$ in the current state, $\boldsymbol{\pi}^{vnf}_{x,y,\chi}(t)$.
Randomly choose a VNF component $f_{s,j}$ associated with $s$.
Identify the source node $k$ to which $f_{s,j}$ is currently mapped ($x^{s,j}_{k} = 1$) in the current
    state, $\boldsymbol{\pi}^{vnf}_{x,y,\chi}(t)$.
Determine potential target nodes $f^{\mathcal{N}}_{s,j}$ for relocation, considering resource availability
    and constraints.
Randomly select a suitable target node $k$ from $f^{\mathcal{N}}_{s,j}$.
Perform the relocation operation: Update the VNF mapping, $\boldsymbol{x}^{Next}$, from the current
    state, $\boldsymbol{\pi}^{vnf}_{x,y,\chi}(t)$, by setting $x^{s,j}_{k}$ to 0 and $x^{s,j}_{h}$ to 1.
Adjust virtual link mapping, $\boldsymbol{y}^{Next}$ from $\boldsymbol{x}^{Next}$ to account for the relocation.
Adjust VNF scheduling, $\boldsymbol{\chi}^{Next}$ from $\boldsymbol{x}^{Next}$ and $\boldsymbol{y}^{Next}$
return  $\boldsymbol{\pi}^{vnf,Next}_{x,y,\chi}(t) \leftarrow \{\boldsymbol{x}^{Next}, \boldsymbol{y}^{Next}, \boldsymbol{\chi}^{Next}\}$
```

is set as the current state, initialized using the initial state $\boldsymbol{\pi}^{vnf,0}_{x,y,\chi}(t)$ *(line 2)*. The main goal of the `FASD_MASC` algorithm is to determine an optimal VNF mapping and scheduling strategy that improves service delay margin fairness among competing mission-critical service requests. The algorithm iterates a maximum of L times for each current temperature value $Temp$. Within each iteration, the algorithm generates the next state $\boldsymbol{\pi}^{vnf,Next}_{x,y,\chi}(t)$ using Algorithm 4 based on the current state $\boldsymbol{\pi}^{vnf}_{x,y,\chi}(t)$ *(line 5)*. `FASD_MASC` uses Algorithm 5 to calculate the fitness of a given VNF mapping and scheduling in a specific state. Fitness is determined based on the highest service delay margin $\eta^{vnf}_{s,d}(t)$ among active service requests, as indicated in *line 2* of Algorithm 5. Consequently, the algorithm evaluates the fitness of both the current and next states *(line 6)*. The next state replaces the current state if $\exp(\frac{fitness(\boldsymbol{\pi}^{vnf}_{x,y,\chi}(t))-fitness(\boldsymbol{\pi}^{vnf,Next}_{x,y,\chi}(t))}{\mathcal{B}T}) > p$, where $\mathcal{B}$ represents the Boltzmann constant, and this value is compared against the predetermined acceptance probability threshold p. Upon completing L iterations, the algorithm decreases the current temperature by the cooling factor r_{cool} *(line 11)*.

Algorithm 5: FITNESS COMPUTATION

```
Input: $\boldsymbol{\pi}^{vnf}_{x,y,\chi}(t)$        ▷ VNF Mapping and Scheduling strategy
Output: $fitness(\boldsymbol{\pi}^{vnf}_{x,y,\chi}(t))$        ▷ fitness for $\boldsymbol{\pi}^{vnf}_{x,y,\chi}(t)$
1: Compute the service delay margin $\boldsymbol{\eta}$ from $\boldsymbol{\pi}^{vnf}_{x,y,\chi}(t)$ for requests using Equation (2.12).
2: $\eta^{vnf}_{s,d}(t) \leftarrow \max_{s} \{\boldsymbol{\eta}^{vnf}_{d}(t)\}$        ▷ The highest service service delay margin
3: return $fitness(\boldsymbol{\pi}^{vnf}_{x,y,\chi}(t)) \leftarrow \eta^{vnf}_{s,d}(t)$
```

Algorithm 4 outlines the procedure for generating the next state $\boldsymbol{\pi}^{vnf,Next}_{x,y,\chi}(t)$ from the current state $\boldsymbol{\pi}^{vnf}_{x,y,\chi}(t)$. The algorithm starts by selecting a service request s from the set of active mission-critical service requests $\boldsymbol{S}_{sfc}(t)$ in the current state $\boldsymbol{\pi}^{vnf}_{x,y,\chi}(t)$ *(line 1)*. Next, it randomly selects a VNF component $f_{s,j} \in s^{vnf}$ of the chosen service s *(line 2)*. Then, it identifies the source node k to which $f_{s,j}$ is currently mapped (with $x^{s,j}_{k} = 1$) in the current state $\boldsymbol{\pi}^{vnf}_{x,y,\chi}(t)$ *(line 3)*. Furthermore, it identifies potential target nodes $f^{\mathcal{N}}_{s,j}$ for relocation, considering resource availability and constraints outlined in Constraints (2.13f), (2.13g), and (2.13h) *(line 4)*. Then, it randomly selects one of the candidate target nodes $k \in f^{\mathcal{N}}_{s,j}$ and performs the relocation or swapping operation by setting $x^{s,j}_{k} = 0$ and $x^{s,j}_{h} = 1$ *(line 6)*. The algorithm updates the VNF node mapping strategy, $\boldsymbol{x}^{Next}$, and then adjusts the virtual link mapping, $\boldsymbol{y}^{Next}$, by applying the Dijkstra shortest path selection algorithm from $\boldsymbol{x}^{Next}$ to account for the swap *(line 8)*. Similarly, the VNF scheduling, $\boldsymbol{\chi}^{Next}$, is adjusted by applying the demand-based scheduling scheme from $\boldsymbol{x}^{Next}$ and $\boldsymbol{y}^{Next}$ to account for the relocation *(line 9)*. The final next state is generated by updating the next state with $\boldsymbol{x}^{Next}$, $\boldsymbol{y}^{Next}$, and $\boldsymbol{\chi}^{Next}$ *(line 10)*.

2.5 Performance Evaluation

In this section, we compare our proposed algorithms with the benchmark algorithm `TS_MAPSCH` [61] and the optimization-based optimal `INLP` solution obtained using decomposition and relaxation methods discussed in Section 2.3.4. The benchmark algorithm completes VNF mapping and scheduling of service requests on a *service-by-service* basis, indicating that the mapping and scheduling of one service must be finished before proceeding to the next one. In `TS_MAPSCH`, VNF re-mapping and re-scheduling occur only when there are violations in the E2E service delay demand.

Table 2.1: Simulation parameters for VNF mapping and scheduling

Parameters	**Value**
Topology	SES O3b MEO constellation [74]
Number of SEC	20 [74]
Satellite Ground Station Locations	9 [74]
Number of snapshots	24 [74]
Service Arrival Rate	$[0.05 - 0.20]$ requests/ms [61]
VNF Processing delay	$[5 - 10]$ ms [61]
ISL propagation delay	$[20 - 40]$ ms [74]
Uplink/downlink propagation delay	$[29 - 75]$ ms [89]
Maximum tolerable E2E service delay	$\{200, 250, 300\}$ ms [83, 84]
Number of VNFs per an SFC request	$[3 - 5]$ [74]
Number of VNFs per an SEC node	$[1 - 5]$ [74]
Processing capacity of an SEC node	96 vCPUs [77]
Storage capacity of an SEC node	112 GB [77]
Processing units required by a VNF	$[2 - 4]$ vCPU [77]
Storage units required by a VNF	$[4 - 8]$ GB [77]
Capacity of physical ISL	1 Gbps [77]
Capacity of physical uplink/downlink	1 Gbps [77]
Minimum tolerable E2E service bandwidth	$[10 - 20]$ Mbps [77]

2.5.1 Simulation Settings

Our simulations utilize the MEO constellation as the network topology, consisting of 20 SEC nodes and 9 satellite ground station locations, where users can submit service requests. We use the System Toolkit (STK) [89] to generate network topologies and data in each snapshot. As a result, we consider 24 snapshots (*i.e.,* $M = 24$) for the periodic satellite network with a recurrence period of 6 hours. Hence, each snapshot has a duration of 15 minutes. Furthermore, we assume 10 mission-critical service types with specified VNF components, E2E delay, and bandwidth requirements. We assume that each mission-critical service request is provisioned on the SEC nodes, characterized by a service type, a continuity indicator indicating whether it spans multiple snapshots, and source-destination addresses. We randomly generate service requests with varied parameters to ensure both heterogeneity and dynamism in traffic demand. We modeled the inter-arrival time among service requests using the Poisson distribution (*i.e.,* $M/M/1$ queuing model), expressed as: $Tr_{pois}^{arr} = -\frac{\log(1-R)}{\lambda}$, where R and λ represent the probability of a service request arrival and the mean arrival rate, respectively [90]. We choose a service arrival rate $[0.05 - 0.20]$ requests/ms for mathematical simplicity [61], aligning with specified service delay demands in milliseconds [83,84]. Additionally, the uplink and downlink propagation delays are randomly generated within the specified range outlined in Table 2.1, showcasing how the delay varies based on the user's location on the ground. For `FASD_MASC`, SA parameters are set as: the initial temperature $Temp_0 = 100$, Markov chain length $L_0 = 200$, cooling rate $r = 0.97$, and acceptance probability $p = 0.85$ [91]. The remaining simulation parameters and their values are summarized in Table 2.1.

2.5.2 Running time comparison

In this subsection, we compare the proposed solutions with both the benchmark and optimization-based optimal `INLP` solutions in terms of their running time, considering the significance of this metric in network infrastructures with limited resources, such as SEC-enabled networks. Note that all algorithms run on DELL Precision 3551 laptop, Intel(R) Core(TM) RAM 32.0 GB i9-10885H CPU @ 2.40GHz. We assume the arrival rate $\lambda = 0.05$, and R is a random variable uniformly distributed over the interval $R \sim \mathcal{U}(0, 1)$.

As shown in Table 2.2, the proposed algorithms `FAGD_MASC` and `FASD_MASC` converge in approximately 0.01357 and 0.0876 seconds, respectively. In contrast, the optimization-based optimal `INLP` solution takes at least 60 seconds to converge. As a result, the optimization-based optimal solution for `INLP` is impractical to meet the demands of mission-critical applications due to its significant computational time requirements, which do not align with the urgent and time-sensitive decision-making needs of mission-critical applications. This motivated the proposal of FAGD_MASC and FASD_MASC algorithms to achieve reasonable results in shorter convergence times. Specifically, FAGD_MASC achieves faster convergence compared to the benchmark algorithm, while FASD_MASC shows a comparable convergence time.

Table 2.2: Running time comparison

Metrics	FAGD_MASC	FASD_MASC	TS_MAPSCH	INLP
Running Time	$0.01357 sec$	$0.0876 sec$	$0.0872\ sec$	$> 60 sec$
$\eta^{vnf}_{s,d}(t)(t) = \max_{s} \{\boldsymbol{\eta}^{vnf}_{d}(t)\}$	1.04791	1.04782	1.04791	1.04597

2.5.3 Performance difference between the proposed solution and optimization-based optimal INLP solution

Initially, we employ the optimization solver *CVX* to solve the optimization-based optimal solution for INLP and obtain the optimal solution. For each snapshot $(G_1, G_2, \ldots, G_{24})$, we conduct individual experiments, and the final results are obtained by calculating the average values across all snapshots. We evaluate the performance of the proposed algorithms in terms of their deviation from the optimization-based optimal INLP solution in three scenarios with varying arrival rates λ: *Low* ($\lambda = 0.05$), *Medium* ($\lambda = 0.1$), and *High* ($\lambda = 0.2$) traffic scales. We compare the proposed solutions against the optimization-based optimal INLP solution in terms of two metrics: service request acceptance rate and service delay margin fairness.

Definition 2. *Service Acceptance Rate*
We define this performance metric as the ratio of service requests that satisfy their E2E service delay requirement to the total number of service requests. We express a service request s acceptance function at snapshot t as $s^{accept}(t)$ *as:*

$$s^{accept}(t) = \begin{cases} 1, & if \quad \eta^{vnf}_{s,d}(t) \leq 1, \\ 0, & Otherwise. \end{cases} \tag{2.18}$$

where $\eta^{vnf}_{s,d}(t)$ *is the service delay margin of the service request* s*. Therefore, the acceptance rate of the service requests at snapshot time* t *is expressed as* [74]*:*

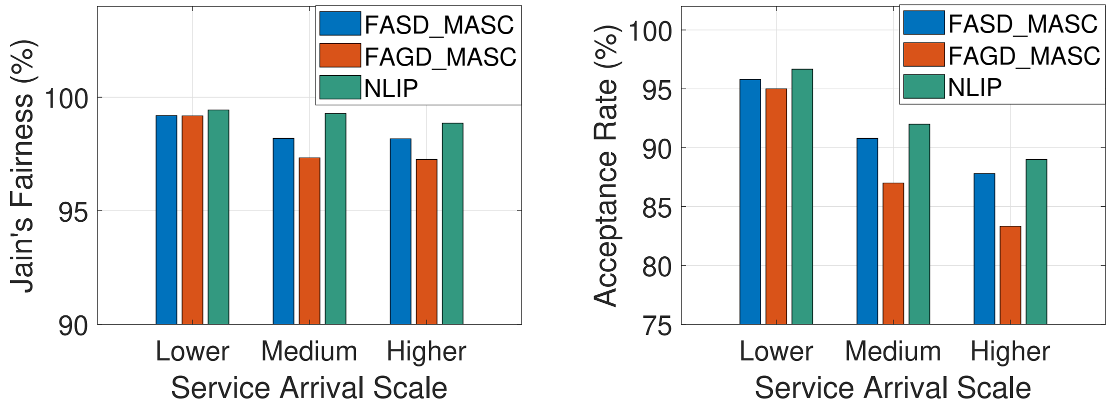
$$AR(t) = \frac{\sum\limits_{s \in \boldsymbol{S}_{sfc}(t)} s^{accept}(t)}{|\boldsymbol{S}_{sfc}(t)|}, \tag{2.19}$$

Definition 3. *The service delay margin fairness is defined using Jain's fairness index* $JF(\boldsymbol{\eta}_d^{vnf}(t))$ *[92]:*

$$JF(\boldsymbol{\eta}_d^{vnf}(t)) = \frac{\left(\sum\limits_{s\in\boldsymbol{S}_{sfc}(t)} \eta_{s,d}^{vnf}(t)\right)^2}{|\boldsymbol{S}_{sfc}(t)| \sum\limits_{s\in\boldsymbol{S}_{sfc}(t)} (\eta_{s,d}^{vnf}(t))^2}, \tag{2.20}$$

where $\boldsymbol{\eta}_d^{vnf}(t)$ *is the service delay margin matrix for* $\boldsymbol{S}_{sfc}(t)$ *at* t*.*

In Figure 2.2, the proposed algorithms exhibit a performance comparable to the optimization-based optimal INLP solution under low traffic arrival scale, with deviations of less than 0.70% and 0.07% in service acceptance rate and service delay margin fairness, respectively. However, unlike the FAGD_MASC algorithm, FASD_MASC demonstrates robustness even with high traffic rates, showing less than 1.1% and 0.6% deviation in service acceptance rate and service delay margin fairness compared to the optimization-based optimal INLP solution as shown in Figures (2.2b) and (2.2a), respectively.



(a) Service delay margin fairness.

(b) Service acceptance rate.

Figure 2.2: Performance comparison of the proposed solutions with the optimization-based optimal INLP solution

2.5.4 Performance comparison between the proposed solutions and the benchmark

In this subsection, we evaluate the performance of the proposed algorithms against the benchmark solution in terms of service request delay margin fairness, acceptance rate, and node resource utilization. We assess the results individually in each snapshot, and the final result is the average value in all snapshots.

Fairness for Varied Service Request Arrival Rates

This study aims to maximize fairness in service delay margins between competing mission-critical service requests. As shown in Figure 2.3a, both the proposed algorithms `FAGD_MASC` and `FASD_MASC` outperform the benchmark by more than 14.6% and 19.5% in fairness for the arrival rates of $\lambda = 0.05$ and $\lambda = 0.2$, respectively. The reason for this is that the `TS_MAPSCH` algorithm performs VNF mapping and scheduling to achieve maximum profit for a service provider. It is obvious that maximum profit could be achieved while some requests are discriminated against, resulting in poor service delay margin fairness. The proposed algorithms prioritize maintaining fairness in the service delay margins for all requests, resulting in better fairness performance compared to the benchmark algorithm, as evident in Figure 2.3a. However, fairness is expected to decrease with increasing arrival rates. This is due to the challenge of finding link and node mapping strategies to maintain fairness when faced with an influx of mission-critical service requests and limited network resources. However, the proposed algorithms consistently strive to maintain service delay margin fairness, unlike the benchmark solution, resulting in gradual and slight degradation.

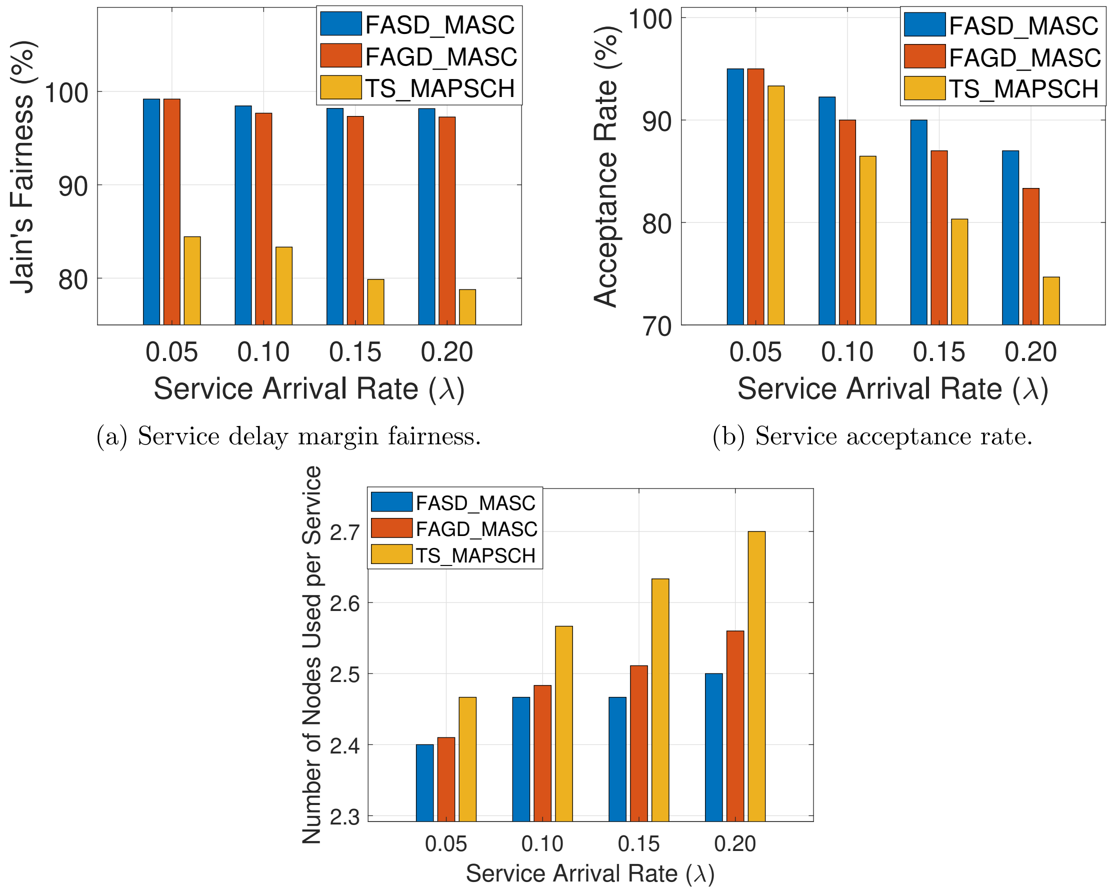
Acceptance rates for different service request arrival rates

We compare the performance of the proposed algorithms with the benchmark in terms of the acceptance rate of service requests at different arrival rates. As depicted in Figure 2.3b, the proposed solutions exhibit higher service acceptance rates compared to the benchmark. At a service rate of $\lambda = 0.05$, there is a gap of approximately 2.0% between the proposed algorithms and the benchmark. This gap widens further at higher service rates (*e.g.*, at $\lambda = 0.2$, the gap exceeds 12.6%). This improvement comes from two main factors: i) Unlike `TS_MAPSCH`, `FAGD_MASC` and `FASD_MASC` employ a fine-grained VNF mapping and scheduling strategy at the VNF level, enhancing resource utilization efficiency to accommodate more service requests. ii) The topology-aware VNF mapping and scheduling strategies of the proposed algorithms result in accepting more service requests, considering factors such as the continuity indicator and link availability (as described in Equation (2.13l) and elaborated in Section (2.5.4)).

Average Number of Nodes Used Per Service

The number of nodes used per service is defined as the total number of nodes required to process the VNF components to complete a corresponding service request. As shown in

Figure 2.3c, the proposed algorithms accept service requests with fewer nodes than `TS_MAPSCH`. Unlike the benchmark, the proposed algorithms apply the VNF mapping and scheduling strategy at the VNF level instead of the service level (*service-by-service* scheme), which provides a more fine-grained strategy to deploy service requests with fewer SEC nodes and network links. For example, as shown in Figure 2.3c, the number of nodes used to deliver service requests shows that the gap between the proposed algorithms and the benchmark reaches approximately 8.75% and 22.46% for $\lambda = 0.05$ and $\lambda = 0.20$, respectively. In other words, the `FASD_MASC` algorithm enables the provisioning of 100 service requests with at least 8-22 fewer nodes than the benchmark. Therefore, the proposed algorithms ensure better resource utilization than the benchmark, which consumes fewer nodes and links.



(a) Service delay margin fairness.

(b) Service acceptance rate.

(c) Average number of nodes used per service.

Figure 2.3: Performance evaluation with varying service arrival rate

Acceptance Rate with Varying Service Request Continuity

To assess the effect of dynamic satellite network topology on the acceptance rate of service requests spanning multiple snapshots, we compare the proposed algorithms with the benchmark. We initially analyze scenarios where no requests continue between snapshots (0% continuation rate), gradually increasing the continuation rate to 100% in each experiment. As illustrated in Figure 2.4, the service acceptance rate of the proposed algorithms experiences a slight decline with an increasing number of service requests continuing into the next snapshot. In contrast, the performance of the benchmark deteriorates notably as the proportion of continuing requests increases. For example, when no requests continue to the next snapshot (*i.e.,* 0% of available requests), the benchmark achieves an acceptance rate of around 90.9%. However, when all service requests continue to the next snapshot (*i.e.,* 100% of available requests), the service request acceptance rate of the benchmark drops to approximately 83%, signifying a degradation of nearly 7.9% due to the time-varying topology. This degradation is expected to be more pronounced in highly dynamic network topologies like LEO constellations. In contrast, the proposed algorithms show only a slight degradation of less than 0.81% because they consider both the state and availability of the link in their VNF mapping and scheduling strategies. Therefore, substantial improvements in the performance of the proposed algorithms over the benchmark can be attributed to their effective mitigation of time-varying topology challenges.

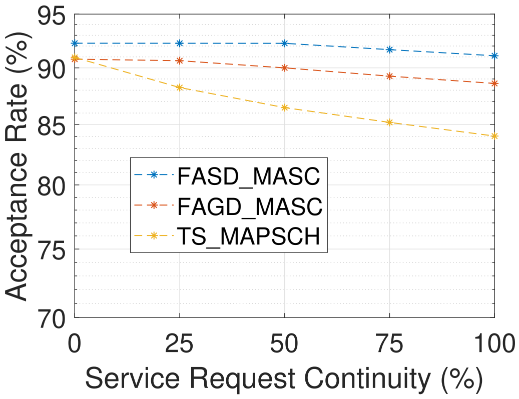


Figure 2.4: Service acceptance rate continuity over snapshots

2.5.5 Algorithm Complexity

Time Complexity Analysis for Algorithm 1

In this algorithm, the innermost function, the *Dijkstra shortest path algorithm*, is invoked $\mathcal{O}(n_{vnf})$ times, where n_{vnf} is the maximum number of VNF components per service request. The time complexity of the *Dijkstra shortest path algorithm* is $\mathcal{O}(|\mathbb{E}|+|\mathbb{N}|\log|\mathbb{N}|)$ in a graph $G = (\mathbb{N}, \mathbb{E})$ [93], where $|\mathbb{N}|$ and $|\mathbb{E}|$ are the number of nodes and links in the graph topology, respectively. Therefore, Algorithm 1 has a time complexity of $\mathcal{O}(n_{vnf} \cdot (|\mathbb{E}|+|\mathbb{N}|\log|\mathbb{N}|))$, depending on the number of VNF components in a service request and the size of the network.

Time Complexity Analysis for FAGD_MASC Algorithm

If we denote max_{itr} as the maximum number of iterations to find the optimum solution, the *Dijkstra shortest path algorithm* is invoked max_{itr} times for each service request. Therefore, the time complexity of Algorithm 2 can be defined as $\mathcal{O}(|\mathbb{N}|\cdot|\boldsymbol{S}_{sfc}(t)|\cdot n_{vnf} \cdot (|\mathbb{E}|+|\mathbb{N}|\log|\mathbb{N}|))$, which mainly depends on the size of the network ($|\mathbb{N}|$ and $|\mathbb{E}|$), the number of requests ($|\boldsymbol{S}_{sfc}(t)|$), and the number of VNF components per service request (n_{vnf}).

Time Complexity Analysis for FASD_MASC Algorithm

The time complexity of the FASD_MASC algorithm depends on the SA parameters, including the initial temperature $Temp_0$, cooling rate r_{cool}, and Markov chain length L. The time complexity of an SA algorithm can be defined by $L \log_{\frac{1}{r_{cool}}} Temp_0$ [88]. Therefore, the time complexity of FASD_MASC shown in Algorithm 3 is $\mathcal{O}(L \log_{\frac{1}{r_{cool}}} Temp_0 \cdot |\boldsymbol{S}_{sfc}(t)|\cdot n_{vnf} \cdot (|\mathbb{E}|+|\mathbb{N}|\log|\mathbb{N}|))$, which depends not only on the initial SA parameters but also on the sizes of the service requests and the network.

The TS_MAPSCH algorithm has a time complexity of $\mathcal{O}(|\boldsymbol{S}_{sfc}(t)|\cdot \log(|\boldsymbol{S}_{sfc}(t)|))$ for sorting service sets and $\mathcal{O}(|\mathbb{N}|\cdot n_{neigh} \cdot |\mathbb{N}|\cdot \log(|\mathbb{N}|) + |\boldsymbol{S}_{sfc}(t)|\log|\boldsymbol{S}_{sfc}(t)|)$ overall, where n_{neigh} is the number of neighbors and max_{itr} is the maximum allowed iterations.

The computational complexity of TS_MAPSCH is higher than that of FASD_MASC in terms of service requests $\mathcal{O}(|\boldsymbol{S}_{sfc}(t)|\cdot \log(|\boldsymbol{S}_{sfc}(t)|))$. However, FAGD_MASC shares the same complexity as TS_MAPSCH in this aspect. Moreover, both the proposed algorithms and the benchmark have similar complexities regarding network size, such as the number of SEC nodes. While FASD_MASC exhibits higher computational complexity than FAGD_MASC due to its initial parameters, it performs steadily in high service arrival scenarios. Thus, FASD_MASC is preferable for high service arrival rates, while FAGD_MASC is better suited for low arrival rates due to its lower convergence time and comparable performance.

Algorithm	Complexity
FAGD_MASC	$\mathcal{O}(\lvert\mathbb{N}\rvert\cdot\lvert\boldsymbol{S}_{sfc}(t)\rvert\cdot n_{vnf}\cdot(\lvert\mathbb{E}\rvert+\lvert\mathbb{N}\rvert\cdot log(\lvert\mathbb{N}\rvert)))$
FASD_MASC	$\mathcal{O}\Bigg(L\cdot\log_{\frac{1}{r_{cool}}}(Temp_0)\cdot\lvert\boldsymbol{S}_{sfc}(t)\rvert\cdot n_{vnf}\cdot(\lvert\mathbb{E}\rvert+\lvert\mathbb{N}\rvert\cdot log(\lvert\mathbb{N}\rvert))\Bigg)$
TS_MAPSCH	$\mathcal{O}(\lvert\mathbb{N}\rvert\cdot M\cdot\lvert\mathbb{N}\rvert\cdot log(\lvert\mathbb{N}\rvert)+\lvert\boldsymbol{S}_{sfc}(t)\rvert\cdot log(\lvert\boldsymbol{S}_{sfc}(t)\rvert))$ [61]

Table 2.3: Algorithm complexity comparison

In summary, the proposed FASD_MASC outperforms benchmarks in service acceptance rate and fairness, with improvements of 10% and 15%, respectively. This superior performance is due to two main factors: (1) It facilitates VNF re-mapping and re-scheduling, dynamically adjusting strategies to match traffic demand and infrastructure changes. (2) Its topological awareness, considering link state and availability, enhances performance. Besides its superior performance, the proposed solution's lower time complexity renders it well-suited for prompt service provisioning, especially for mission-critical services. From an end-user perspective, this improvement ensures consistent service delivery, reduced latency, and higher reliability. For service providers, optimized resource utilization and increased service acceptance rates enhance operational efficiency and support a larger user base without compromising quality.

2.6 Conclusion

In this chapter, we study dynamic VNF mapping and scheduling strategies in SDN/NFV-enabled satellite edge networks for mission-critical applications. We proposed two heuristic algorithms, a greedy-based and an SA-based approach, to address the VNF mapping and scheduling problem in a dynamic NGSO system. The algorithms facilitate VNF re-mapping and re-scheduling, dynamically adjusting strategies to match traffic demand and infrastructure changes. Their topological awareness, considering link state and availability, results in superior performance compared to existing solutions. In conclusion, our study presents a novel VNF mapping and scheduling strategy for mission-critical applications, advancing satellite edge computing for timely and reliable communication. The approach holds potential for applications in disaster response, remote medical care, and other time-sensitive operations.

Chapter 3

Content Caching in Multi-Layer Satellite Edge Networks

Satellite Edge Computing (SEC) integrated with multi-layer networks is a promising solution for on-board content caching to meet the stringent content delivery requirements in the future network. On-board content caching is challenging in satellite edge networks because of limited on-board resources, dynamic topology, and fluctuating traffic demand. Addressing the challenges of dynamic network topology and fluctuating traffic demands requires frequent cache updates. However, this introduces cache reconfiguration overhead, which overloads feeder links and increases cache update delays. To address these challenges, we propose an on-board dynamic cache reconfiguration strategy that maximizes the cache hit rate while minimizing reconfiguration overhead. We propose a proximity-based content popularity model and an AoI-aware caching strategy to optimize on-board satellite resources, enhancing the cache hit rate and ensuring content freshness. We propose a greedy algorithm for an initial and fast solution for caching contents and enhance it with a GA-based approach. Simulations with realistic datasets demonstrate that the proposed solution outperforms the Cooperative Content Caching (CCC) benchmark by 11.63% and 9.25% in terms of cache hit ratio and feeder link load, respectively.

3.1 Introduction

In recent years, there has been a surge in demand for mobile data traffic. According to the Ericsson forecast, mobile data traffic is expected to triple by 2030 [94], with video traffic comprising over 74% of the total. With the exponential growth of such data-hungry applications and increasing demand for ubiquitous connectivity, satellite networks are becoming integral to extending network services to remote and underserved areas. However, the inherent latency and limited bandwidth of satellite links remain significant challenges in traditional bent-pipe satellite architecture, especially for delay-sensitive or data-intensive applications.

To alleviate these challenges, on-board content caching is considered as a promising solution. On-board caching involves storing frequently requested or latency-sensitive content directly on satellites, which significantly reduces content retrieval latency and feeder link load, enhancing the overall Quality of Experience (QoE). In contrast to conventional caching strategies that rely on ground-based infrastructure, on-board caching leverages satellite storage resources, Inter-Satellite Links (ISLs), and global satellite network coverage to facilitate more efficient and ubiquitous content distribution. Satellite Edge Computing (SEC)which integrates computing and storage resources within satellite constellations [95] is considered a key enabler of on-board caching by providing satellite-based storage capabilities.

To achieve seamless global coverage and address the diverse performance requirements of next-generation services, modern satellite networks are evolving into multi-layer architectures comprising Low Earth Orbit (LEO), Medium Earth Orbit (MEO), and Geostationary Earth Orbit (GEO) satellites [96]. Integrating on-board caching across all layers unlocks its full potential by leveraging the distinct advantages of each orbital layer. SEC-enabled GEO satellites offer persistent wide-area coverage and are well-suited for long-term caching of globally popular or archival content. However, their long distance from the ground results in high content distribution and cache update delays, limiting their suitability for real-time content delivery. In contrast, with their limited coverage and low latency, LEO satellites are well-suited for caching locally popular and time-sensitive content. However, their high mobility results in frequent handovers, and their limited coverage constrains consistent and sustained content distribution. MEO satellites, orbiting between GEO and LEO, offer a balance of coverage, latency, and handover frequency, making them suitable for caching content moderately in terms of global and local popularity and delay sensitivity. This hierarchical caching strategy enhances scalability, content availability, and network robustness to support applications with diverse requirements.

Although on-board content caching in multi-layer satellite networks is a promising solution for next-generation applications, it also introduces new challenges due to the unique characteristics of satellite networks. These challenges include dynamic network coverage and topology, as well as limited on-board resources, which can impact the content availability and reliability of content distribution. Satellites have limited storage capacity, which restricts the number of contents that can be cached on-board. This limitation necessitates an efficient and effective caching strategy, making the design of such strategies challenging in order to meet diverse user requirements. Content popularity-based content caching is a commonly used content cache placement strategy that prioritizes caching frequently requested content to improve retrieval speed and reduce latency [97, 98]. By guiding which content is cached on each satellite node, the content popularity model directly influences storage allocation. Given the limited on-board capacity, leveraging an accurate and adaptive popularity model is vital to optimize storage utilization, enhance content availability across satellite nodes and ultimately improve overall system performance.

The inherent mobility of non-geostationary orbit (NGSO) satellites leads to a dynamic network topology, adding significant complexity to content caching strategies. Figure 3.1 illustrates the impact of satellite mobility on system performance. Satellite mobility in the LEO and MEO layers causes link connectivity to change dynamically over time, resulting in a time-varying network topology that directly affects content delivery paths and overall service reliability. For example, at time $\tau = 1$, satellite S_5 cached the requested content and successfully delivers it to the user through access satellite S_4 via an established path (shown by red arrows). At $\tau = 2$, the link between satellites S_1 and S_4 becomes unavailable due to relative mobility and changes in line of sight, requiring content access through satellite S_2 and resulting in increased delivery delay. At $\tau = 3$, the disappearance of the link between S_2 and S_5 renders the content inaccessible to the user, resulting in a cache miss unless the cache is updated accordingly. Beyond these link dynamics, satellite mobility also causes time-varying coverage areas that impact both user connectivity and the accuracy of content popularity estimation. Since the content popularity model relies on request patterns within a satellites current coverage area, retaining historical user request statistics without adaptation can lead to irrelevant caching and inefficient content placement. Consequently, this inefficiency may cause longer delivery delays or increased cache misses.

Furthermore, the dynamic and heterogeneous nature of ground service requests, combined with the rapid movement of NGSO satellites, complicates caching decisions in multi-layer satellite networks. To mitigate these issues, frequent and timely cache updates are necessary;

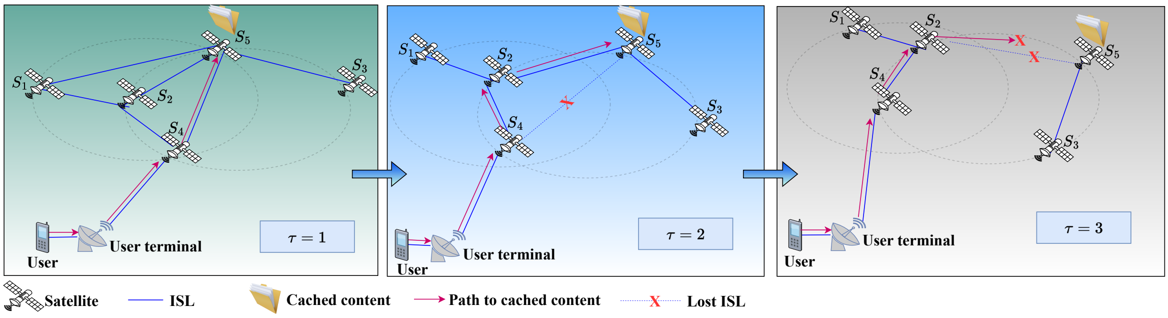


Figure 3.1: Time-varying satellite network topology.

however, this introduces cache reconfiguration overhead, which can overload feeder links and lead to increased update delays. A cache-to-cache update scheme can reduce this overhead by updating caches using existing on-board content rather than relying on ground-based sources. Nevertheless, repeatedly relying on on-board caches as content servers risks serving outdated or less relevant information, particularly for highly dynamic content such as social media feeds, weather updates, and road condition reports that require timely refreshes.

3.1.1 Contributions

In this work, we propose an on-board content caching with a dynamic cache reconfiguration (*OCCR*) scheme for multi-layer satellite networks consisting of SEC-enabled LEO, MEO, and GEO satellites that jointly optimizes cache hit rate and cache reconfiguration overhead. In the context of this work, the caching process involves two key decisions: *(1)* content cache placement, which identifies an optimal satellite node to cache a content item, and *(2)* content server selection, which determines an optimal node to update the designated cache. Cache reconfiguration overhead refers to the combined impact of feeder link load and content cache placement delay. Moreover, *OCCR* performs the cache placement by optimizing storage resources in the satellite constellation to maximize cache hit rate and ensure user requirements. Concurrently, *OCCR* proposes an on-board cache update scheme where satellites with cached content serve as content servers to minimize reconfiguration overhead. However, using on-board caches repeatedly as content servers can lead to outdated or less relevant information, especially for dynamic content such as social media feeds, weather updates, and road condition reports, which need frequent updates to remain relevant. Therefore, in this study, we utilize the AoI [99], a widely recognized metric that quantifies the time elapsed since the last content update. We model the AoI of on-board cached content to assess its freshness and relevance- an important aspect that existing studies on on-board content caching have not adequately addressed. The key contributions of this work are summarized as follows:

- We design a proximity-based content popularity model that optimizes content caching by leveraging nearby satellite resources to enhance cache hit rates.
- We present an on-board cache update scheme that allows cache nodes to serve as additional content servers alongside traditional cloud-based servers, thereby reducing reconfiguration overhead. Moreover, we introduce an AoI-aware on-board content caching scheme to monitor and ensure content freshness.
- We formulate a joint on-board content cache placement and server selection problem as an Integer Linear Programming (`ILP`), which aims to maximize the cache hit rate while minimizing reconfiguration overhead.
- We propose a greedy heuristic approach to obtain a fast initial solution to the `ILP` problem. Subsequently, we introduce a GA-based approach to enhance the cache hit rate and reduce the cache reconfiguration overhead.
- We perform extensive simulations to evaluate the effectiveness of OCCR. The simulation results show that the GA-based solution achieves a higher cache hit rate, lower feeder link load, and cache placement delay compared to benchmarks.

3.2 Related Works

In this subsection, we discuss the existing literature on content caching in satellite networks, content popularity-based cache placement, and cache reconfiguration. Table 3.1 summarizes the literature review.

3.2.1 Content Caching in Satellite Networks

Existing literature [113] primarily utilizes bent-pipe architecture satellites for content distribution. In addition, several existing works focus on content caching within a single orbit, particularly on GEO [105, 108] and LEO [102] satellites. Du et al. [105] utilized Deep Reinforcement Learning (DRL) for content caching in GEO-based 6G networks. In [108], the authors investigated resource allocation for content caching using a GEO satellite as a cloud server. The study in [110] focused on cache placement in LEO satellite networks.

Table 3.1: Literature review for on-board content caching

Related Works	**Satellite Types**			**Popularity**		**AoI**	**Reconfiguration Overhead**	**Time-Varying Topology**
	LEO	MEO	GEO	Local	Proximity-based			
Zhu et al. [95], Bao et al. [100]	✓	✓	X	✓	X	X	X	X
Rui et al. [101], Liu et al. [98], He et al. [102], Liang et al. [103], Hu et al. [104]	✓	X	X	✓	X	X	X	✓
Du et al. [105]	X	X	✓	✓	X	X	X	✓
Nguyen et al. [106], Han et al. [107], Gu et al. [108], Chen et al. [97]	✓	X	X	✓	X	X	X	X
Manyou et al. [109]	X	X	X	✓	X	✓	X	X
Jiang et al. [110]	✓	X	X	X	X	X	X	✓
Liu et al. [111]	✓	X	✓	✓	X	X	X	✓
Tang et al. [112]	✓	X	X	✓	X	X	X	✓
OCCR	✓	✓	✓	✓	✓	✓	✓	✓

Some studies also explored cooperative content caching among satellites [95, 100], [112]. However, these studies are limited to satellites within the same orbital constellation. In [95] and [112], the authors investigated cooperative content caching in NGSO satellites within a single-orbit constellation. Some works [101–104, 111], [112] incorporate dynamic network topology constraints into their cache placement strategies. The authors of [111] propose a spectral clustering scheme to partition the satellite network to manage its dynamic topology and the Artificial Bee Colony (ABC) algorithm for optimal cache placement. Rui *et al.* [101] proposed a content caching strategy for LEO satellite networks with time-varying topology. In [104], Hu *et al.* investigated a data-sharing strategy among NGSO satellites within a single orbital constellation, considering satellite movement constraints. The authors in [102] introduced a framework for learning-based content caching with dynamic topologies in STINs. Liang *et al.* [103] explored content cache placement and scheduling in LEO-based STINs with dynamic network topologies. To reduce content delivery latency, the authors of [112] explored content caching within the LEO satellite constellation by employing a game theory-based cooperative caching algorithm to optimize caching decisions across STINs while considering the dynamic satellite coverage area and user request preferences.

3.2.2 Content Popularity-based Cache Placement

Some recent works on content caching in satellite networks in [97,98] use content popularity functions to prioritize content items and select corresponding cache nodes. These functions dictate on-board resource utilization efficiency, ultimately determining overall system performance. Chen *et al.* [97] proposed an energy-aware cache placement strategy for LEO STINs by leveraging file popularity. In [98], the authors introduced a popularity-based cache replacement scheme to improve caching performance.

3.2.3 Cache Reconfiguration

Liu *et al.* [111] investigated collaborative cache placement with inter-satellite updates. The authors proposed transferring a copy of content items offloaded from a cloud-based content provider between on-board caches. Manyou et al. *et al.* [109] proposed a queue-aware cache content update scheduling algorithm to minimize the average AoI in terrestrial networks. They leveraged user request queues and employed a constrained Markov Decision Process (CMDP) and DRL to model the AoI of dynamic contents.

As depicted in Table 3.1, the existing literature does not address content caching in a multi-layer satellite network integrating GEO, MEO, and LEO satellites, nor on-board resource utilization through an appropriate content popularity design. In addition, prior studies overlook the impact of cache reconfiguration overhead and content freshness on system performance. To bridge this gap, in this work, we optimize the cache hit rate and reconfiguration overhead while ensuring content freshness. Unlike existing literature, this work focuses on efficient on-board resource utilization by designing an optimized content popularity function to maximize the cache hit rate. Additionally, it leverages on-board cached content for updates to minimize reconfiguration overhead.

3.3 System Model and Problem Formulation

As shown in Figure 3.2, we consider a multi-layer satellite network supporting on-board content caching and distribution. RF or FSO ISLs connect adjacent satellites. Remote users lack terrestrial network access and are covered by at least one satellite. They request content items via RF uplinks using VSATs as user terminals. Each satellite gateway is connected to specified satellites via FSO feeder links and to a cloud-based content provider server via

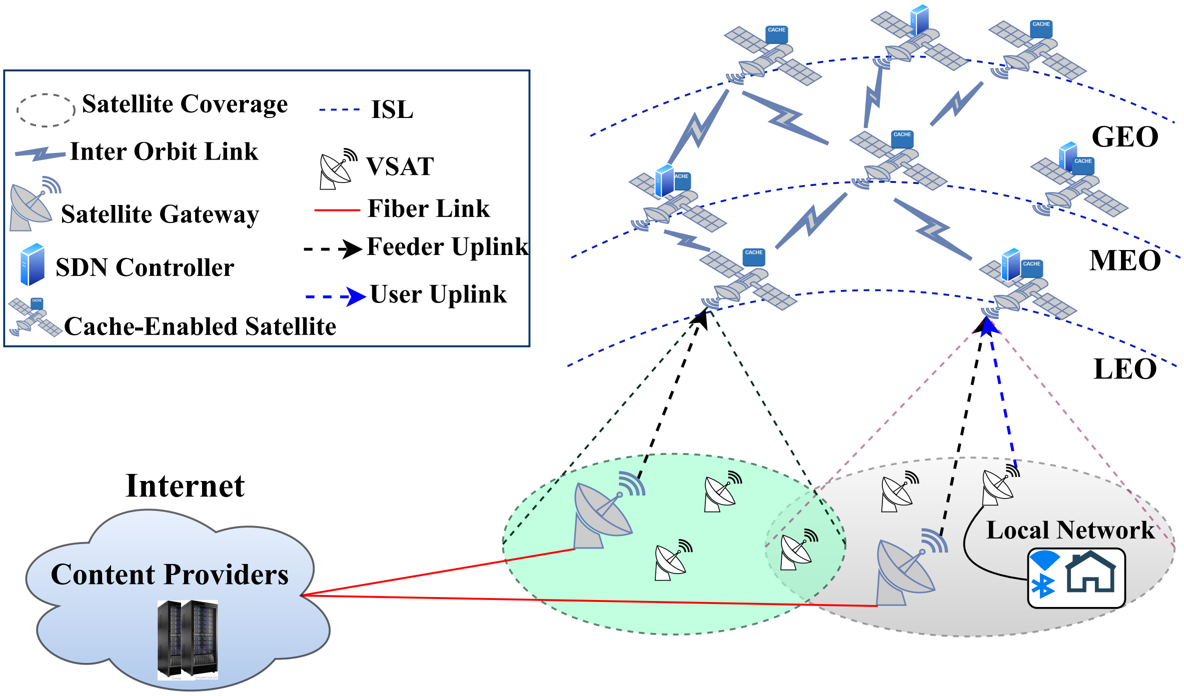


Figure 3.2: Multi-layer satellite edge network for on-board content caching

fiber cables. The content provider server initially stores all content items, each representing the most recent and up-to-date version. Based on content requests, copies of selected content items are offloaded from the content provider server through the feeder links and cached on designated satellites. To address the dynamic topology challenge, we segment the network into sequential ordered time slots, within which the topology remains static.

We consider a distributed set of SDN controllers to manage and orchestrate content cache placement and server selection. During each time slot, each satellite collects content request statistics within its coverage area and reports them to a designated SDN controller at the end of the time slot. The controllers then collaboratively aggregate data from their designated satellites and other SDN controllers to make content caching decisions at the start of the next time slot. Any standard SDN controller placement algorithm for satellite networks [114] can be used to determine the SDN controllerh locations in our use case. Satellites with up-to-date cached content serve as potential content servers for their respective items. Each satellite receives content requests from its coverage area and associates each request with the nearest node with the requested content.

3.3.1 Network Model

In this work, we define a set of sequentially ordered time slots as $\mathbb{T} = \langle 1, \ldots, t, \ldots, \mathcal{T} \rangle$, where each time slot has a duration of Δt time units and $\mathcal{T}$ represents the recurrence period. We model the multi-layer network at $t \in \mathbb{T}$ as an undirected graph $G_t = (\mathbb{N}, \mathbb{E})$, where $\mathbb{N}$ and $\mathbb{E}$ denotes the set of all nodes and links in the network at time t. $\mathbb{N}$ is further divided into subsets $\mathbb{G}$, $\mathbb{M}$, $\mathbb{L}$, $\mathbb{U}$, $\mathbb{J}$, and cps, representing the sets of SEC-enabled GEO, MEO, and LEO satellites, VSATs, gateways, and the content provider server, respectively. Thus, $\mathbb{N} = \mathbb{G} \cup \mathbb{M} \cup \mathbb{L} \cup \mathbb{U} \cup \mathbb{J} \cup cps$. We assume that the set of satellites, denoted $\mathbb{S} = \{\mathbb{G} \cup \mathbb{M} \cup \mathbb{L}\}$, where each satellite $k \in \mathbb{S}$ has a storage capacity k^{β} and a set of neighboring satellites $k^{\mathcal{N}(t)}$ at t. Each link $e_{h,k} \in \mathbb{E}$ between nodes $h \in \mathbb{N}$ and $k \in \mathbb{N}$ has a bandwidth capacity $e^{\Omega}_{h,k}(t)$ and a propagation delay $e^{D}_{h,k}(t)$ at t. Each user terminal $k \in \mathbb{U}$ is identified by its geographical location and has no caching capability $k^{\beta} = 0$.

We assume $\mathcal{A}$ as the global coverage area that includes all possible areas the satellite network can cover. $\mathcal{A}$ is represented as the set of M small regions $\mathcal{A} = \langle 1, \cdots, m, \cdots, M \rangle$. We define the substellar point of $k \in \mathbb{S}$ at t as $e_k(R_E, \theta_k(t), \phi_k(t))$, where R_E is the Earth's radius, $\theta_k(t)$ represents the complementary angle to the latitude of $k \in \mathbb{S}$, $\frac{\pi}{2} - \mathit{latitude}$, and $\phi_k(t)$ is the longitude of $k \in \mathbb{S}$. The collection of all points within the $k \in \mathbb{S}$ coverage area at t is defined as [50]:

$$\begin{aligned} \mathcal{A}_k(t) = \{e_k(R_E, \theta, \phi) | &\sin\theta \sin\theta_k(t) \cos(\phi - \phi_k(t)) \\ &+ \cos\theta \cos\theta_k(t) \leq \cos\Psi\}, \end{aligned} \tag{3.1}$$

where Ψ, θ, and ϕ represent the half-cone angle to the Earth's core, the polar angle corresponding to the latitude, and the azimuthal angle corresponding to longitude, respectively. We define $\mathfrak{R}^k_m(t) = 1$ if satellite $k \in \mathbb{S}$ covers region m at time t (*i.e.,* $m \in \mathcal{A}_k(t)$) and 0 otherwise. Given the periodic motion of the satellite, $\mathfrak{R}^k_m(t)$ is predetermined. We assume each region is relatively small compared to each satellite coverage area. User terminals are randomly distributed across the global coverage area. We define $\ell^m_k(t)$ as a location indicator for $k \in \mathbb{U}$ at t, which equals 1 if $k \in \mathbb{U}$ is located in region $m \in \mathcal{A}$ and 0 otherwise. Each user terminal is located in exactly one region at t, and this is expressed as $\sum\limits_{m \in \mathcal{A}} \ell^m_k(t) = 1$ where $k \in \mathbb{U}$. Two satellites $h \in \mathbb{S}$ and $k \in \mathbb{S}$ are visible to each other at t if both satellites are in

Line-of-Sight (LoS), denoted as $LoS_k^h(t)$ and defined as:

$$LoS_k^h(t) = \begin{cases} 1 & \text{if } D_{euc}^{h,k}(t) \leq D_{\max}^{h,k}(t) \\ 0 & \text{otherwise,} \end{cases} \tag{3.2}$$

where $D_{euc}^{h,k}(t)$ is the Euclidean distance between $h \in \mathbb{S}$ and $k \in \mathbb{S}$ at t, and $D_{\max}^{h,k}(t)$ is their LoS threshold [110]. In general, we model the visibility between nodes $h \in \mathbb{N}$ and $k \in \mathbb{N}$ using an adjacency matrix, which is a common approach in graph-theoretic representations of network topology. It is defined as:

$$\mathcal{V}_h^k(t) = \begin{cases} LoS_k^h(t) & \text{if } h \in \mathbb{S} \text{ and } k \in \mathbb{S} \\ \sum\limits_{m \in \mathcal{A}} \ell_k^m(t) \mathfrak{R}_k^h(t) & \text{if } h \in \mathbb{S} \text{ and } k \in \mathbb{U}, \end{cases} \tag{3.3}$$

We assume mutual visibility between nodes, i.e., $\mathcal{V}h^k(t) = \mathcal{V}k^h(t)$. Additionally, no direct connection exists between any two user terminals, i.e., $\mathcal{V}_k^h(t) = 0$ for $k, h \in \mathbb{U}$.

3.3.2 Content Caching And Content Server Selection Model

We consider a set of C content items originally stored at the cloud content provider cps to be cached on-board, denoted as $\mathbb{C} = \langle 1, \ldots, c, \ldots, C \rangle$. Each content item $c \in \mathbb{C}$ has a specific storage requirement c^β and packet size c^L. We define $c^{r_k}(t) = 1$ if there is at least one request for content item $c \in \mathbb{C}$ from user terminal $k \in \mathbb{U}$ at t and $c^{r_k}(t) = 0$ otherwise. We model the set of content popularity at satellite $k \in \mathbb{S}$ as $\boldsymbol{P}_k(t)$, where $p_k^c(t) \in \boldsymbol{P}_k(t)$ denotes the hierarchical proximity-based content popularity of $c \in \mathbb{C}$ at $k \in \mathbb{S}$ at t.

Local Content Popularity Model

The local content popularity of $c \in \mathbb{C}$ at satellite $k \in \mathbb{S}$ is measured based on requests for the content item c directly requested within the satellite coverage area, $\mathcal{A}_k(t)$, at t. We assume this popularity follows a Zipf-like distribution, defined as [115]:

$$p_k^{1,c}(t) = \frac{n^{-\delta}}{\sum\limits_{l \in \mathbb{C}} l^{-\delta}}, \tag{3.4}$$

where δ is the Zipf exponent that characterizes content preference and n is the rank of $c \in \mathbb{C}$ at $k \in \mathbb{S}$ at t. A high δ indicates that demand is concentrated on a few popular content items, while a low δ shows a more even distribution of requests across content items.

Proximity-Based Content Popularity Model

We model proximity-based content popularity $p_k^{2,c}(t)$ in t is the popularity of a content item $c \in \mathbb{C}$ requested in the coverage area of source satellite $h \in \mathbb{S}$ and is measured at a target satellite $k \in \mathbb{S}$, where both h and k belong to the same orbital layer. This popularity model helps to utilize the storage resources of k to cache c that is requested at $\mathcal{A}_h(t)$.

$$p_k^{2,c}(t) = \frac{1}{|\mathbb{S}|} \sum_{h \in \mathbb{S} | h \neq k} \frac{p_h^{1,c}(t)}{\left(H_k^h(t) + 1\right)^{\alpha}}, \tag{3.5}$$

where, $p_h^{1,c}(t)$ represents the local popularity of $c \in \mathbb{C}$ within $\mathcal{A}_h(t)$ at t. $H_k^h(t) > 0$ is a proximity factor, which quantifies the number of hops between the target satellite $k \in \mathbb{S}$ and the source satellite $h \in \mathbb{S}$. The exponent $\alpha \geq 0$ determines the rate at which proximity influences the propagation of content popularity. A higher α decreases the contribution of more distant source satellites, ensuring that content requests from satellites closer to the target satellite $k \in \mathbb{S}$ have a greater impact on proximity-based popularity estimation.

Hierarchical Proximity-Based Content Popularity

We calculate hierarchical proximity-based content popularity by considering the visibility between satellites across different layers. In the multi-layer satellite network, the content popularity of each item at a satellite in an orbital layer is the aggregated content popularity from its visible satellites in the lower layer. The hierarchical proximity-based popularity of $c \in \mathbb{C}$ at an LEO satellite $k \in \mathbb{L}$ at t is defined as:

$$\hat{p}_k^c(t) = \frac{p_k^{1,c}(t) + p_k^{2,c}(t)}{2}, \quad \forall k \in \mathbb{L} \tag{3.6}$$

where $p_k^{1,c}(t)$ and $p_k^{2,c}(t)$ represent the local and proximity based popularity of $c \in \mathbb{C}$ at $k \in \mathbb{S}$ at t, respectively. For an MEO satellite $k \in \mathbb{M}$ at time t, the hierarchical proximity-based popularity of $c \in \mathbb{C}$ is defined as the weighted sum of the hierarchical proximity-based popularity of $c \in \mathbb{C}$ at the set of LEO satellites $h \in \mathbb{L}$ visible to k at t ($i.e.$, $\mathcal{V}_h^k(t) = 1$).

$$\hat{p}_k^c(t) = \frac{\sum\limits_{h \in \mathbb{L}} \mathcal{V}_k^h(t) \hat{p}_h^c(t)}{\sum\limits_{h \in \mathbb{L}} \mathcal{V}_h^k(t)}, \quad \forall k \in \mathbb{M} \tag{3.7}$$

The hierarchical proximity-based popularity of c at a GEO satellite $k \in \mathbb{G}$ at t is defined as the weighted sum of the hierarchical proximity-based popularity of c at the set of visible MEO satellites $h \in \mathbb{M}$ to k at t ($i.e.$, $\mathcal{V}_h^k(t) = 1$).

$$\hat{p}_k^c(t) = \frac{\sum\limits_{h \in \mathbb{M}} \mathcal{V}_h^k(t) \hat{p}_h^c(t)}{\sum\limits_{h \in \mathbb{M}} \mathcal{V}_h^k(t)}, \quad \forall k \in \mathbb{G} \tag{3.8}$$

Remark 2. *From equations* (3.6), (3.7), *and* (3.8), *it is evident that global content popularity increases for the transition from lower-layer (LEO) to higher-layer (GEO). Consequently, the proposed content popularity model suggests that globally popular content should be prioritized for placement on higher-layer satellites (MEO or GEO). In contrast, locally popular content should precede lower-layer satellites.*

Finally, to ensure a valid probability distribution ($i.e.$, $\sum\limits_{c \in \mathbb{C}} p_k^c(t) = 1, \ \forall k \in \mathbb{S}$), we normalize the popularity derived from Equations (3.6), (3.7), and (3.8).

$$p_k^c(t) = \frac{\hat{p}_k^c(t)}{\sum\limits_{c' \in \mathbb{C}} \hat{p}_k^{c'}(t)}. \tag{3.9}$$

AoI Model For Cached Content

To ensure the freshness and relevance of cached content, we introduce an AoI-aware caching strategy. Specifically, for each content item $c \in \mathbb{C}$ cached at satellite node $k \in \mathbb{S}$, the parameter $a_k^c(t)$ quantifies its age at time slot t. This parameter enforces a freshness constraint, ensuring that only timely and relevant content is eligible to be used as a source for cache updates. By prioritizing fresher on-board content from neighboring satellites, the proposed approach reduces reliance on terrestrial cloud servers and mitigates feeder link load and content placement delay, thereby minimizing cache reconfiguration overhead. In communication systems, status update information is regularly transmitted from a source to a remote destination to track the status of the destination system or data [116]. These status updates are

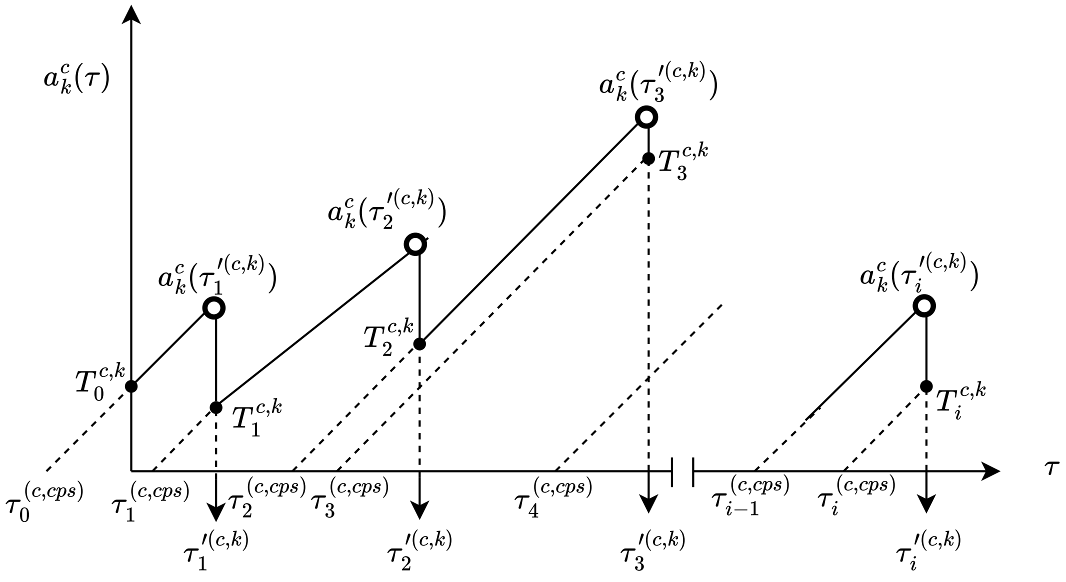


Figure 3.3: Representation of a sample AoI process of content, c cached at node k with timestamp, τ.

messages, encapsulated in packets, each containing information about one or more variables of interest along with the timestamp indicating when the sample was generated [117]. In this work, we consider the cloud-based content provider server, $cps \in \mathbb{N}$, as the source node for dynamic content items and a selected satellite node as the destination for each cached content item. To track and update the timeliness of cached content, status update are periodically transmitted from the cloud content provider server to the corresponding on-board cache node. Using content servers from the on-board cache at different time slots can compromise content freshness. We propose an AoI-based model to ensure the freshness of cached content. Inspired by [118], we model cached content status information updates as a communication system involving a cloud-based content provider, $cps \in \mathbb{N}$ and SEC-enabled satellites, $k \in \mathbb{S}$, as source and destination nodes, respectively, interconnected via a communication link that includes the terrestrial links, gateways, and feeder links. The content provider server generates and transmits status updates to designated on-board cache nodes to monitor the timeliness and freshness of cached content. This process is modeled as an $M/M/1$ queueing system, where status update follow a Poisson distribution with an arrival rate λ and are transmitted at an average service rate μ [118].

We consider a status update i for $c \in \mathbb{C}$ be generated at cloud server cps at time $\tau_i^{(c,cps)} \geq 0$

and received by $k \in \mathbb{S}$ at time ${\tau'_i}^{(c,k)} \geq \tau_i^{(c,cps)}$. In this chapter, we adopt the AoI model of the sawtooth function [118]. Figure 3.3 illustrates a sample AoI process for a content item cached on-board with varying timestamps. $\tau_i^{'(c,k)}$ denotes the arrival time at destination node k of the i^{th} update packet belonging to content c. Conversely, $\tau_i^{(c,cps)}$ represents the generation time of the i^{th} update packet for content c at the source node (content provider) cps. First, upon the arrival of the first update, (*i.e.,* $i = 1$) at the cache node, we initialize the AoI for c at k as the system delay, ${T_i}^{(c,k)}$ as:

$$a_k^c({\tau'_i}^{(c,k)}) = {T_i}^{(c,k)} = {\tau'_i}^{(c,k)} - {\tau_i}^{(c,cps)}, \tag{3.10}$$

In addition, during the interval $\tau \in [{\tau'_{i-1}}^{(c,k)}, {\tau'_i}^{(c,k)}]$, if no updates are received at the destination $k \in \mathbb{S}$, the AoI of $c \in \mathbb{C}$ increases linearly and is defined as:

$$a_k^c(\tau) = {T_{i-1}}^{(c,k)} + (\tau - {\tau'_{i-1}}^{(c,k)}) \tag{3.11}$$

Upon receiving an update for $i \geq 2$, the AoI resets to the system time, which is the delay in traversing the communication medium and is defined as:

$$a_k^c(\tau) = {T_i}^{(c,k)} = a_k^c({\tau'_i}^{(c,k)}) - ({\tau_i}^{(c,cps)} - {\tau_{i-1}}^{c,cps}) \tag{3.12}$$

Therefore, the AoI is updated after the arrival of the update as $a_k^c(\tau) = T_i^{(c,k)}$. In addition, we model the interarrival time $Tr_{pois}^{arr}(c,k)_i$ between the i^{th} and $(i-1)^{th}$ status updates of $c \in \mathbb{C}$ in $k \in \mathbb{S}$ employing an exponential distribution, which is characteristic of the Poisson arrival process in a $M/M/1$ queuing model [90] and defined as $Tr_{pois}^{arr}(c,k)_i = -\frac{\log(1-R)}{\lambda}$. Where λ is the mean arrival rate, and $R \sim U(0,1)$ is a uniformly distributed random variable.

From our on-board content cache placement strategy, placement is executed at the beginning of each time slot defined as $\tau = t\Delta t$. Therefore, the AoI of content $c \in \mathbb{C}$ cached $k \in \mathbb{S}$ at the beginning of the t^{th} time slot, $t\Delta t \in [{\tau'_{i-1}}^{(c,k)}, {\tau'_i}^{(c,k)})$, $i \geq 1$ is computed using Equation (3.11):

$$a_k^c(t) = {T_{i-1}}^{(c,k)} + (t\Delta t - {\tau'_{i-1}}^{(c,k)}), \tag{3.13}$$

The AoI increases linearly until the next update is received. If no status update is received at $\tau = t\Delta t$, the AoI follows the linear increase model. If a status update arrives at $\tau = t\Delta t$, the

AoI is updated using Equations (3.10) and (3.12). The following binary variable expresses whether $c \in \mathbb{C}$ cached at $k \in \mathbb{S}$ is up-to-date at t:

$$c^{\gamma,k}(t) = \begin{cases} 1, & \text{if } a_k^c(t) \leq a_{max}^c, \\ 0, & \text{otherwise.} \end{cases} \tag{3.14}$$

where a_{max}^c is the maximum allowable AoI threshold for $c \in \mathbb{C}$. A cached item is considered outdated if its AoI exceeds this threshold. We consider the cloud-based content provider *cps* stores all fresh content items (*i.e.*, $a_{cps}^c(t) = 0, \ \forall c \in \mathbb{C}$).

3.3.3 Problem Formulation

This section presents the formulation for on-board content caching with dynamic reconfiguration, focusing on content cache placement and server selection strategies. The formulation incorporates two key constraints: reconfiguration overhead and cached content freshness.

We define the decision variable $x_k^{'c}(t)$ as a binary variable that indicates whether $c \in \mathbb{C}$ is cached at $k \in \mathbb{S}$ at t

$$x_k^{'c}(t) = \begin{cases} 1, & \text{if content } c \text{ is cached at satellite } k \text{ at } t, \\ 0, & \text{Otherwise.} \end{cases} \tag{3.15}$$

We consider a content server h that can either be an on-board cache, where a copy of the cached content is used to update another cache without relying on the cloud-based content provider server, or a cloud-based content provider server, denoted as $h \in \{\mathbb{S} \cup cps\}$. The destination is a satellite $k \in \mathbb{S}$. Content cache placement involves migrating a selected $c \in \mathbb{C}$ from the chosen content server h to the designated cache k at t. We define a content server selection strategy, $y_{k,h}^{'c}(t)$, which determines whether $h \in \{\mathbb{S} \cup cps\}$ is selected as the content server for $c \in \mathbb{C}$ to cache at $k \in \mathbb{S}$ and expressed as:

$$y_{k,h}^{'c}(t) = \begin{cases} 1, & \text{if } h \in \mathbb{N} \text{ is selected to update on-board cache } k \in \mathbb{S} \text{ to cache } c \text{ at } t, \\ 0, & \text{Otherwise.} \end{cases} \tag{3.16}$$

Cache Hit Rate

The cache hit rate is defined as the probability of serving a request from an on-board cache. The hit rate of contents at a node $k \in \mathbb{S}$ at t is:

$$\eta_k^{cnt,H}(t) = \frac{1}{|\mathbb{C}|} \sum_{c \in \mathbb{C}} p_k^c(t)\ x_k^{\prime c}(t) \tag{3.17}$$

where $p_k^c(t)$ is the popularity of $c \in \mathbb{C}$ at $k \in \mathbb{S}$ at t.

Cache Reconfiguration Overhead

We model the cache reconfiguration overhead in terms of content placement delay and feeder link load.

- Content Placement Delay

  The content placement delay of $c \in \mathbb{C}$ from $h \in \{\mathbb{S} \cup cps\}$ to cache at $k \in \mathbb{S}$ at t is defined as:

$$d_{k,h}^{cnt,c}(t) = y_{k,h}^{\prime c}(t)\Bigg(\sum_{e_{h,k} \in \mathcal{E}_{h,k}(t)} e_{h,k}^D(t) + d_{k,h}^{txt,c}(t)\Bigg), \tag{3.18}$$

  where $\mathcal{E}_{h,k}(t)$ denotes the set of links that belong to the shortest path between $h \in \mathbb{N}$ and $k \in \mathbb{S}$ at t, $e_{h,k}^D(t)$ is the propagation delay of $e_{h,k} \in \mathbb{E}$, and $d_{k,h}^{txt,c}(t)$ signifies the transmission delay for c from h to k at t. The normalized placement delay Equation (3.18) is expressed as:

$$\eta_k^{cnt,D}(t) = \frac{1}{|\mathbb{C}|(|\mathbb{S}|+1)} \sum_{c \in \mathbb{C}} \sum_{h \in \{\mathbb{S} \cup cps\}} \frac{d_{k,h}^{cnt,c}(t) - d_{k,k}^{cnt,c}(t)}{d_{k,cps}^{cnt,c}(t) - d_{k,k}^{cnt,c}(t)}, \tag{3.19}$$

  $d_{k,cps}^{cnt,c}(t)$ represents the cache placement delay when cps is selected as the content server for caching c at k. Since the feeder link is a bottleneck for content cache placement, we focus on its capacity to model the transmission delay from cps to the selected satellites although there can be multiple links between gateways and cps. The placement delay for c from cps to k is given by:

$$d_{k,cps}^{cnt,c}(t) = y_{k,cps}^{\prime c}(t) \sum_{j \in \mathbb{J}} \zeta_k^j(t)(e_{cps,j}^D(t) + e_{j,k}^D(t) + d_{j,k}^{txt,c}(t)), \tag{3.20}$$

where $\mathbb{J}$ denotes the set of gateways associated with satellites. $\zeta_k^j(t)$ is a binary variable, equal to 1 if gateway $j \in \mathbb{J}$ is assigned to satellite $k \in \mathbb{S}$ at time slot t, and 0 otherwise. We assume that each satellite node is associated with exactly one gateway, i.e., $\sum\limits_{j \in \mathbb{J}} \zeta_k^j(t) = 1$. Additionally, $e_{cps,j}^D(t)$ and $e_{j,k}^D(t)$ represent the propagation delay of the link between the cloud server *cps* and gateway j, and the feeder link between gateway j and satellite node k, respectively, at time slot t. $d_{j,k}^{txt,c}(t)$ denotes the transmission delay for content item c over the feeder link $e_{j,k}$ at time slot t.

- Feeder Link Load

The feeder link load is the ratio of consumed bandwidth to its total capacity, comprising two main components:

1. Status Update Information Streaming

   This component represents the bandwidth allocated to transmit status updates to track the timeliness of cached content and ensure its relevance. We assume that update packets are generated exclusively by the content provider *cps* and are delivered through feeder links to satellites caching the corresponding content item. The average bandwidth allocation for streaming the status update of $c \in \mathbb{C}$ cached at $k \in \mathbb{S}$ via feeder link $e_{k,j} \in \mathbb{E}$ at t is given by:

$$\omega_{k,j}^{\mathbf{1},c}(t) = \zeta_k^j(t) \lim_{\Delta\tau \to 0} \frac{1}{\Delta\tau} \int_{t\Delta t}^{t\Delta t + \Delta\tau} \lambda^c c^L x_k^{'c}(t) dt, \tag{3.21}$$

   where, c^L and λ^c represent the packet size and arrival rate of $c \in \mathbb{C}$ at k, respectively.

2. Content Cache Placement

   This component quantifies the bandwidth of a feeder link utilized for transferring $c \in \mathbb{C}$ from *cps* to $k \in \mathbb{S}$ at t (*i.e.,* $x_k^{'c}(t) = 1,\ y_{k,cps}^{'c}(t) = 1$). The resulting traffic load on feeder link $e_{k,j} \in \mathbb{E}$ due to caching c at k is given by:

$$\omega_{k,j}^{\mathbf{2},c}(t) = \zeta_k^j(t) \frac{c^\beta}{d_{k,cps}^{cnt,c}(t)} y_{k,cps}^{'c}(t). \tag{3.22}$$

   The total normalized traffic load per content item on the feeder link associated with $k \in \mathbb{S}$ at t is defined as:

$$\eta_k^{cnt,L}(t) = \frac{1}{|\mathbb{C}|\;|\mathbb{J}|} \sum_{j\in\mathbb{J}} \zeta_k^j(t) \sum_{c\in\mathbb{C}} \frac{\omega_{k,j}^{\mathbf{1},c}(t) + \omega_{k,j}^{\mathbf{2},c}(t)}{e_{k,j}^{\Omega}(t)}. \tag{3.23}$$

where $|\mathbb{J}|$ and $|\mathbb{C}|$ denote the number of gateways and content items in the network, respectively. Moreover, $e_{k,j}^{\Omega}(t)$ represents the bandwidth capacity of the link $e_{k,j} \in \mathbb{E}$ at time t.

The on-board content caching with dynamic cache reconfiguration problem aims to determine the optimal content cache placement and content server selection strategy in each time slot that jointly maximizes cache hit rate and minimizes reconfiguration overhead while ensuring content freshness. $x^{'}(t)$ and $y^{'}(t)$ are the decision variables for content cache placement and content server selection, respectively, at t. The problem formulation is as follows:

$$\max_{x^{'}(t),y^{'}(t)} \sum_{k\in\mathbb{S}} \Bigg(w_1\eta_k^{cnt,H}(t) - (w_2\eta_k^{cnt,D}(t) + w_3\eta_k^{cnt,L}(t)) \Bigg), \tag{3.24a}$$

$$\text{s.t.} \quad y_{k,h}^{'c}(t) \le x_k^{'c}(t) c^{\gamma,h}(t) x_h^{'c}(t-1), \;\; \forall k \in \mathbb{S}, \forall h \in \mathbb{S} \cup cps, \;\forall c \in \mathbb{C}, \tag{3.24b}$$

$$y_{k,h}^{'c}(t) \le 1 - c^{\gamma,k}(t) x_k^c(t-1), \;\; , \;\; \forall k \in \mathbb{S}, \forall h \in \mathbb{S} \cup cps, \;\forall c \in \mathbb{C} \tag{3.24c}$$

$$\sum_{h\in\mathbb{S}\cup cps} y_{k,h}^{'c}(t) \le x_k^{'c}(t), \;\; \forall k \in \mathbb{S}, \;\forall c \in \mathbb{C} \tag{3.24d}$$

$$\sum_{c\in\mathbb{C}} \sum_{k\in\mathbb{S}} \sum_{h\in\mathbb{S}\cup cps} \frac{c^{\beta}}{d_{k,h}^{cnt,c}(t)} y_{k,h}^{'c}(t) \le e_{m,n}^{\Omega}(t), \;\forall e_{m,n} \in \mathcal{E}_{h,k}(t), \tag{3.24e}$$

$$\sum_{c\in\mathbb{C}} (\omega_{k,j}^{\mathbf{1},c}(t) + \omega_{k,j}^{\mathbf{2},c}(t)) \le e_{k,j}^{\Omega}(t), \;\forall k \in \mathbb{S}, \;\forall j \in \mathbb{J} \tag{3.24f}$$

$$\sum_{c\in\mathbb{C}} c^{\beta} x_k^{'c}(t) \le k^{\beta}, \;\; \forall k \in \mathbb{S}, \tag{3.24g}$$

$$\sum_{h\in\mathbb{S}\cup cps} y_{k,h}^{'c}(t) \ge (1 - x_k^{'c}(t-1) c^{\gamma,k}(t)) x_k^{'c}(t), \forall k \in \mathbb{S}, \forall c \in \mathbb{C}, \tag{3.24h}$$

$$x_k^{'c}(t) \le \sum_{u\in\mathbb{U}} c^{r_u}(t), \;\; \forall k \in \mathbb{S}, \;\forall c \in \mathbb{C}, \tag{3.24i}$$

$$\sum_{n\in k^{\mathcal{N}(t)}} x_n^{'c}(t) \le 1, \;\; \forall k \in \mathbb{S}, \;\forall c \in \mathbb{C}, \tag{3.24j}$$

$$c^{\gamma,cps}(t) x_{cps}^{'c}(t) \ge 1, \;\; \forall c \in , \forall t \in \mathbb{T}, \tag{3.24k}$$

$$\{x_k^{'c}(t), y_{k,h}^{'c}(t)\} \in \{0,1\}, \;\forall k \in \mathbb{S}, \;\forall h \in \mathbb{S} \cup cps, \;\forall c \in \mathbb{C}, \tag{3.24l}$$

where w_1, w_2, and w_3 represent weight factors for the cache hit rate, content placement delay, and feeder link load, respectively, with $w_1 + w_2 + w_3 = 1$. The weighting factors w_1, w_2, and w_3 are operator-defined parameters, empirically tuned to balance the trade-off between cache hit rate and cache reconfiguration overhead, in line with network policy and service priorities. Constraint (3.24b) ensures that cache reconfiguration is initiated only if: 1) an on-board cache node is selected to cache a corresponding content item in the current time slot. 2) The selected content server has cached the content item, and 3) the content item is still up-to-date in the current time slot t which ensures the freshness of the contents. Constraint (3.24c) prevents redundant cache placement of a content item that is already cached and up-to-date on a node, while constraint (3.24d) ensures that each content cache is updated by a single designated content server, thereby enhancing operational efficiency. Constraints (3.24f) and (3.24e) ensure that a feeder link and ISL do not exceed their bandwidth capacity, preventing traffic congestion, respectively. Constraint (3.24g) guarantees that the on-board cache remains within the storage limits of each satellite. Constraint (3.24h) ensures that a cache reconfiguration occurs if the following conditions are met: the on-board cache is selected to store a content item, and either the content item is outdated or the cache does not contain the specified content item. The parameter $a_k^c(t)$ quantifies the age of content item $k \in \mathbb{C}$ at satellite node $k \in \mathbb{S}$ and serves as an offline-computed input to enforce content freshness constraints, ensuring only timely content is eligible for cache updates. In such cases, a content server must be selected from the available content servers in the network. Constraint (3.24i) ensures that a content item cannot be cached unless it has been requested. Constraint (3.24j) ensures that only one type of content is cached on adjacent satellites, promoting the efficient use of ISLs and storage resources. Constraint (3.24k) ensures that the content provider maintains all possible content items, and these items remain fresh or up-to-date throughout all time slots, while Constraint (3.24l) enforces the binary nature of the decision variables.

The problem in Equation (3.24) is an `ILP` problem with binary decision variables $x^{'}(t)$ and $y^{'}(t)$. Since content cache placement problems are typically *NP-hard* [119], we propose a GA-based solution for its ability to explore complex solution spaces and fast and near-optimal solutions efficiently.

3.4 Proposed Dynamic Content Caching For Time-Varying Satellite Network

Greedy algorithms are widely used in satellite content caching due to their low time complexity and ease of implementation [103]. However, they often converge to local optima and are generally insufficient to solve large-scale problems. On the other hand, GAs are effective in solving large-scale problems and can provide near-optimal solutions [119] but are prone to slow convergence. To take advantage of the low complexity of greedy algorithms along with the large-scale optimization capabilities of GAs, we propose Greedy-enhanced Genetic algorithm for content Cache (*G-GenCache*), a two-stage heuristic algorithm to obtain suboptimal solutions to the ILP problem. In the first stage, a greedy-based approach generates a fast initial solution. In the second stage, a GA-based algorithm refines the solution, enhancing accuracy and overcoming local optima to achieve near-optimal performance. This allows the GA to start from a solution that is closer to the optimum, unlike the randomly initialized GA, 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 significantly accelerates the overall convergence of the GA by providing a more informed starting point.

3.4.1 Greedy Based Algorithm

We design a Greedy algorithm for Content Caching and Server Selection (*GR_CCSS*) to obtain a fast initial solution to the ILP problem. Algorithm 6 outlines the workflow of *GR_CCSS*, which takes a set of content items $\mathbb{C}$, SEC-enabled satellites $\mathbb{S}$, content popularity $\mathbf{P}(t)$, and network topology G_t at each time slot t as input. The algorithm returns a jointly optimized strategy for content cache placement and content server selection at each time slot t as output $\{x^{'*}(t), y^{'*}(t)\}$. Initially, $x^{'*}(t)$ and $y^{'*}(t)$ are defined as zero matrices of dimensions $|\mathbb{C}|\times|\mathbb{S}|$ and $|\mathbb{C}|\times|\mathbb{S}|\times(|\mathbb{S}|+1)$, respectively (*line 1*), signifying that all caches are initially unoccupied and no content servers have been designated. Next, the content items at each satellite node are ranked in descending order based on their popularity values, computed using Equation (3.9) (*line 2*). The caching process then proceeds sequentially, prioritizing the most popular content items while ensuring compliance with Constraints (3.24g), (3.24i), and (3.24j) (*line 6*). Concurrently, for each cached content item, the nearest node that satisfies

```
Algorithm 6: Proposed GR_CCSS Algorithm
    Input: $\mathbb{S}, \mathbb{C}, \mathbf{p}(t), G_t$
    Output: $\pi^{cnt,*}(t) \leftarrow \{x^{'*}(t), y^{'*}(t)\}$ ▷ Optimal solution at $t$
 1: $\{x^{'*}(t), y^{'*}(t)\} \leftarrow \{0^{|\mathbb{C}|\times|\mathbb{S}|}, 0^{|\mathbb{C}|\times|\mathbb{S}|\times(|\mathbb{S}|+1)}\}$
 2: Sort content items $\mathbb{C}$ on each satellite by popularity $\mathbf{p}(t)$ in descending order
 3: for Each satellite, $k \in \mathbb{S}$ in the network do
 4:     for Each content type, $c \in \mathbb{C}$ in the sorted list do
 5:         if Constraints (3.24g), (3.24i) and (3.24j) holds True then
 6:             $x^{'c}_{k}(t) \leftarrow 1$ ▷ Cache $c$
 7:             $y^{'c}_{k,h}(t) \leftarrow 1$ ▷ Find the nearest node $h$ satisfying constraints (3.24b) – (3.24f),
                (3.24h) and (3.24k)
 8:         end if
 9:     end for
10: end for
11: return $\pi^{cnt,*}(t) \leftarrow \{x^{'*}(t), y^{'*}(t)\}$ ▷ Update the optimal solution at $t$.
```

Constraints (3.24g), (3.24i), and (3.24j) is selected as the designated content server (*line 7*), where the *Dijkstra algorithm* is used to find the path between the cache and potential content servers. Finally, the algorithm updates and returns the optimal solution, $\pi^{cnt,*}(t) = \{x^{'*}(t), y^{'*}(t)\}$ (*line 11*).

In Algorithm 6, the *Dijkstra algorithm* is executed $\mathcal{O}(|\mathbb{S}|\times|\mathbb{C}|)$ times, each with a time complexity of $\mathcal{O}(|\mathbb{E}|+|\mathbb{N}|\log|\mathbb{N}|)$ [119]. Thus, the overall time complexity of Algorithm 6 is $\mathcal{O}(|\mathbb{S}|\times|\mathbb{C}|\ (|\mathbb{E}|+|\mathbb{N}|\log|\mathbb{N}|))$, indicating its dependence on both network size and content volume.

3.4.2 GA Based Algorithm

In this stage, we develop a GA-based meta-heuristic algorithm within *G-GenCache* to obtain a near-optimal solution to the ILP problem, achieving better performance than the Greedy-based approach within a reasonable computational time. GA is well-suited for solving this problem as it efficiently explores large solution spaces and proven convergence. Moreover, GAs provide a robust balance between exploration and exploitation, making them highly effective for *NP-hard* problems, including content caching tasks [119]. Inspired by biological evolution, GAs utilize terms such as *chromosome*, *gene*, *population*, *parents*, *offspring*, and *fitness function*. In GAs, each individual in a population represents a potential solution encoded by a chromosome composed of genes. In each generation, individuals with high fitness values are selected as parents for the next generation through a natural selection process. These parents undergo crossover operations to generate offspring, followed by mutation, which enhances genetic diversity by randomly modifying genes to prevent premature convergence. The proposed algorithm can address the risk of premature convergence to local optima by carefully selecting its parameters and leveraging the GA exploration-exploitation functionality. The primary components of GA are as follows:

Chromosome Encoding

G-GenCache has two components for chromosome encoding: content cache placement, $x^{'}(t)$ and content server selection, $y^{'}(t)$ strategy. We formulate chromosome q based on the solution represented by the joint decision variables $x^{'}(t)$ and $y^{'}(t)$, which have dimensions $|\mathbb{C}|\times|\mathbb{S}|$ and $|\mathbb{C}|\times|\mathbb{S}|\times(|\mathbb{S}|+1)$, respectively. Each gene within the chromosome corresponds to a binary entry $x^{'c}_{k}(t) \in x^{'}(t)$ and $y^{'c}_{k,h}(t) \in y^{'}(t)$.

Population Initialization

In *G-GenCache*, we begin by creating a population of size $\mathcal{N}_p$. Each chromosome q is generated by randomly generating the genes $x^{'c}_{k}(t) \in x^{'}(t)$ and $y^{'c}_{k,h}(t) \in y^{'}(t)$ while ensuring they satisfy Constraints (3.24b) to (3.24j). A diversified population enhances the ability to explore new solutions and avoid premature convergence.

Fitness Evaluation

In *G-GenCache*, the fitness of an individual q, denoted as $fitness(q)$, is evaluated using the objective function in Equation (3.24a). It is expressed as:

$$fitness(q) = \sum_{k\in\mathbb{S}}\left(w_1\eta_k^{cnt,H}(t) - (w_2\eta_k^{cnt,D}(t) + w_3\eta_k^{cnt,L}(t))\right) \tag{3.25}$$

where $\eta_k^{cnt,H}(t)$, $\eta_k^{cnt,D}(t)$, and $\eta_k^{cnt,L}(t)$ represent the average cache hit rate, content cache placement delay, and feeder link load, respectively, for the corresponding $x^{'}(t)$ and $y^{'}(t)$ of q.

Selection Operation

The selection process prioritizes individuals with higher fitness, increasing their chances of being chosen as parents for the next generation. We employ the *roulette wheel selection method* [120], which assigns selection probabilities to chromosomes based on their fitness. The selection probability of chromosome q from the current population is given by: $\mathcal{P}(q) = \frac{fitness(q)}{\sum_{l=1}^{\mathcal{N}_p} fitness(l)}$. Next, we determine how often a chromosome should appear in the next generation based on its expected count, defined by selection probability. The expected value of the selection probability, $\mathcal{P}(q)$ is:

$$E[\mathcal{P}(q)] = \frac{fitness(q)}{\frac{1}{\mathcal{N}_p}\sum_{l=1}^{\mathcal{N}_p} fitness(l)}. \tag{3.26}$$

The actual count is obtained by rounding the expected value to the nearest natural number, indicating which chromosomes are selected and their frequencies. Therefore, individuals with $E[\mathcal{P}(q)] \geq 1$ are selected as parents to generate offspring for the next generation. The frequency of an individual refers to how often it appears as a parent, with a higher frequency indicating its replacement of weaker individuals.

Crossover Operation

In the *G-GenCache*, we use the *uniform crossover method* [121] to create diverse offspring. This involves randomly selecting two parents ($q1$ and $q2$) and combining their cache placement strategies $\mathbf{x}^{q1}$ and $\mathbf{x}^{q2}$ using a random mask matrix. The mask matrix dictates which parent's bit each offspring element inherits. If an element of the mask matrix is greater than the crossover rate r_c, the bit is inherited from the first parent. Otherwise, it is inherited from the second parent. Careful selection of the crossover rate is important to avoid stagnation in local optima, ensuring the genetic algorithm maintains a balance between exploring new solutions and refining existing ones. If any of the offspring bits violates at least one of the constraints (3.24h) - (3.24j), the violating element is set to 0 to maintain validity. The next step is to determine the content server selection strategy $\mathbf{y}^q$ from the content cache placement strategy $\mathbf{x}^q$ after the crossover.

Algorithm 7 illustrates the content server selection process in *G-GenCache* for an individual q. It takes the cache placement strategy $\mathbf{x}^q$ of q, content freshness, $c^{\gamma,k}(t) \in \mathbb{C}^{\gamma}(t)$, network topology G_t, and available resources at each time slot t as inputs, returning a feasible content server selection strategy $\mathbf{y}^q$. Initially, $\mathbf{y}^q$ is set as a zero matrix of size $|\mathbb{C}|\times|\mathbb{S}|\times(|\mathbb{S}|+1)$ (*line 1*). For each satellite $k \in \mathbb{S}$ and content $c \in \mathbb{C}$ (where $x_k^{\prime c} = 1$ in $\mathbf{x}^q$), a content server $h \in \{\mathbb{S} \cup cps\}$ is randomly selected from nodes satisfying constraints (3.24b) to (3.24f) and

Algorithm 7: Content server Selection Algorithm

```
Input: $\mathbf{x}^q$, $G_t$, $\mathbb{C}^\gamma(t)$
Output: Content server selection strategy: $\mathbf{y}^q$
1: $\mathbf{y}^q \leftarrow 0^{|\mathbb{C}|\times|\mathbb{S}|\times(|\mathbb{S}|+1)}$
2: for Each satellite, $k \in \mathbb{S}$ in the network do
3:     for Each content type, $c \in \mathbb{C}$ do
4:         $h \leftarrow$ Select a source node for $c$ randomly from the nodes, $\mathbb{S} \cup cps$
5:         if $x_k^{\prime c} == 1$, then $y_{k,h}^{\prime c} \leftarrow 1$
6:     end for
7: end for
8: return $\mathbf{y}^q$ ▷ Update the content server selection strategy
```

(3.24h) (*line 4*). The corresponding node selection strategy is set to 1 ($y_{k,h}^{\prime c} = 1$ in $\mathbf{y}^q$), otherwise set to 0 (*line 6*). This process iterates for all satellites and contents. Finally, the algorithm returns the content server selection strategy $\mathbf{y}^q$ of q (*line 8*). The time complexity of Algorithm 7 is $\mathcal{O}(|\mathbb{S}|\cdot|\mathbb{C}|)$.

Mutation Operation

The mutation process introduces changes in the chromosomes. In this work, we generate a random value between 0 and 1 to determine the mutation. If this value exceeds the mutation rate r_m, a swap is performed between two satellites - one that already caches the content and one that does not. This swap updates the cache placement. The content server selection strategy is then computed using Algorithm 7, with the updated content cache placement strategy $\mathbf{x}^q$ as input. Mutation introduces diversity to explore better solutions. Careful selection of the mutation rate is important to avoid stagnation in local optima, ensuring the genetic algorithm maintains a balance between exploring new solutions and refining existing ones.

Termination

The execution of the GA algorithm terminates when the number of generations exceeds the upper bound $\mathcal{N}_g$, or the fitness of the elite remains unchanged (fitness gap $\epsilon_{GA} \approx 0$) for a specified number of consecutive generations, $const_M$.

We initialize the population of size $\mathcal{N}_p$ by randomly selecting cache nodes and content servers that satisfy the constraints (3.24b) – (3.24j) (*line 2*). Additionally, we determine the elite solution by computing the optimal strategy using Algorithm 6 (*line 4*). We identify the worst-performing individual with the lowest fitness value in the population (*line 5*). If its fitness is lower than that of the elite, we replace it with the elite and update the population accordingly (*line 7*). The elite individual is then updated with the best-performing individual in the population, *i.e.*, the one with the highest fitness value (*line 7*). The algorithm then iterates through a loop for $\mathcal{N}_g$ generations (*line 11*). In each generation, parents are selected from the current population based on the selection operation (*line 13*). These parents undergo crossover to generate offspring according to the crossover rate r_c (*line 14*). The offspring are then mutated based on the mutation rate r_m to introduce diversity and explore new solutions (*line 15*). We then evaluate the fitness of these offspring and identify the worst-performing individual (*line 16*). If its fitness is lower than that of the elite from the previous generation, the elite replaces the individual (*line 18*) to maintain solution quality. The population is updated with the new offspring, and the best individual becomes the elite of the current generation (*line 19*). This iterative process continues until the iteration counter reaches its maximum generation, $\mathcal{N}_g$, or the fitness of the elite remains unchanged for a specified number of consecutive generations. Finally, the algorithm updates the resources and outputs the final elite as the joint optimal content cache placement and content server selection strategy (*line 23*).

The innermost functions of Algorithm 8 include population selection, crossover, and mutation operations, each with a time complexity of $\mathcal{O}(\mathcal{N}_p)$. In addition, the algorithm incorporates the fitness function, which involves two main components: calculating the placement delay using the *Dijkstra algorithm*, with time complexity of $\mathcal{O}(|\mathbb{E}|+|\mathbb{N}|\log|\mathbb{N}|)$, and cache hit rate function, with time complexity of $\mathcal{O}(|\mathbb{S}|)$. These functions are invoked at most $\mathcal{N}_g$ times. Therefore, the overall time complexity of the *G-GenCache* algorithm is $\mathcal{O}(\mathcal{N}_p\ \mathcal{N}_g(|\mathbb{E}|+|\mathbb{N}|\log|\mathbb{N}|))$, determined by the number of individuals in each generation, the total number of generations, the number of contents, and the size of the network. *G-GenCache* algorithm is executed on the SDN controller, orchestrating caching across all satellites. In this model, the AoI parameter is precomputed and serves as input to the on-board caching algorithms, which do not significantly impact their convergence.

Algorithm 8: *G-GenCache* ALGORITHM

```
Input: $\mathbb{S}$, $\mathbb{C}$, $\mathbf{p}(t)$, $G_t$, $\mathcal{N}_p$, $\mathcal{N}_g$, $r_c$, $r_m$, $w_1$, $w_2$, $w_3$
Output: $\pi^{cnt,*}(t) \leftarrow \{x^{'*}(t), y^{'*}(t)\}$ Optimal strategy at $t$
Set $g \leftarrow 0$ ▷ Initialize the current generation with 0
Initialize population with randomly selected cache nodes and content servers that comply with the constraints (3.24b) – (3.24j)
$bestFit \leftarrow$ Fitness based on $\mathcal{E}^g$
Set $\mathcal{E}^g \leftarrow$ Optimal solution using Algorithm 6
Set $q \leftarrow$ The wort performing individual in the population based on fitness (3.25)
$worstFit \leftarrow$ Fitness of $q$
if $worstFit \leq bestFit$, then $q \leftarrow \mathcal{E}^g$
$\mathcal{E}^g \leftarrow$ Update the best performing individual in the population as elite
$Converged \leftarrow False$
$Count \leftarrow 0$
while $g \leq \mathcal{N}_g$ and not $Converged$ do
    $g \leftarrow g + 1$ ▷ select the next generation
    Select parents from the population based on the Selection Operation (3.26)
    Perform crossover to generate offspring based on $r_c$
    Apply mutation to each offspring based on $r_m$
    $q \leftarrow$ The worst performing individual from the mutated offspring
    $worstFit \leftarrow$ Fitness of $q$
    if $worstFit \leq$ Fitness of $\mathcal{E}^{g-1}$, then $q \leftarrow \mathcal{E}^{g-1}$
    $\mathcal{E}^g \leftarrow$ Update the best performing individual in the population as elite at $g$
    if (|Fitness of $\mathcal{E}^g$ − Fitness of $\mathcal{E}^{g-1}$|$\leq \epsilon_{GA}$) then $Count \leftarrow Count + 1$ else $Count \leftarrow 0$
    if ($Count \leq const_M$) then $Converged \leftarrow True$
end while
return $\{x^{'*}(t), y^{'*}(t)\} \leftarrow \mathcal{E}^g$ ▷ Update the optimal solution with the elite
```

3.5 Performance Evaluation

3.5.1 Simulation Settings

We evaluate the performance of *G-GenCache* by conducting simulations in the MATLAB platform, utilizing two datasets: (1) the *SES Two-Line Element (TLE) CelesTrak* dataset and *Iridium NEXT TLE CelesTrak* dataset [122] to develop a time-varying multi-layer satellite network topology and (2) the *US Zip Codes with Latitude and Longitude* dataset [123] to identify the locations of user terminals (*i.e.,* VSATs). For the simulation, we consider a multi-layer satellite network topology that includes 3 GEO satellites from SES, 20 MEO satellites from SES, and 50 LEO satellites from Iridium NEXT. The *US ZIP Code Geolocation* dataset contains approximately 40,000 ZIP codes across the United States as of 2018 with their latitude and longitude coordinates. We use the *Satellite Communications Toolbox* in

MATLAB to simulate the satellite network. This allows us to identify satellite-to-satellite and satellite-to-user terminal visibility over a *24 hour* period (*Jan. 24 - 25, 2024*) and determine the evolution of network topology during this time. Through extensive simulations, we identified a $\mathcal{T} = 6$ *hours* recurrence period during which the same topology reoccurs and remains unchanged for Δt = *15 minutes* (*i.e.*, each time slot duration) based on satellite-to-satellite and satellite-to-user terminal visibility, resulting in a total of $\mathcal{T} = 24$ time slots.

We consider 1000 content items initially stored at cloud-based server, *cps*. The cloud content provider stores all content items throughout the time slots, with an AoI of 0 to indicate that they remain fresh. To account for dynamic content requests, we randomly select 100 users from the pool of $40,000$ locations at each time slot. These user terminals then trigger a random set of content item requests following the Zipf probability distribution. The Zipf distribution exponent is set to $\delta = 0.5$. Before the first time slot ($t = 1$), all satellite caches are empty and have not stored any content on-board. We assume that all content items are initially stored at the cloud-based content provider. Consequently, each cache retrieves content exclusively from the cloud-based content provider in the first time slot. We set H as the number of hops between the target satellite and its other satellite along the shortest path between them. The proximity factor parameter α is set to 10, chosen by varying $\alpha \geq 0$ and analyzing its impact on the cache hit rate. Our observations indicate that the cache hit rate peaks around $\alpha = 10$, striking an optimal balance between content localization and resource utilization by effectively tuning the influence of proximity-based popularity relative to local content popularity. We also set the weight values as $w_1 = 0.8$, $w_2 = 0.1$, and $w_3 = 0.1$ to prioritize cache hit rate. The AoI threshold is set to be $a^c_{\max} \in [1, 12]$ time slots [109]. We set the GA parameters as follows: population size $\mathcal{N}_p = 100$, maximum number of generations $\mathcal{N}_g = 1000$, mutation rate $r_m = 0.05$, and the crossover rate $r_c = 0.95$. The rest of the simulation parameters are listed in Table 3.2.

3.5.2 Benchmark Scheme for *OCCR*

We evaluate the performance of *G-GenCache* against three benchmarks: Region Features Prediction cache (*RFP*), Cooperative Content Caching (*CCC*), and the Satellite Network Cache Placement Strategy (*PNCCP*), as proposed in [112], [95], and [111], respectively. RFP minimizes content delivery delay and bandwidth consumption by leveraging a region features prediction model based on ridge regression to update user preferences across geographic areas

Table 3.2: Simulation parameters for on-board content caching

Parameters	**Value**
Storage capacity of a satellite	[10-50] GB
User-LEO latency	[10- 20]ms
User-MEO latency	[29 - 70]ms
ISL propagation delay	[20-40]ms
User-GEO latency	[240 - 280]ms
Gateway-cloud delay	[100 - 150] ms
Content packet update generation rate, λ	[0.05 0.3] $\frac{Updates}{time\ slot}$ [109]
Packet size of content c, c^L	{ 32, 64K, 128K }
Content size of item c, c^β	[1 - 2] GB
Feeder uplink capacity	1 Gbps
ISL capacity	1 Gbps
μ	0.3 $\frac{Updates}{time\ slot}$

and employing a game theory-based cooperative caching algorithm for distributed decisions. However, it overlooks cache reconfiguration overhead and relies solely on ground-based cloud resources for cache updates. *CCC* aims to minimize cache content delivery delay by proposing a delay reduction gain for each cache. This approach focuses on caching content in neighboring nodes for collaborative caching and utilizing neighboring node resources. However, *CCC* does not consider cache reconfiguration overhead, freshness, timeliness of the cached contents, or caching from existing cached content copies. On the other hand, *PNCCP* seeks to minimize user retrieval delay. It begins by partitioning the topology using a spectral clustering algorithm. The next step involves collaborating nodes to cache content based on popularity. *PNCCP* is proposed for content caching in a single-layer (LEO) satellite network, aiming to minimize cache fetching delay without considering reconfiguration overhead or the timeliness and relevance of cached content items. Additionally, we compare *G-GenCache* with three commonly used cache replacement policies [124]: (1) First Input First Output (*FIFO*), which replaces the oldest content first; (2) Least Recently Used (*LRU*), which replaces content that hasnt been accessed recently; and (3) Least Frequently Used (*LFU*), which replaces the content with the fewest accesses. All benchmarks used cloud-based content provider servers c_p as the primary content servers, except for *PNCCP*, which also utilizes on-board caches.

3.5.3 Key Performance Indicators (KPIs)

- **Average Feeder Link Load** is the average bandwidth consumption ratio to a feeder link capacity, representing the traffic load of the feeder link during content cache placement (Equation (3.23)).

- **Average Content Placement Delay** is the average duration to transfer a content item from a designated server to its selected on-board cache (Equation (3.18)).

- **Average Cache Fetching Duration**: is the average delay per request to deliver a content item to an end user. We assume that a user request is associated with its nearest cache. In addition, cache-missed requests are associated to the cloud-based server. Moreover, the routing path between a user and its designated node consists of a set of links that belong to the shortest path determined using the Dijkstra shortest path algorithm. The average cache fetching delay per request is:

$$d_{av} = \frac{1}{\mathcal{T}} \sum_{t=1}^{\mathcal{T}} \left(\frac{\sum\limits_{u \in \mathbb{U}} \sum\limits_{c \in \mathbb{C}} d_{k,u}^{cnt,c} \; Q_u^c(t) \; x_k^{\prime c}(t)}{\sum\limits_{u \in \mathbb{U}} \sum\limits_{c \in \mathbb{C}} Q_u^c(t)} \right), \tag{3.27}$$

  where $d_{k,u}^{cnt,c}$ denotes the cache fetching duration of content item $c \in \mathbb{C}$ to user terminal $u \in \mathbb{U}$, including both transmission and propagation delays, and $Q_u^c(t)$ represents the number of requests made by user terminal u for content c during time slot t.

- **Average Cache-Hit Rate**: is the ratio of user requests successfully served from the on-board cache with fresh and relevant content (i.e., within its validity timeline) to the total number of requests at each time slot.

3.5.4 Results And Discussion

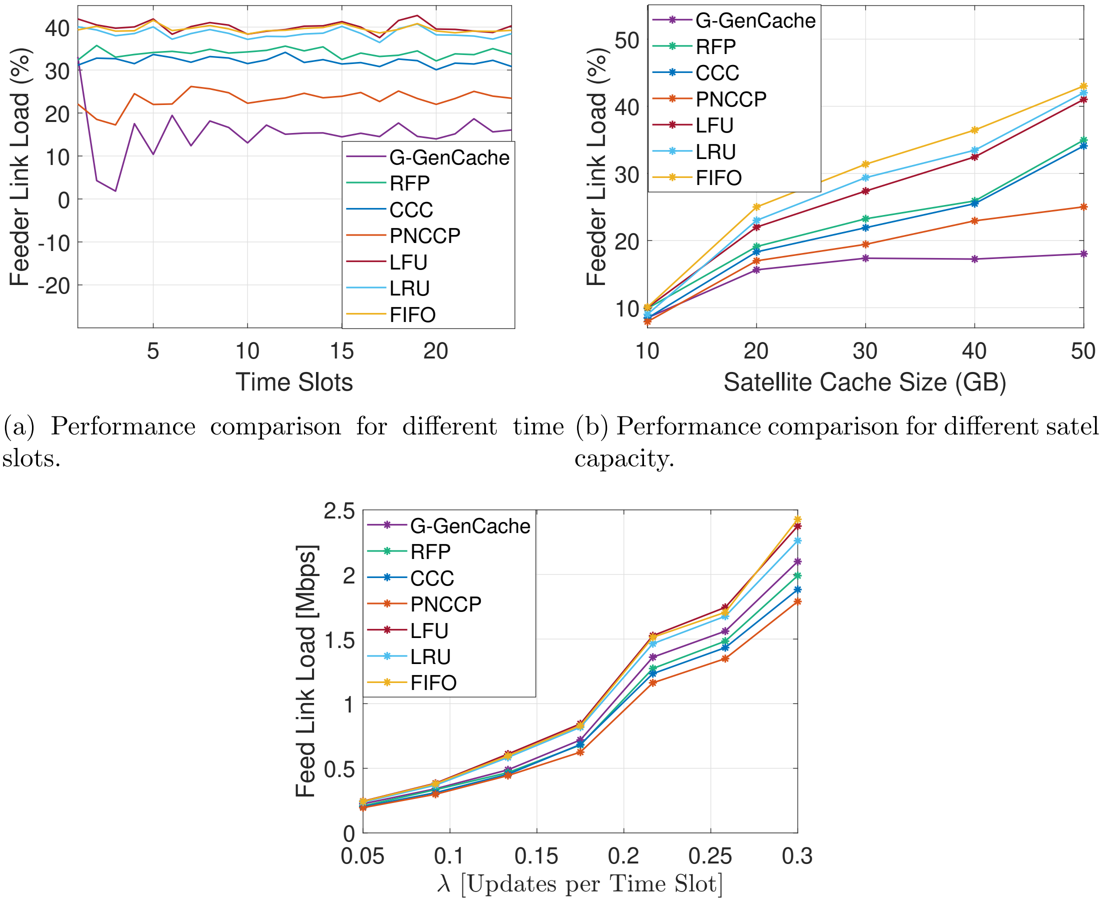
Feeder Link Load

As depicted in Figure 3.4a, *G-GenCache*, *RFP* and *CCC* initially show similar performance, both relying exclusively on cloud-based server to update on-board caches via feeder links. In contrast, the *FIFO*, *LFU*, and *LRU* algorithms perform poorly due to their sole reliance on cloud-based content server and inefficient content update models that depend on cache access timing, leading to excessive content uploads. *PNCCP* initially results in the lowest feeder link load compared to other algorithms, as it leverages content copies from other caches to update caches with similar content. In subsequent time slots ($t > 1$), *G-GenCache* consistently demonstrates a reduced feeder link load compared to all benchmark strategies. For instance, at the third time slot ($t = 3$), *G-GenCache* lowers the feeder link load by 15.4%

relative to *RFP*, by 30.4% compared to *CCC*, and by 31.4% compared to *PNCCP*. Moreover, the proposed solution achieves even greater reductions when compared to traditional caching strategies, with a 36.2% reduction relative to *LRU*, a 37.2% reduction compared to *LFU*, and a 37.9% reduction compared to *FIFO* at the same time slot ($t = 3$). Similarly, at the sixth time slot ($t = 6$), *G-GenCache* reduces the feeder link load by 16.1% compared to *RFP*, by 12.6% compared to *CCC*, and by 4.1% compared to *PNCCP*. The feeder link load of the conventional caching algorithms (*LRU*, *LFU*, and *FIFO*) remains consistently high due to their reliance on access time rather than content popularity, often causing eviction of popular content and frequent offloading from the ground. The proposed solution outperforms the benchmarks by explicitly accounting for cache reconfiguration overhead and employing an efficient cache-to-cache update scheme. This approach leverages on-board cached content to update caches, minimizing dependence on ground infrastructure and significantly reducing feeder link load.

Figure 3.4b demonstrates that applying the proposed *G-GenCache* results in a lower feeder link load than all benchmarks across varying satellite storage capacities. For instance, when the on-board storage capacity is 20 GB, *G-GenCache* reduces feeder link load by 1.34% compared to *PNCCP*, 2.67% compared to *CCC*, 3.46% compared to *RFP*, 6.36% compared to *LFU*, 7.36% compared to *LRU*, and 9.36% compared to *FIFO*. Similarly, when the on-board storage capacity increases to 50 GB, G-GenCache reduces the feeder link load by 7.0% compared to *PNCCP*, 16.08% compared to *CCC*, 16.95% compared to *RFP*, 24.0% compared to *LFU*, 25.0% compared to *LRU*, and 25.1% compared to *FIFO*. This superior performance of the proposed solution stems from two key factors. First, unlike all benchmark schemes, *G-GenCache* explicitly accounts for cache reconfiguration overhead in its optimization process, avoiding unnecessary cache updates that would otherwise increase feeder link load. Second, unlike most benchmarks (except *PNCCP*), *G-GenCache* employs a cache-to-cache update mechanism, leveraging already cached on-board content to update neighboring caches, which reduces reliance on terrestrial infrastructure and significantly lowers the feeder link load. In Figure 3.4b, the feeder link load fluctuates over time slots due to dynamic content requests and varying network topologies.

Figure 3.4c illustrates the impact of status update packet streaming on feeder link load across varying update packet arrival rates, λ. The results demonstrate an approximately exponential increase in feeder link load as the arrival rate increases. This feeder link load is



(a) Performance comparison for different time slots.

(b) Performance comparison for different satellite capacity.

(c) Performance comparison for different status update rate.

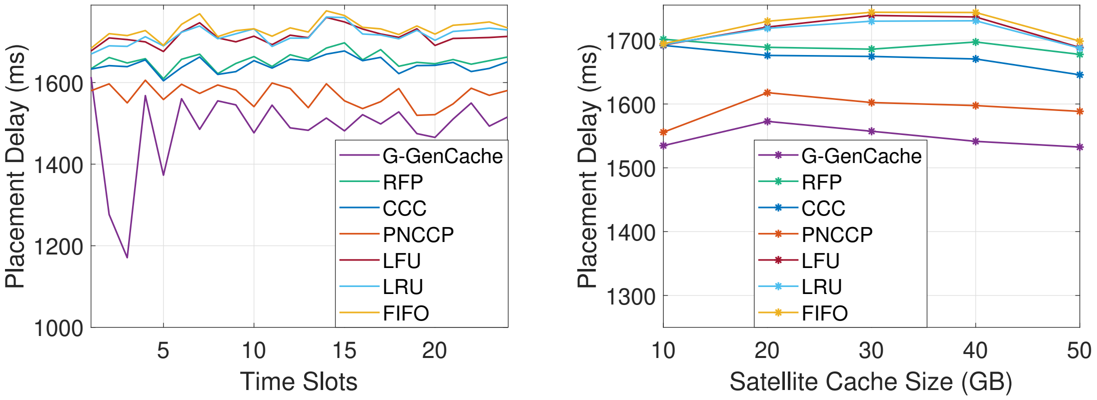
Figure 3.4: Average feeder link load for different on-board content caching schemes

influenced by the number of uniquely cached content items on-boardthe greater the number of unique content items cached, the more status update packets are required to maintain content freshness. The figure shows that the *FIFO*, *LRU*, and *LFU* algorithms exhibit higher feeder link loads due to their suboptimal caching strategies. These algorithms cache content primarily based on time of arrival or access frequency, rather than content popularity or relevance, leading to a higher number of unique content items that require frequent updates. In contrast, the proposed algorithm, *G-GenCache*, incurs a slightly higher feeder link load compared to *RFP*, *CCC*, and *PNCCP* because it utilizes on-board resources more efficiently and caches a larger number of content items. However, since status update packets are relatively small, their contribution to the overall feeder link load remains significantly lower compared to the load associated with caching or offloading primary content items. Consequently, the impact of λ is negligible at lower arrival rates but becomes increasingly significant as λ grows. As demonstrated in Figures 3.4a and 3.4b, the overall feeder link load of the proposed solution remains lower than that of the benchmark. Although the feeder link load due to status update packet streaming is slightly higher in the proposed solution, its impact is negligible and is dominated by the feeder link load associated with primary content caching on-board.

Content Placement Delay

As shown in Figure 3.5a, during the first time slot, *G-GenCache* performs similarly to *RFP* and *CCC* in terms of content cache placement delay since all schemes rely exclusively on the cloud-based content provider to update initially empty satellite caches, resulting in uniformly high delays. In subsequent time slots ($t > 1$), the content placement delay for *G-GenCache* significantly decreases compared to the benchmarks. For example, at the third time slot ($t = 3$), the average content cache placement delay of the proposed solution is reduced by 32.41% compared to *PNCCP*, 39.98% compared to *CCC*, 41.02% compared to *RFP*, 45.67% compared to *LFU*, 44.25% compared to *LRU*, and 46.48% compared to *FIFO*. Similarly, at the sixth time slot ($t = 6$), the average content cache placement delay is reduced by 2.29% compared to *PNCCP*, 4.92% compared to *CCC*, 5.69% compared to *RFP*, 10.44% compared to *LFU*, 10.47% compared to *LRU*, and 11.65% compared to *FIFO*. The improvement in performance in *G-GenCache* is due to its consideration of cache reconfiguration overhead and the use of cache-to-cache updates by selecting nearby content servers from the on-board cached content, reducing reliance on ground-based content providers and lowering content placement delays. The variation in content placement delays across time slots, as shown in Figure 3.5a, is due to dynamic content requests and fluctuating network topologies.

Figure 3.5b compares the performance of *G-GenCache* against benchmark algorithms in content placement delay across varying satellite cache capacities. The figure shows that on-board cache capacity slightly influences the content cache placement delay. Instead, the delay primarily depends on content server selection. The figure shows that *G-GenCache* achieves faster content placement, reducing delay by 8.37%, 10.10%, 12.27%, 12.26%, 12.67%, and 13.16% compared to *PNCCP*, *CCC*, *RFP*, *LFU*, *LRU*, and *FIFO*, respectively, at a satellite capacity of *40 GB*. The performance improvement of *G-GenCache* primarily stems from its focus on minimizing reconfiguration overhead, which directly reduces content cache placement delays. In addition, the AoI-aware caching scheme in *G-GenCache* enhances on-board cache



(a) Performance comparison for different time slots.

(b) Performance comparison for different satellite capacity.

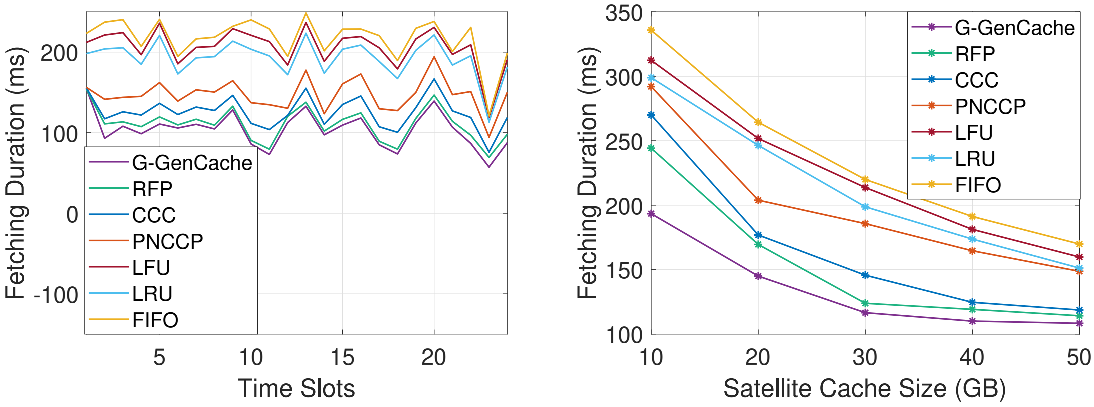
Figure 3.5: Average content placement delay for different on-board content caching schemes

utilization by using satellites as content servers instead of relying solely on the cloud-based content provider. This enables cache updates from one satellite to another, facilitating faster updates.

Cache Fetching Duration

As shown in Figure 3.6a, *G-GenCache* consistently achieves the fastest average cache fetching duration compared to all benchmarks across all time slots. For example, at the third time slot (t = 3), *G-GenCache* fetches content 33.18% faster than *PNCCP*, 16.59% faster than *CCC*, 5.04% faster than *RFP*, 107.57% faster than *LFU*, 90.14% faster than *LRU*, and 122.48% faster than *FIFO*. Similarly, at the sixth time slot (t = 6), *G-GenCache* fetches content 31.55% faster than *PNCCP*, 15.78% faster than *CCC*, 3.61% faster than *RFP*, 75.26% faster than *LFU*, 63.41% faster than *LRU*, and 83.98% faster than *FIFO*. These improvements demonstrate the capability of *G-GenCache* to accelerate content delivery and enhance system responsiveness in satellite edge networks. This superior performance of *G-GenCache* stems from the efficient on-board resource utilization through the proximity-based content popularity caching scheme proposed in this chapter. This strategy ensures that more popular content is cached on-board, which reduces cache fetching duration. Furthermore, as observed from Figure 3.6a, the cache fetching duration varies across time slots due to fluctuations in the topology. The route from the end user to the caches changes based on the availability of links across time slots, which leads to fluctuations in the cache fetching duration.

Figure 3.6b illustrates that *G-GenCache* consistently achieves the fastest content distri-



(a) Performance comparison for different time slots.

(b) Performance comparison for different satellite capacity.

Figure 3.6: Average cache fetching duration for different on-board content caching schemes

bution to end users across different satellite capacities compared to all benchmarks. For example, at a 10GB satellite capacity, *G-GenCache* outperforms *PNCCP* by 50.96%, *CCC* by 39.59%, *RFP* by 26.27%, *LFU* by 61.54%, *LRU* by 54.53%, and *FIFO* by 73.63%. At a higher capacity of 50GB, it maintains a significant advantage, fetching content 37.22% faster than *PNCCP*, 9.53% faster than *CCC*, 5.41% faster than *RFP*, 47.47% faster than *LFU*, 39.53% faster than *LRU*, and 56.74% faster than *FIFO*. This performance improvement of *G-GenCache* is because of the efficient utilization of on-board resources through the proximity-based content popularity model. Furthermore, we can observe from the figure that as satellite capacity increases, the cache fetching duration decreases because larger caches can store more content locally, allowing user requests to be served directly from the onboard cache instead of relying on the ground infrastructure.

Cache Hit Rate

As shown in Figure 3.7a, in the initial time slot, where the cloud-based content provider server serves as the sole content source for all content items, *G-GenCache* demonstrates a cache hit rate comparable to that of *CCC* and *RFP* (*i.e.,* approximately *72.65%*). This is because no content is cached on-board initially, requiring all content cache updates to be fetched from the cloud-based provider via feeder links. As a result, the increased feeder link load and content cache placement delay contribute to a lower cache hit rate in *G-GenCache*, as the algorithm seeks to balance the cache hit rate with cache reconfiguration overhead. However, initially, *G-GenCache* performs similarly to *CCC* and *RFP* due to its effective utilization of on-board cache resources, enabled by the novel proximity-based content popularity caching scheme. However, *G-GenCache* outperforms the benchmarks in the rest of the time slots, achieving an average cache hit rate of 82.92%, compared to 77.52% of *RFP*, 74.36% for *CCC*, 70.35% for *PNCCP*, 50.75% for the *FIFO*, 53.02% for *LFU*, and 54.15% for *LRU* algorithms. The performance improvement of *G-GenCache* can be attributed to its focus on maximizing the cache hit rate through efficient utilization of on-board resources. Incorporating proximity-based content popularity caching and AoI-aware content caching schemes optimizes the on-board content caching, resulting in a higher cache hit rate. As shown in Figure 3.7a, the cache hit rate varies over time due to fluctuations in satellite cache capacities and the mobility of satellites. Cache node selection is influenced by content requests and satellite capacities, with satellites having higher cache capacities likely moving closer to users, thus increasing the cache hit rate. Conversely, lower-capacity satellites may be closer to users, limiting on-board content storage and reducing the cache hit rate. Therefore, the mobility of satellites with varying capacities leads to fluctuations in the cache hit rate across time slots.

Figure 3.7b compares the performance of *G-GenCache* with the benchmarks across different satellite cache capacities. As shown in the figure, *G-GenCache* consistently achieves a higher cache hit rate than the benchmarks across all satellite cache capacities. At a satellite capacity of 10GB, as shown in the figure, *G-GenCache* outperforms *PNCCP* by 18.23%, *CCC* by 16.10%, and *RFP* by 11.26%. It also achieves a 19.94% higher cache hit rate than *LFU*, 18.84% better than *LRU*, and 21.65% better than *FIFO*. At a 50GB satellite capacity, as shown in the figure, *G-GenCache* shows a 7.64% improvement over *PNCCP*, a 1.12% improvement over *CCC*, and a 0.12% improvement over *RFP*. Notably, it outperforms *LFU* by 25.88%, *LRU* by 21.90%, and *FIFO* by 30.02% at the 50GB satellite capacity. The enhanced performance of *G-GenCache* over the benchmarks is because it efficiently utilizes on-board resources, enabled by integrating the proximity-based content popularity model and the AoI-aware caching scheme. Moreover, all algorithms show an exponential increase in cache hit rate as satellite cache capacity grows, as shown in the figure. In addition, the performance gap narrows as satellite cache capacity increases, since a larger cache allows more content to be stored on-board, reducing the impact of optimization decisions.

Furthermore, we compare the initial greedy-based solution for *G-GenCache* (*i.e., GR_CCSS* algorithm) to assess the benefits of the proposed GA-based solution. As shown in Figure 3.7b, *G-GenCache* improves the cache hit rate by *9%* on average over the *GR_CCSS*. However, in scenarios with large cache sizes, both algorithms achieve similar performance since high-capacity satellite caches provide greater flexibility to store more content. As a result, the advantage of the GA-based solution diminishes compared to the greedy-based solution, GR_CCSS, which benefits from faster convergence and lower computational complexity.

To assess the impact of AoI-awareness, we compare the cache hit rate performance of the proposed G-GenCache algorithm with and without AoI-awareness across multiple time slots. As illustrated in Figure 3.7d, the cache hit rate margin between the AoI-aware and AoI-unaware schemes increases over time. This improvement is attributed to the fact that, without AoI-awareness, outdated content is repeatedly utilized as a source for cache updates, leading to a gradual decline in content relevance. In contrast, AoI-awareness ensures that only fresh and timely content is used, which improves cache hit rate and overall system performance in the long term. The cache hit rate margin between the AoI-aware and AoI-unaware G-GenCache increases over time, as the repeated use of on-board cached content without freshness consideration leads to content obsolescence and reduced relevance. This improvement demonstrates that incorporating AoI-awareness, along with the utilization of on-board cached content as source content, not only reduces cache reconfiguration overhead but also provides additional flexibility to maintain fresher and more popular content on-board, thereby enhancing the cache hit rate over time.

Figure 3.7c presents a performance comparison of the proposed solution against the benchmark caching strategies under varying Zipf exponents. The results demonstrate that the proposed caching strategy consistently outperforms the benchmark strategies across all values of δ. When δ is small, content popularity becomes more evenly distributed; in other words, user requests are spread across a broader range of content items. In such scenarios, increasing the diversity of cached contents helps improve the average cache hit rate by efficiently utilizing on-board resources. However, the limited storage capacity of satellites constrains the number of contents that can be cached, leading to a relatively lower cache hit rate. Conversely, when δ is large, content popularity becomes highly skewed, with a few content items dominating user requests. Caching these highly popular content items results in a significantly higher cache hit rate. Consequently, as shown in the figure, the cache hit rate improves as δ increases. The proposed proximity-based content popularity model ensures efficient utilization of on-board resources, outperforming all benchmark strategies across both low (*e.g.*, $\delta = 0.2$) and high (*e.g.*, $\delta = 1.0$) Zipf exponents. Specifically, at $\delta = 0.2$, the proposed G-GenCache algorithm

Table 3.3: Hierarchical layer contributions in the three-tier satellite network.

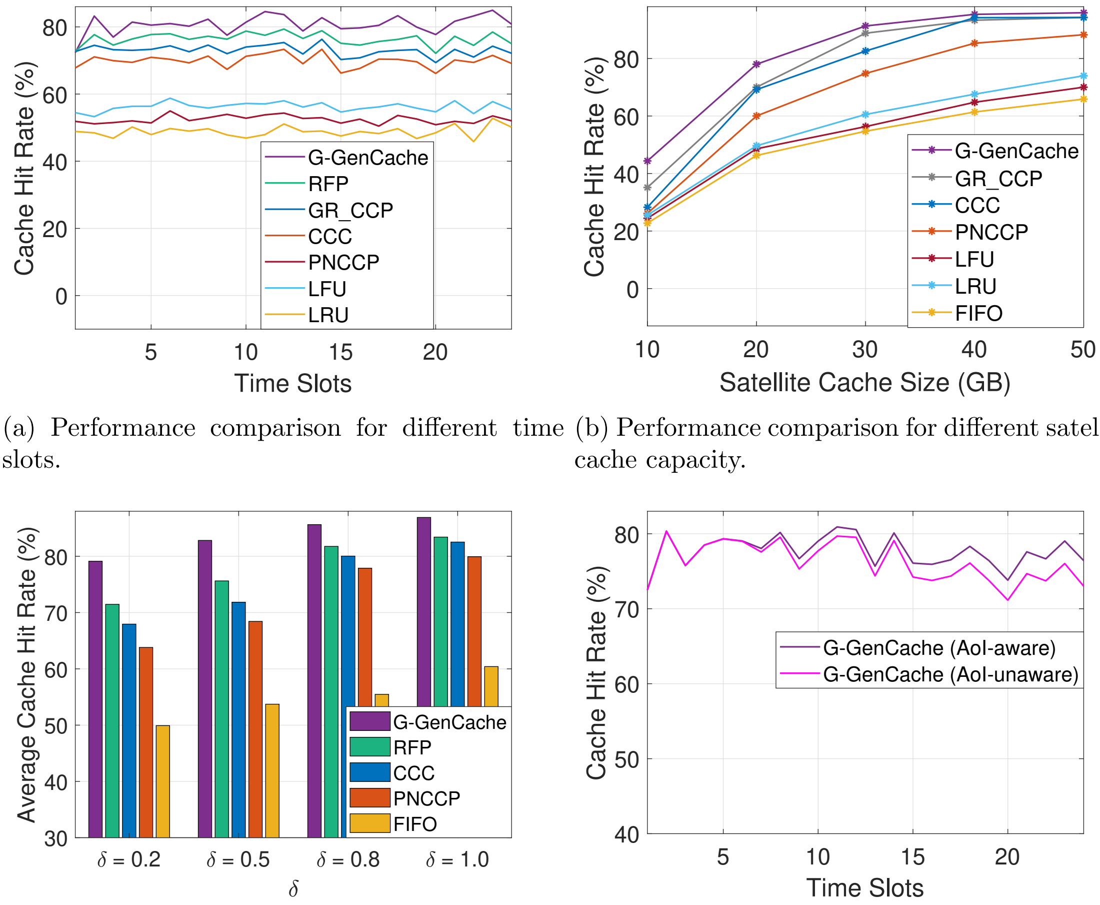
Cache Capacity [GB]	**LEO [%]**	**MEO [%]**	**GEO [%]**	**System Performance [%]**
10	31.8345	19.5036	2.3926	54.333
20	47.6681	29.4819	2.5101	78.419
30	56.3456	32.3611	2.6141	90.675
40	60.7304	33.1016	2.9176	94.184
50	61.3322	34.5085	2.9350	94.847

surpasses RFP by 8%, CCC by 13%, PNCCP by 17%, and FIFO by 30%. When $\delta = 1.0$, where requests are highly concentrated on a few popular contents, all content popularity-based caching strategies except FIFO achieve comparable performance due to the dominance of a small set of contents.

Table 3.3 summarizes the contribution of each satellite layer (LEO, MEO, and GEO) to the overall cache hit rate in the multi-tier satellite network. We observe that the sum of the individual cache hit rates from each layer exceeds the overall system cache hit rate. This is because certain content is redundantly cached across multiple layers to ensure content availability and improve reliability in delivery. Further analysis shows that the contribution to the cache hit rate decreases as we move from the LEO to the higher layers (MEO and GEO). This is attributed to the reduced number of satellites in higher layers, as fewer satellites are available in MEO and GEO compared to LEO. Additionally, higher-layer satellites are able to cache a larger proportion of globally popular content, benefiting from their broader coverage area that attracts a larger number of user requests. On the other hand, lower-layer satellites primarily cache locally popular content due to the nature of the Zipf distribution and their limited coverage area. These findings underscore the role of different satellite layers in optimizing the cache hit rate, with each layer playing a unique role in balancing global and local content caching and highlighting the trade-off between coverage and the number of available satellites.

Comparison With Optimal Solution

This subsection evaluates the performance deviation, or optimality gap, between our proposed *G-GenCache* and the optimal solution. We used the `ILP` model formulated in Equation (3.24), solved with the CVX MOSEK optimization solver [125], considering 50 content items and a 5GB on-board cache capacity per satellite. To introduce diversity, we set the Zipf exponent $\delta = 0.1$, considering only the first 5 time slots with an AoI threshold of 30 minutes



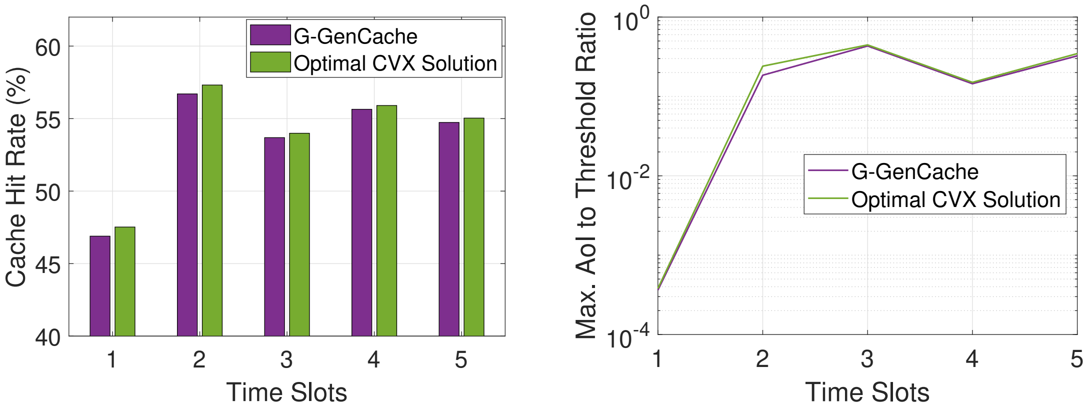
(a) Performance comparison for different time slots.

(b) Performance comparison for different satellite cache capacity.

(c) Performance comparison for different Zipf exponents.

(d) Impact of AoI-awareness on cache hit rate for different time slots.

Figure 3.7: Average cache hit rate for different on-board content caching schemes



(a) Average cache hit rate

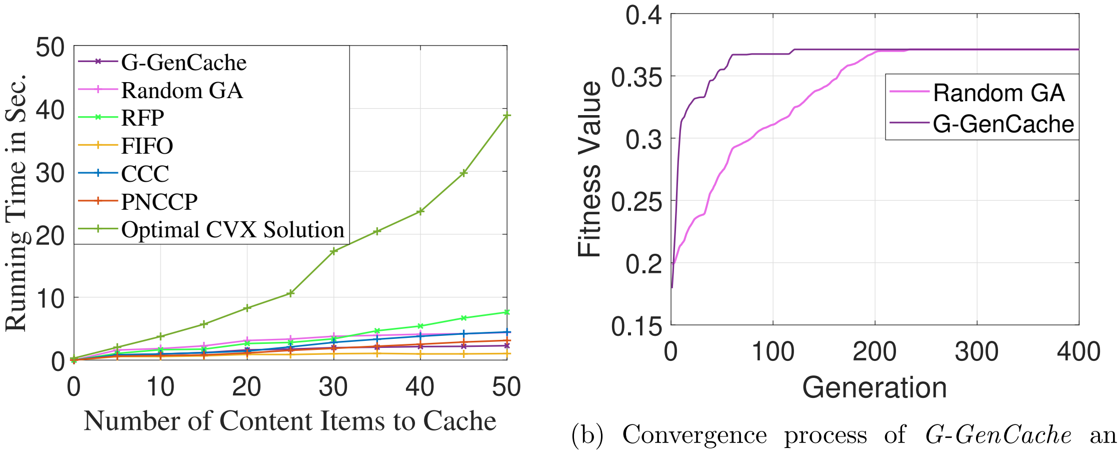
(b) Cached content freshness

Figure 3.8: Performance comparison with optimal *CVX* solution

(2 Δt). Figure 3.8a shows that *G-GenCache* performs comparably to the optimal *CVX* solution, with minimal optimality gaps (*0.92%* for cache hit rate). Figure 3.8b compares the maximum cached content's AoI with the corresponding threshold, highlighting any violations of the content freshness constraints. *G-GenCache* employs a caching strategy that ensures content relevance while approaching the optimal *CVX* solution performance. The small margin observed in Figure 3.8 demonstrates that the optimality gap of the proposed algorithm is minimal, confirming that our approach yields near-optimal performance.

Computational Time

Figure 3.9 compares the time complexity of the proposed *G-GenCache* algorithm against benchmark schemes in terms of their running time as the number of content items increases. As shown in Figure 3.9a, the running time of the optimal *CVX* solution increases almost exponentially with the number of content items, making it impractical for large-scale problems due to its high computational cost. Similarly, the game-theoretic *RFP* algorithm demonstrates relatively low running time at small scales but grows nearly quadratically as the number of content items increases, limiting its scalability. The proposed *G-GenCache* algorithm, which combines a greedy algorithm for initialization with a *GA* for refinement, consistently outperforms the randomly initialized *GA* (*Random GA*) in running time across all content sizes. This improvement is attributed to the lower complexity of the greedy initialization, which significantly accelerates convergence, as further illustrated in Figure 4.6. While *FIFO* shows the lowest computational complexity, it lacks optimization capability. *PNCCP* and *CCC*



(a) Running time comparison.

(b) Convergence process of *G-GenCache* and *Random GA*.

Figure 3.9: Time complexity comparison

exhibit slightly higher running times than *FIFO* but remain more efficient than *Random GA* and *CVX*. *G-GenCache* achieves a balanced trade-off by leveraging the fast convergence of greedy initialization and the refinement capability of *GA*, resulting in better performance with manageable computational complexity. As the number of content items increases, the running time of the benchmark algorithms grows significantly faster than that of the proposed solution. In contrast, the proposed G-GenCache algorithm slightly increases in running time, indicating its stability and scalability, which makes it suitable for large-scale problems.

3.6 Conclusion

This chapter addresses dynamic on-board content cache placement and content server selection to maximize cache hit rate while minimizing cache reconfiguration overhead in multi-layer satellite networks, encompassing SEC-enabled LEO, MEO, and GEO satellites. The *G-GenCache* algorithm is developed to provide suboptimal solutions to the proposed `ILP` problem. We also introduce an AoI-aware content caching approach and a proximity-based content popularity model to ensure efficient on-board storage use while maintaining cached content freshness. Extensive simulation results demonstrate the effectiveness of the proposed algorithm in improving content cache hit rate while reducing cache reconfiguration overhead. Simulation results demonstrate that *G-GenCache* outperforms *FIFO*, *LFU*, *LRU*, *RFP*, *CCC*, and *PNCCP* by achieving a higher cache hit rate while reducing cache fetching duration, feeder link load, and content placement delay, all with low computational complexity. Specifically, *G-GenCache* attains an average cache hit rate that is 7.2% higher than *RFP*, 13.1% higher than *CCC*, 16.5% higher than *PNCCP*, and over 27.8% higher than *FIFO*. As future work, we plan to extend our framework by jointly integrating edge computing tasks with caching decisions to optimize resource utilization further. Additionally, leveraging machine learning techniques for adaptive content popularity prediction in highly dynamic network topologies will be explored to enhance caching efficiency and responsiveness.

Chapter 4

Content Distribution in Satellite-Terrestrial Integrated Networks

With the surging demand for data-intensive applications, ensuring seamless content delivery in STINs is crucial, especially for remote users. DAI enhances monetization and user experience, while MEC in STINs enables distributed content caching and ad insertion, improving both content provider revenue and user experience. However, satellite mobility and time–varying topologies cause service disruptions, while excessive or poorly placed ads risk user disengagement, impacting revenue. This chapter addresses service continuity–aware caching and revenue–driven collaborative seamless content distribution with DAI in STINs while modeling user disengagement cost. The problem is formulated as two hierarchical *ILP* optimizations: one maximizing cache hit rate and another optimizing content distribution with DAI to maximize revenue, minimize end–user costs, and enhance user experience. We proposed algorithms for greedy content caching, greedy content distribution for a fast initial solution, and *BPSO*–based content distribution as an enhanced solution with a collaborative strategy to handle unassociated requests. Extensive simulations demonstrate that the proposed approach closely approximates the optimal *ILP* solution while outperforming the benchmarks, achieving over a 5% increase in revenue and a 33% reduction in cache retrieval duration.

4.1 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 [42]. STINs enhance broadband connectivity in remote and underserved areas by combining satellite coverage with terrestrial network capacity [47]. 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 MEC with satellite systems, advancing the concept of SEC. For instance, Axiom Space and Red Hat take edge computing into orbit to support space–based cloud services [126]. SEC reduces feeder link congestion and delivery delays by allowing satellite nodes to cache content and process data on–board. 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 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 [127]. 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) [128]. However, misaligned SCTE–35 markers [129] (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 SSAI, which define the ad stitching process at the user device and server, respectively [130]. The SSAI architecture is preferred over CSAI for its robustness against ad blockers, as it stitches ads directly at a cloud server [130]. 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.

We utilize multi–layer satellite networks, comprising GEO and 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.

This chapter investigates service continuity–aware caching and revenue–driven collaborative seamless content distribution with DAI in STINs while modeling user disengagement cost. The user disengagement cost is modeled using four engagement sensitivity factors: content type, ad runtime relative to time slot duration, ad–content language misalignment[1],

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

and user demographics. The proposed approach includes algorithms for optimizing content caching placement, greedy content distribution with DAI, enhanced content distribution through BPSO, and managing unassociated content requests in a distributed real–time decision–making framework. Our contributions are summarized as follows:

1. We formulate revenue–driven content caching and distribution with DAI as a hierarchical *ILP* optimization to maximize provider revenue while minimizing end–user costs. We also propose a distributed SSAI–based ad stitching architecture, treating ad insertion as a network function executed at cache nodes constrained by computing capacity.

2. To tackle the *NP–hard* hierarchical *ILP* problem, we propose a Greedy–based algorithm for efficient and fast content caching and distribution, and a *BPSO*–based approach to refine user–cache associations and ad insertion. Additionally, we discuss a distributed, real–time decision–making strategy to resolve unassociated requests via proximity–based collaboration.

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.

4.2 Related Works

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

4.2.1 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. [131] presented a delivery delay minimization strategy for cache placement. In our previous work [132], 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.* [133] proposed a content delivery scheme to minimize the delivery delays. Tang *et al.* [134] 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.* [110] proposed a density–based cache distribution strategy for STINs, considering dynamic satellite topology and uneven user distribution. Similarly, the authors of [135] studied collaborative caching between LEO satellites and distribution approach, modeling the time–varying topology as a time–varying graph. Shushi *et al.* [136] explored a layered coded cache placement and distribution strategy, enabling collaborative distribution between satellites and base stations.

Although existing works addressed time–varying topology by analyzing content requests and network topology on a per–time–slot basis, assuming a quasi–static topology within each slot [110, 135, 136], they did 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.

4.2.2 DAI in STINs

Advertising is vital for monetizing content delivery, driving standardization efforts like SCTE–35 and SCTE–104 [137] and advancements in ad insertion architectures such as SSAI and CSAI solutions [130]. For example, Pham *et al.* [138] integrated ad insertion standards with MPEG DASH for HTML5–based platforms to enable interoperable ad insertion. Similarly, the authors of [137] explored ad substitution using SCTE–35 in DASH workflows for Over–The–Top (OTT) services. On the other hand, the authors of [130] proposed SSAI architectures in cloud–based and cloud–assisted content delivery to enhance ad personalization. Seeliger *et al.* [139] 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 [130]. 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 content is stored. 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 ad insertion boosts revenue for content providers, poor DAI design leads to content abandonment, ultimately reducing provider revenue. Factors such as ad runtime duration, relevance to the user (*e.g.*, language alignment with the primary content), and the frequency of ads influence user engagement. The 2021 Conviva report states that nearly 20% of viewers abandon the content after a 5 seconds ad delay [140]. This chapter considers four key factors that contribute to user disengagement: content type (*e.g.,* live sports are more ad–sensitive than on–demand videos) [141], user demographics (*e.g.,* younger viewers are more sensitive to ads than older ones) [142,143], ad duration relative to primary content [144], and ad–content language misalignment with the main content [145].

4.3 System Model and Problem Formulation

This section presents the mathematical model for collaborative revenue–driven content distribution with DAI in an MEC–enabled STIN.

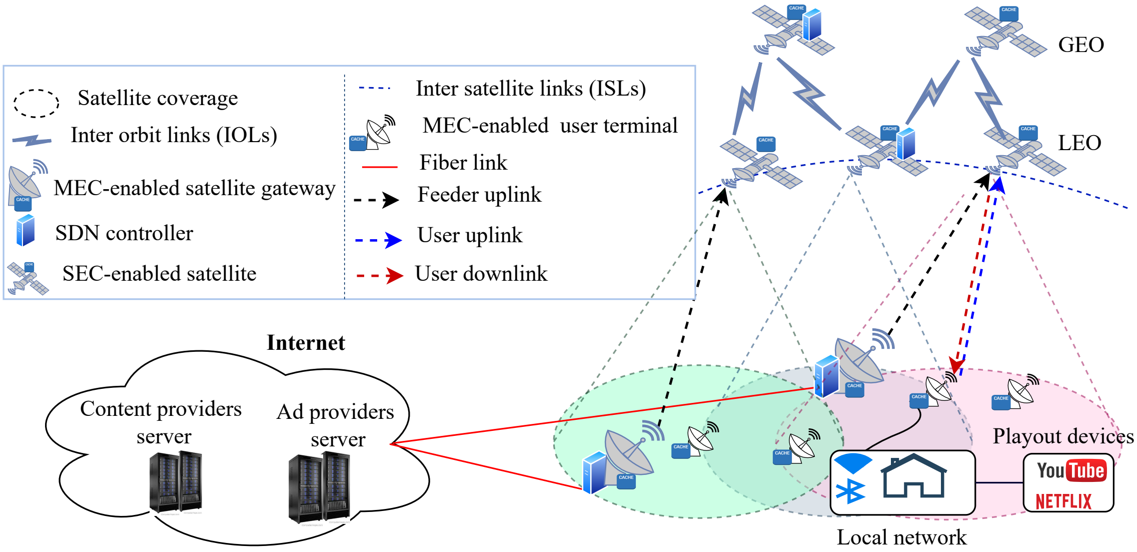


Figure 4.1: MEC–enabled STIN for supporting content caching and distribution with DAI.

4.3.1 System Model

As illustrated in Figure 4.1, 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 ISLs and inter–orbital links RF or FSO links. The terrestrial network includes a cloud–based Content Provider (CP), a cloud–based Ad Provider (AP) server, MEC–enabled satellite gateways, and local networks with MEC–enabled terminals like 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 [146]. 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 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 [114] can be used to determine their location in our use case.

Network Model

We model the time–varying STIN topology as sequential time slots, each with a duration of Δt time units ($\mathbb{T} = \langle 1, \cdots, t, \cdots, \mathcal{T} \rangle$), where $\mathcal{T}$ is the recurrence period. During each t, the topology is considered quasi–static and represented by an undirected graph $G_t = (\mathbb{N}, \mathbb{E})$, where $\mathbb{N}$ denotes the set of all nodes (*i.e.,* satellites, gateways, VSATs, content provider server, and ad provider server) and $\mathbb{E}$ represents the set of links within the time slot, connecting the nodes in the network. Each node $k \in \mathbb{N}$ is represented by a tuple $\langle k^{\beta}, k^{\mu}, k^{\mathcal{N}(t)} \rangle$, where k^{β}, k^{μ}, and $k^{\mathcal{N}(t)}$ represents its storage capacity, computing capacity, and set of neighboring nodes at t, respectively. We define $\mathbb{U} \subset \mathbb{N}$ as the set of user terminals. Additionally, we represent the cloud content and ad provider servers by *cps* and *aps*, respectively. A link $e_{h,k} \in \mathbb{E}$ between nodes $h \in \mathbb{N}$ and $k \in \mathbb{N}$ at t is defined by $\langle e^{\gamma}_{h,k}(t), e^{\Omega}_{h,k}(t), e^{D}_{h,k}(t) \rangle$, 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 LoS, as discussed in our previous work [132]. We define $e^{\gamma}_{h,k}(t)$ as an indicator function, where $e^{\gamma}_{h,k}(t) = 1$ if $e_{h,k}$ is available at time t, and $e^{\gamma}_{h,k}(t) = 0$ otherwise. The set of links in the shortest path between nodes $h \in \mathbb{N}$ and $k \in \mathbb{N}$ at t is represented by a set of links $\mathcal{E}_{h,k}(t) \in \mathbb{E}$, obtained by *Dijkstra's shortest path* algorithm.

Content Request and Ad Insertion Model

We consider a set of content items, denoted as $\mathbb{C}$, which are initially stored on *cps* and subsequently cached in nodes closer to end users based on their requests and resource availability. Each content $c \in \mathbb{C}$ is characterized by its storage requirement c^{β} and packet size c^{L}, both measured in bits. In addition, we consider a set of ads denoted as $\mathbb{A}$, stored on *aps*. Each ad $a \in \mathbb{A}$ is defined by its display duration a^{σ} (in seconds) and computing requirement a^{μ} (in vCPUs). We consider a mid–roll ad insertion scheme [147], 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 [130].

We consider proximity–based content probability model [132] to define the popularity of a content item $c \in \mathbb{C}$ at a node $k \in \mathbb{N}$ at t, denoted as $p_k^c(t)$. We define $\mathcal{S}_{cnt}(t)$ as the set of new and ongoing content requests at t. Each request $s \in \mathcal{S}_{cnt}(t)$ is defined as $\langle s^{src}, s^{dst}, s^{cnt}, s^{B}, s^{\mathcal{D}}, s^{\psi}, s^{t,arr}, s^{\Upsilon}(t)\rangle$, where $s^{src} \in \mathbb{U}$ denotes a source user terminal connected to the local network where the request originates. $s^{dst} = cps$ is a destination node. $s^{cnt} \in \mathbb{C}$ denotes requested content item, s^{B} is the minimum data rate required to deliver the content, $s^{\mathcal{D}}$ indicates the estimated display duration, $s^{\psi} \in \{0,1\}$ is a binary variable indicating the users subscription status (1 for premium and 0 for non–premium), and $s^{t,arr}$ specifies the arrival time of the request. Furthermore, $s^{\Upsilon}(t)$ indicates whether the request s continues from time slot t to the subsequent time slot $t+1$. Mathematically,

$$s^{\Upsilon}(t) = \begin{cases} 1, & \text{if } s^{\mathcal{D}} \geq t\Delta t - -s^{t,arr} \\ 0, & \text{otherwise.} \end{cases} \tag{4.1}$$

We assume that all content requests of a specific type have the same data rate requirement. Additionally, all requests from a given user terminal for the same content type are associated with a single cache node. For mathematical simplicity, we assume that all such requests from the same user terminal share a common subscription status value. We define $\zeta_{u,s}^{c}(t)$ if request

$s \in \mathcal{S}_{cnt}(t)$ is for $c \in \mathbb{C}$ from $u \in \mathbb{U}$ at t expressed as follows.

$$\zeta^{c}_{u,s}(t) = \begin{cases} 1, & \text{if } s^{cnt} = c \text{ and } s^{src} = u \text{ at t} \\ 0, & \text{otherwise.} \end{cases} \tag{4.2}$$

We define the binary decision variable $x^{'c}_{k}(t) \in \{0,1\}$ as the content caching strategy, indicating whether $c \in \mathbb{C}$ is cached on $k \in \mathbb{N}$ at t.

$$x^{'c}_{k}(t) = \begin{cases} 1, & \text{if the content } c \text{ is cached on node } k \text{ at } t, \\ 0, & \text{Otherwise.} \end{cases} \tag{4.3}$$

The binary decision variable $z^{c}_{u,k}(t) \in \{0,1\}$ represents the user–cache association (content distribution) strategy, indicating whether cache node $k \in \mathbb{N}$ serves content requests from user terminal $u \in \mathbb{U}$ for content $c \in \mathbb{C}$ at t,

$$z^{c}_{u,k}(t) = \begin{cases} 1, & \text{if } k \text{ serves requests from } u \text{ for } c \text{ at } t. \\ 0, & \text{otherwise.} \end{cases} \tag{4.4}$$

We set $s^{dst} = k$ if $z^{c}_{u,k}(t) = 1$ for request $s \in \mathcal{S}_{cnt}(t)$ for content $c \in \mathbb{C}$ is generated from user terminal $u \in \mathbb{U}$.

Furthermore, we define a binary decision variable $z^{'c}_{u,a}(t) \in \{0,1\}$ as the DAI strategy, indicating whether ad $a \in \mathbb{A}$ is inserted into the content requests for content $c \in \mathbb{C}$ and requested via the user terminal $u \in \mathbb{U}$ during t.

$$z^{'c}_{u,a}(t) = \begin{cases} 1, & \text{if } a \text{ is inserted into requests for } c, \text{ via } u \text{ at } t, \\ 0, & \text{otherwise.} \end{cases} \tag{4.5}$$

Content Delivery Delay Model

We consider that the content delivery delay consists of propagation, transmission, and queuing delays along the shortest path from the source user terminal to the cache node.

- Propagation Delay

The propagation delay for request $s \in \mathcal{S}_{cnt}(t)$ for content $c \in \mathbb{C}$, from user terminal

$u \in \mathbb{U}$ to cache node $k \in \mathbb{N}$ at t is:

$$d_{s,k}^{prop}(t) = \sum_{e_{m,n} \in \mathcal{E}_{u,v}(t)} e_{m,n}^{D}(t) z_{u,k}^{c}(t), \tag{4.6}$$

where $\mathcal{E}_{u,v}(t)$ represents the set of links that form the shortest path between the selected cache node k and the source user terminal node u 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 c over a link $e_{h,k}$ at t is $\frac{c^L}{e_{h,k}^{\Omega}(t)}$, where c^L is the packet size in *bits* and $e_{h,k}^{\Omega}(t)$ is the link bandwidth in $\frac{bits}{second}$. The transmission delay for request $s \in \mathcal{S}_{cnt}(t)$ for content $c \in \mathbb{C}$, from user terminal $u \in \mathbb{U}$ to cache node $k \in \mathbb{N}$, is determined by the link with the minimum bandwidth along the shortest path. Thus, the transmission delay $d_{s,k}^{txt}(t)$ is given by:

$$d_{s,k}^{txt}(t) = \frac{c^L}{\min\limits_{e_{m,n} \in \mathcal{E}_{u,k}(t)} (e_{m,n}^{\Omega}(t))} z_{u,k}^{c}(t), \tag{4.7}$$

- 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 $\lambda_k(t)$ represent the arrival rate in $\frac{requests}{millisecond}$ and $\mu_k(t)$ the service rate in $\frac{requests}{millisecond}$ at cache node $k \in \mathbb{N}$. To simplify, we model the queuing delay as the mean waiting time for a request $s \in \mathcal{S}_{cnt}(t)$ at $k \in \mathbb{N}$ for content item $c \in \mathbb{C}$ requested via terminal $u \in \mathbb{U}$ at t, denoted as $d_{s,k}^{qq}(t)$.

$$d_{s,k}^{qq}(t) = \frac{\rho_k(t)}{\mu_k(t)(1 - \rho_k(t))} z_{u,k}^{c}(t), \tag{4.8}$$

where $\rho_k(t) = \frac{\lambda_k(t)}{\mu_k(t)}$ denotes the utilization of cache k at t.

The overall content request $s \in \mathcal{S}_{cnt}(t)$ delivery delay $d_{s,k}^{\text{tot}}(t)$ is defined as:

$$d_{s,k}^{\text{tot}}(t) = d_{s,k}^{prop}(t) + d_{s,k}^{txt}(t) + d_{s,k}^{qq}(t). \tag{4.9}$$

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 [148], where pricing depends on data consumption. The service subscription revenue for a content request $s \in \mathcal{S}_{cnt}(t)$, requesting content $c \in \mathbb{C}$ via user terminal $u \in \mathbb{U}$, given by:

$$\mathcal{R}_s^{sub}(t) = c^R c^\beta z_{u,k}^c(t), \tag{4.10}$$

where c^R is the subscription price of content item c per bit. Additionally, we adopt the Cost–per–Impression ad revenue strategy [149], which advertisers are charged based on the number of ad impressions. The advertisement monetization for a content request $s \in \mathcal{S}_{cnt}(t)$, which requests content $c \in \mathbb{C}$ via user terminal $u \in \mathbb{U}$, is expressed as follows:

$$\mathcal{R}_s^{adv} = \sum_{a \in \mathbb{A}} a^R z'^c_{u,a}(t), \tag{4.11}$$

where a^R is the price per impression for ad item a.

We model the probability of user disengagement using the *Kaplan–Meier estimator* [150] and the exponentially decaying survival function [151], which increases exponentially with sensitivity factors such as content type, user demographies, ad–content language misalignment, and ad runtime, where low values indicate minimal disengagement. The disengagement cost for the content provider for a request $s \in \mathcal{S}_{cnt}(t)$ for $c \in \mathbb{C}$ from $u \in \mathbb{U}$ at t is expressed as:

$$\mathcal{R}_s^{dis}(t) = c^R c^\beta \left(1 - \sum_{a \in \mathbb{A}} e^{-\kappa_s^0 \left(\kappa_s^t + \kappa_s^d + \kappa_s^m + \frac{a^\sigma}{\Delta t}\right)} z'^c_{u,a}(t)\right) \tag{4.12}$$

where κ_s^0 is a disengagement rate, and κ_s^t, κ_s^d, and κ_s^m represent factors related to content type, user demographics, and ad–content language misalignment, respectively. This model quantifies the revenue loss for content providers due to user disengagement. We define $s^\eta(t)$

as a binary variable that indicates if a request $s \in \mathcal{S}_{cnt}(t)$ is disengaged or not.

$$s^{\eta}(t) = \begin{cases} 1, & \text{if} \sum_{a \in \mathbb{A}} e^{-\kappa_s^0 \left(\kappa_s^t + \kappa_s^d + \kappa_s^m + \frac{a^{\sigma}}{\Delta t}\right)} z'^{c}_{u,a}(t) \geq 1, \\ 0, & \text{otherwise.} \end{cases} \tag{4.13}$$

The total net revenue for delivering a request $s \in \mathcal{S}_{cnt}(t)$, which seeks content $c \in \mathbb{C}$ from user terminal $u \in \mathbb{U}$ at time t, can be derived from Equations (4.10), (4.11), and (4.12), and is given by:

$$\mathcal{R}_s^{tot}(t) = \mathcal{R}_s^{sub}(t) + \mathcal{R}_s^{adv}(t) - \mathcal{R}_s^{dis}(t) \tag{4.14}$$

The content delivery cost represents the revenue penalty for distributing content to the end user. For request $s \in \mathcal{S}_{cnt}(t)$, requesting for content $c \in \mathbb{C}$ from user terminal $u \in \mathbb{U}$ at t is:

$$\mathcal{R}_s^{cos}(t) = \frac{\mathcal{R}_s^{sub}(t)}{d_{s,cps}^{tot}(t)} d_{s,k}^{\text{tot}}(t), \tag{4.15}$$

where $d_{s,cps}^{tot}(t)$ is the content delivery delay from the cloud server to the end user during t (when $z_{u,k}^{c}(t) = 1 | k = cps$).

4.3.2 Problem Formulation

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

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 decision variable $x^{'}(t)$ represents the content cache placement strategy at time slot t. The optimization problem is formulated as follows:

$$
\begin{aligned}
&\mathbf{P}_1 : \max_{x^{'}(t)} \sum_{c\in\mathbb{C}} \sum_{k\in\mathbb{N}} p_k^c(t) \cdot x_k^{'c}(t), && (4.16a)\\
&\text{s.t.} \quad \sum_{c\in\mathbb{C}} c^{\beta} x_k^{'c}(t) \leq k^{\beta}, \quad \forall k \in \mathbb{N}, && (4.16b)\\
&\qquad \sum_{k\in\mathbb{N}} x_k^{'c}(t) \geq 1, \quad \forall c \in \mathbb{C}, && (4.16c)\\
&\qquad x_k^{'c}(t) \geq \sum_{s\in\mathcal{S}_{cnt}(t-1)} \frac{z_{u,k}^c(t-1)\cdot s^{\Upsilon}(t-1)}{|\mathcal{S}_{cnt}(t-1)|} \zeta_{u,s}^c(t-1), \forall c \in \mathbb{C}, \forall k \in \mathbb{N}, && (4.16d)\\
&\qquad \sum_{k\in\left\{k^{\mathcal{N}(t)}\cup k\right\}} x_k^{'c}(t) \leq 1, \ \forall k \in \mathbb{N}, \ \forall c \in \mathbb{C}, && (4.16e)\\
&\qquad x_k^{'c}(t) \in \{0,1\}, \quad \forall k \in \mathbb{N}, \forall c \in \mathbb{C}, && (4.16f)
\end{aligned}
$$

Constraint (4.16b) limits stored content to each cache node's capacity. Constraint (4.16c) ensures every required content item is available on at least one node, including at the cloud content server, *cps*. Constraint (4.16d) mandates that content items serving ongoing requests from the previous time slot remain cached until those requests are completed. Constraint (4.17e) ensures that only one type of content is cached on adjacent satellites, optimizing ISL and storage usage. Constraint (4.16f) enforces the binary nature of decision variables. The formulated optimization problem $\mathbf{P}_1$ is classified as an *ILP* problem.

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. The decision variables $z(t)$ and $z^{'}(t)$ represent the content distribution (i.e., user–cache association strategy) and ad insertion strategy, respectively. Mathematically,

$$\mathbf{P}_2 : \underset{z(t),\ z^{'}(t)}{\text{Maximize}} \sum_{s\in\mathcal{S}_{cnt}(t)} \Big(w_1\mathcal{R}_s^{tot}(t) - w_2\mathcal{R}_s^{cos}(t)\Big), \tag{4.17a}$$

$$\text{s.t.}\quad z_{u,k}^c(t) \leq x_k^{'c}(t), \quad \forall c \in \mathbb{C}, \forall k \in \mathbb{N}, \forall u \in \mathbb{U}, \tag{4.17b}$$

$$z_{u,a}^{'c}(t) \leq \sum_{k\in\mathbb{N}} z_{u,k}^c(t), \quad \forall a \in \mathbb{A}, \forall c \in \mathbb{C}, \forall u \in \mathbb{U}, \tag{4.17c}$$

$$\sum_{s\in\mathcal{S}_{cnt}(t)} s^B\zeta_{u,s}^c(t)z_{u,k}^c(t) \leq e_{h,k}^{\Omega}(t), \quad \forall c \in \mathbb{C}, \forall k \in \mathbb{N}, \forall u \in \mathbb{U}, \forall e_{h,k} \in \mathcal{E}_{u,v}(t), \tag{4.17d}$$

$$\sum_{k\in\mathbb{N}} z_{u,k}^c(t) \leq 1, \quad \forall c \in \mathbb{C}, \forall k \in \mathbb{N}, \forall u \in \mathbb{U}, \tag{4.17e}$$

$$z_{u,k}^c(t) \leq \frac{1}{Q_u^c(t)} \sum_{s\in\mathcal{S}_{cnt}(t)} \frac{e_{h,k}^{\gamma}(t)\Big(1 + e_{h,k}^{\gamma}(t+1)s^{\Upsilon}(t)\Big)}{1+s^{\Upsilon}(t)}\zeta_{u,s}^c(t),$$
$$\forall c \in \mathbb{C}, \forall k \in \mathbb{N}, \forall u \in \mathbb{U}, e_{h,k} \in \mathcal{E}_{u,k}(t) \tag{4.17f}$$

$$z_{u,a}^{'c}(t) \leq 1 - -\frac{1}{Q_u^c(t)} \sum_{s\in\mathcal{S}_{cnt}(t)} \zeta_{u,s}^c(t)s^{\psi}(t), \forall a \in \mathbb{A}, \forall c \in \mathbb{C}, \forall u \in \mathbb{U}, \tag{4.17g}$$

$$\sum_{a\in\mathbb{A}} z_{u,a}^{'c}(t) \leq \frac{\Delta t}{\mathcal{D}_{min}}, \quad \forall a \in \mathbb{A}, \forall c \in \mathbb{C}, \forall u \in \mathbb{U}, \tag{4.17h}$$

$$s^{\mathcal{D}}\zeta_{u,s}^c(t) \geq \mathcal{D}_{min}z_{u,a}^{'c}(t), \forall s \in \mathcal{S}_{cnt}(t), \forall a \in \mathbb{A}, \forall c \in \mathbb{C}, \forall u \in \mathbb{U}, \tag{4.17i}$$

$$z_{u,a}^{'c}(t)a^{\mu} \leq k^{\mu}z_{u,k}^c(t), \quad \forall c \in \mathbb{C}, \forall k \in \mathbb{N}, \forall u \in \mathbb{U}, \tag{4.17j}$$

$$\{z_{u,k}^c(t), z_{u,a}^{'c}(t)\} \in \{0,1\}, \quad \forall c \in \mathbb{C}, \forall k \in \mathbb{N}, \forall u \in \mathbb{U} \tag{4.17k}$$

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. $Q_u^c(t) = \sum\limits_{s\in\mathcal{S}_{cnt}(t)} \zeta_{u,s}^a(t)$ is the number of requests for content $c \in \mathbb{C}$ from user terminal $u \in \mathbb{U}$ at t. Constraint (4.17b) ensures that a user request is served only if the content is cached on the node. Constraint (4.17c) ensures that ads are inserted only into actively served requests. Constraint (4.17d) 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 (4.17e) guarantees that a request is associated with only one cache node per time slot. Constraint (4.17f) 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 (4.17g) prevents ads from displaying to premium users. Constraint (4.17h) ensures that ads per content item remain within a predefined threshold. Constraint (4.17j) ensures that ads are added only to content exceeding a minimum playback duration, $\mathcal{D}_{min}$. Constraint (4.17i) ensures that ads are inserted only if the node has sufficient processing capacity. Constraint (4.17k) states the binary decision variables. The optimization problem in Equation (4.17) (*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*.

4.4 The Proposed Scheme

We propose the Greedy Enhanced Content Caching (*GE_CC*) algorithm to solve the *ILP* content caching problem $\mathbf{P}_1$. In addition, we introduce the Greedy–based Content Distribution with DAI (*G_DAI*) and Binary Particle Swarm Optimization for Content Distribution with DAI (*BPSO_DAI*) algorithms to solve the *ILP* content distribution with DAI problem $\mathbf{P}_2$. The *G_DAI* algorithm generates a fast initial solution, which is refined by the *BPSO_DAI* algorithm for further optimization.

4.4.1 Greedy Enhanced Content Caching

The *GE_CC* algorithm optimizes service continuity–aware content cache placement for the ILP problem in Equation (4.16). Algorithm 9 outline the workflow of *GE_CC*, which efficiently determines the optimal cache placement strategy $x^{'*}(t)$ at t. The algorithm takes as input the set of network nodes $\mathbb{N}$, content items $\mathbb{C}$, content requests $\mathcal{S}_{cnt}(t)$ and $\mathcal{S}_{cnt}(t-1)$ for the current and previous time slots t and $t-1$ prior cache distribution $z(t-1)$, the current topology G_t, and the maximum number of iterations $max\_Iteration$.

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 $|\mathbb{C}|\times|\mathbb{N}|$ (*line 3*) with nodes sorted randomly (*line 4*). Here, $|\mathbb{C}|$ and $|\mathbb{N}|$ 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 (4.16d), any cached content still serving at least one request remains cached *(line 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 neigh-

Algorithm 9: Proposed *GE_CC* Algorithm

```
    Input: $\mathbb{N}$, $\mathbb{C}$, $z(t-1)$, $\mathcal{S}_{cnt}(t)$, $\mathcal{S}_{cnt}(t-1), G_t$, $max\_Iteration$
    Output: $x^{'*}(t)$: Optimal cache placement strategy at time, $t$
 1: $optimalFitness \leftarrow 0$ ▷ Initialize fitness value.
 2: while $stopConditionNotMet()$ do
 3:     $x^{'} \leftarrow 0^{|\mathbb{C}|\times|\mathbb{N}|}$ ▷ Initialize the cache placement strategy.
 4:     $\mathbb{N} \leftarrow$ Extract all node sets and randomize their order.
 5:     for each node, $k \in \mathbb{N}$ in the network do
 6:         $B_k^{av} \leftarrow k^{\beta}$ ▷ Set available storage.
 7:         for each content $c \in \mathbb{C}$ do
 8:             $x_k^{'c} \leftarrow 1$ ▷ Cache unfinished request content.
 9:             $B_k^{av} \leftarrow B_k^{av} - c^{\beta}$▷ Update storage.
10:         end for
11:         for each content item $c \in sortIDdescend(k)$ do
12:             if $notInNieghbourNode(c,k)$ and $B_k^{av} \geq c^{\beta}$ then
13:                 $x_k^{'c} \leftarrow 1$ ▷ Cache the content item.
14:                 $B_k^{av} \leftarrow B_k^{av} - c^{\beta}$▷ Update storage.
15:             end if
16:             $x^{'} \leftarrow x_k^{'c}$ ▷ Update cache placement strategy.
17:         end for
18:     end for
19:     $currentFitness \leftarrow fitness(x^{'})$ ▷ Compute current fitness using Equation (4.18).
20:     if $currentFitness \geq optimalFitness$ then
21:         $optimalFitness \leftarrow currentFitness$ ▷ Update optimal fitness.
22:         $x^{'*}(t) \leftarrow x^{'}$ ▷ Update optimal strategy.
23:     end if
24: end while
25: $x_{cps}^{'\mathbb{C}} \leftarrow 1$ ▷ Cache all content items at cloud provider.
26: return $x^{'*}(t) \leftarrow x^{'}$ ▷ Return optimal strategy.
```

boring nodes (*Constraint* (4.16e)) and storage requirements are met (*Constraint* (4.16b), *line 11-16*). At each iteration, the strategy with the highest fitness value, computed using Equation (4.16a), is selected (*lines 19-24*). To ensure content availability as per Constraint (4.16c), 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 (4.16a), which represents the system's cache hit rate and is expressed as follows:

$$fitness(x^{'}) = \mathbf{P}^T x^{'}, \tag{4.18}$$

where **P** is the content popularity matrix, and $x^{'}$ is the cache placement matrix.

Algorithm 10: Proposed *G_DAI* Algorithm

```
Input: $\mathbb{N}$, $\mathbb{C}$, $z(t-1)$, $\mathcal{S}_{cnt}(t)$, $\mathcal{S}_{cnt}(t-1), G_t$, $\mathbb{A}$
Output: $\{z^*(t), z^{'*}(t)\}$ ▷ Optimal content request-cache association and ad insertion
    strategies at $t$.
$x^{'} \leftarrow x^{'*}(t)$ ▷ Update the optimal cache placement strategy via Algorithm 9.
$z \leftarrow 0^{|\mathbb{C}|\times|\mathbb{V}|\times|\mathbb{N}|}$, $z^{'} \leftarrow 0^{|\mathbb{C}|\times|\mathbb{V}|\times|\mathbb{A}|}$ ▷ Initialize the user-cache association and ad
    insertion strategies as a zero matrix.
for each content request $s \in \mathcal{S}_{cnt}(t)$ do
    $k$ ← Select the nearest node to $u = s^{src}$ that has cached $c = s^{cnt}$ and satisfies
    Constraints (4.17b) and (4.17d)–(4.17f).
    if $(k \neq \emptyset)$ then $z^c_{u,k} \leftarrow 1$
    $\mathcal{A}$ ← Select the sets of ad contents randomly that satisfies Constraints (4.17c) and
    (4.17g)–(4.17j).
    if $(\mathcal{A} \neq \emptyset)$ then $z^c_{u,\mathcal{A}} \leftarrow 1$
    $z \leftarrow z^c_{u,k}$, $z^{'} \leftarrow z^{'c}_{u,\mathcal{A}}$
end for
return $\{z^*(t), z^{'*}(t)\} \leftarrow \{z, z^{'}\}$ ▷ Updates the optimal strategies.
```

4.4.2 Greedy-based Content Distribution with DAI

We propose the *G_DAI* algorithm to solve the ILP content distribution problem in Equation (4.17), as outlined in Algorithm 10. The algorithm takes as input set of network nodes $\mathbb{N}$, content items $\mathbb{C}$, content requests $\mathcal{S}_{cnt}(t)$ and $\mathcal{S}_{cnt}(t-1)$ for time slots t and $t-1$, the previous content cache distribution strategy $z(t-1)$, 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 Section 4.4.3.

Using Algorithm 9, Algorithm 10 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 $|\mathbb{C}|\times|\mathbb{V}|\times|\mathbb{N}|$ and $|\mathbb{C}|\times|\mathbb{V}|\times|\mathbb{A}|$, respectively *(line 2)*. For each content request at t, the algorithm uses the *Dijkstra Shortest Path* algorithm to select the nearest node to the user terminal source address that has cached the requested content item, while satisfying Constraints (4.17b) and (4.17d)–(4.17f) *(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 (4.17c) and (4.17g)–(4.17j) *(line 6)* and updates the ad insertion strategy for the selected ad content for the request s *(line 7)*. The user cache association and ad insertion strategies are then updated *(line 8)*. The process terminates when the iteration counter reaches its maximum or the best solution remains unchanged for consecutive iterations. Finally, the optimal strategy is updated *(line 10)*.

4.4.3 Binary Particle Swarm Optimization for Content Distribution with DAI

$BPSO\_DAI$ is a BPSO-based approach for optimizing ILP content distribution, offering a more optimal solution than the Greedy-based approach. Binary PSO aligns with the binary nature of decision variables, making it well-suited for this problem. Inspired by the swarm behavior, PSO explores the search space by updating particle positions and velocities to find an optimal solution [152]. We chose BPSO for simplicity, fast convergence, and proven effectiveness as a metaheuristic [153]. 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 11. It takes the following inputs $\mathbb{N}$ (set of nodes), $\mathbb{C}$ (set of content items), $z(t-1)$ (content cache distribution in previous time slot), $\mathcal{S}_{cnt}(t)$ and $\mathcal{S}_{cnt}(t-1)$ (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 11 starts by determining the optimal cache placement strategy using Algorithm 9 *(line 1)*. $BPSO\_DAI$ algorithm initializes the user-cache association and DAI strategy as zero matrices of size $|\mathbb{C}|\times|\mathbb{V}|\times|\mathbb{N}|$ and $|\mathbb{C}|\times|\mathbb{V}|\times|\mathbb{A}|$, respectively. For each content request, the algorithm randomly selects a node from the subset that has cached the requested content, ensuring compliance with Constraints (4.17b) and (4.17d)-(4.17f), and updates the user-

Algorithm 11: Proposed *BPSO_DAI* Algorithm

```
Input: $\mathbb{N}$, $\mathbb{C}$, $z(t-1)$, $\mathcal{S}_{cnt}(t)$, $\mathcal{S}_{cnt}(t-1), G_t$, $\mathbb{A}$, $w_1$, $w_2$, $w$, $c_1$, $c_2$, $\mathcal{N}_{pop}$, $\mathcal{N}_{gen}$
Output: $\{z^{*}(t), z^{'*}(t)\}$ ▷ Optimal user-cache association and DAI strategies at $t$.
 1: $x^{'} \leftarrow x^{'*}(t)$ ▷ Update the optimal cache placement strategy via Algorithm 9.
 2: Set $g \leftarrow 0$ ▷ Initialize the current iteration with 0
 3: for Each particle $i = \{1, \cdots, \mathcal{N}_{pop}\}$ do
 4:     for each content request $s \in \mathcal{S}_{cnt}(t)$ do
 5:         Initialize particle's position of user-cache association by random selection of a node
            that has the requested content and satisfying Constraints (4.17b) and
            (4.17d)–(4.17f).
 6:         Initialize particle's position of ad insertion by random selection of sets of ads that
            meet Constraints (4.17c) and (4.17g)–(4.17j).
 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 $\mathcal{U}(0, 1)$.
 9:     Update the personal best and save it as $P^{i}_{best}(g)$
10: end for
11: Replace the position of the worst performing particle by the optimal solution obtained
    from Algorithm 10.
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 $i = \{1, \cdots, \mathcal{N}_{pop}\}$ do
19:         Compute current velocity of the particle using Equations (4.20) and (4.21).
20:         Update current position of the particle using Equations (4.24) and (4.25).
21:         Update the personal best and save it as $P^{i}_{best}(g)$
22:         if $fitness$ of $P^{i}_{best}(g) \leq fitness$ of $P^{i}_{best}(g-1)$ then $P^{i}_{best}(g) \leftarrow P^{i}_{best}(g-1)$
23:         if $fitness$ of $P^{i}_{best}(g) \geq fitness$ of $G_{best}(g)$ then $G^{i}_{best} \leftarrow P^{i}_{best}(g)$
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 $q^{u} \in \mathcal{S}_{cnt}(t)$ 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 $(z^{'*}(t), z^{'*}(t)) \leftarrow G_{best}(g)$ ▷ Update the optimal solution with the global best
    position.
```

cache association accordingly *(line 5)*. Similarly, the algorithm randomly selects a subset of ad content items that satisfy Constraints (4.17c) and (4.17g)–(4.17j), 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 as the objective function obtained based on the positions of the user-cache, $z(t)$, and ad insertion, $z^{'*}(t)$, at t, expressed as follows:

$$fitness(z(t), z^{'}(t)) = \sum_{s \in \mathcal{S}_{cnt}(t)} [w_1 \mathcal{R}_s^{tot}(t) - w_2 \mathcal{R}_s^{cos}(t)], \tag{4.19}$$

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

$$V_z^{(i)}(g+1) \leftarrow wV_z^{(i)}(g) + c_1 r_1 (P_{best,i,z} - z^{(i)}(g)) + c_2 r_2 (Gbest_z - z^{(i)}(g)), \tag{4.20}$$

$$V_{z^{'}}^{(i)}(g+1) \leftarrow wV_{z^{'}}^{(i)}(g) + c_1 r_1 (P_{best,i,z^{'}} - z^{'(i)}(g)) + c_2 r_2 (Gbest_{z^{'}} - z^{'(i)}(g)), \tag{4.21}$$

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. $P_{best,i,z}$ and $P_{best,i,z^{'}}$ are the personal best positions of particle i for the user-cache association and DAI strategies, respectively, $Gbest_z$ and $Gbest_{z^{'}}$ are the global best positions for these strategies. $z^{(i)}$ and $z^{'(i)}$ represent the current positions of particle i. 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:

$$S_z(V_z^{(i)}(g+1)) = \frac{1}{1 + e^{-V_z^{(i)}(g+1)}}, \tag{4.22}$$

$$S_{z^{'}}(V_{z^{'}}^{(i)}(g+1)) = \frac{1}{1 + e^{-V_{z^{'}}^{(i)}(g+1)}}, \tag{4.23}$$

The algorithm updates the positions of the user-cache association and DAI strategies using Equations (4.24) and (4.25), respectively.

$$z^{(i)}(g+1) = \begin{cases} 1, & \text{if } r \leq S_z(V_z^{(i)}(g+1)), \\ 0, & \text{otherwise.} \end{cases} \tag{4.24}$$

$$z'^{(i)}(g+1) = \begin{cases} 1, & \text{if } r \leq S_{z'}(V_{z'}^{(i)}(g+1)), \\ 0, & \text{otherwise.} \end{cases} \tag{4.25}$$

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 *(line 22)*. Similarly, the global best is updated if a particle outperforms the current global best position of the swarm *(line 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 *(line 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.

4.5 Performance Evaluation

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

4.5.1 Simulation Settings

We use the CelesTrak dataset [122] 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 [154], along with four Iridium ground stations for the LEO satellites situated in Tempe, Arizona, Fairbanks, Alaska, Svalbard, Norway, and Punta Arenas, Chile [155]. Additionally, we use the *2018 US ZIP Code Geolocation* dataset [123] to map user terminal locations generating content requests. This dataset comprises approximately $40,000$ ZIP codes with corresponding geographical coordinates (*i.e.,* 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 [156]. The simulation models a satellite constellation with the gateways and user terminals over *24 hours* (*Jan. 01, 2025*) using the *Satellite Communications Toolbox* in *MATLAB*.

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. Additionally, we model the popularity of generated content requests using a Zipf distribution with a variable exponent $[0.2, 0.8]$ to capture dynamic content popularity, ensuring adaptability to changing request patterns over time. Additionally, we consider the proximity-based content probability model [157] to define the popularity of content item c at cache node k during time slot t. We consider 200 content items initially stored at a cloud-based content provider (cps) 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 the 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_s^0, the disengagement rate, and κ_s^t, κ_s^d, and κ_s^m, 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 Standard Definition (*SD*), High Definition (*HD*), and *4K* resolutions to determine the content display duration. The main simulation parameters are listed in Table 4.1.

Table 4.1: Simulation parameters for on-board content caching and distribution

Parameters	**Value**
Satellite constellation simulation duration	24 hours
Time slot duration	15 minutes
Number of time slots	24
Recurrence period	6 hours
Number of LEO satellites	50 [122]
Number of GEO satellites	3 [122]
Cache node storage capacity	[5-25] GB
User-LEO latency	[20 - 30] ms [158]
LEO-GEO latency	[110- 120] [158]
User-GEO	[120 - 140] ms [158]
User-cloud latency	[100 - 150] ms
Zipf exponent δ	$[0.2, 0.8]$ [131]
Feeder uplink capacity	1 Gbps [77]
Data rate requirement	{3Mbps, 8Mbps, 25Mbps} [159]
Content size	[1 - 2] GB
Number of user terminals	100
Number of content items	200
Number of ad content items	100
Ad runtime duration	[3 - 30] seconds
Ad price per impression	[20 - 10] $ [160]
Content item pricing	{ 0.08, 0.11, 0.12 } $ per GB [161]
Ad insertion processing requirement	[1 - 4] vCPU [157]
Node processing capacity	[1 - 16] vCPU [157]

4.5.2 Benchmark Scheme for On-board Content Caching and Distribution

To evaluate the proposed algorithm *BPSO_DAI*, we compare it against three commonly used cache replacement policies [124]: (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 the Density-based Content Distribution (*DCD*) scheme [110] as another benchmark. While *DCD* minimizes content delivery delay, it does not consider content requests transitioning between time slots, which our proposed algorithms address. Moreover, *DCD* lacks proximity-based content request-cache association in cases of cache misses or unavailable user-cache associations, instead directly forwarding such requests to the cloud provider. In contrast, *BPSO_DAI* prioritizes proximity-based cache association, enabling a more efficient, distributed approach through an online process. Our algorithm also integrates revenue generation through DAI, a feature absent in *DCD*. In addition, *DCD* employs a content caching scheme based on the degree of satellite nodes-the more connections a satellite has, the higher its likelihood of being selected for caching content. In contrast, our proposed solution relies on content popularity while efficiently utilizing neighboring resources via ISLs.

4.5.3 KPIs for On-board Content Caching and Distribution

- ***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:

$$CHR(t) = \frac{1}{|\mathcal{S}_{cnt}(t)|} \sum_{i=1}^{|\mathcal{S}_{cnt}(t)|} \sum_{f=1}^{|\mathbb{C}|} \sum_{k=1, n \neq cps}^{|\mathbb{N}|} \sum_{v=1}^{|\mathbb{U}|} \zeta_{u,s}^{c}(t) z_{u,k}^{c}(t). \tag{4.26}$$

- ***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:

$$R(t) = \frac{1}{|\mathcal{S}_{cnt}(t)|} \sum_{i=1}^{|\mathcal{S}_{cnt}(t)|} \Big(\mathcal{R}_s^{sub}(t) + \mathcal{R}_s^{adv,a} - \mathcal{R}_s^{dis}(t) \Big). \tag{4.27}$$

- ***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:

$$CDD(t) = \frac{1}{|\mathcal{S}_{cnt}(t)|} \sum_{i=1}^{|\mathcal{S}_{cnt}(t)|} \sum_{f=1}^{|\mathbb{C}|} \sum_{k=1}^{|\mathbb{N}|} \sum_{e_{h,k} \in \mathcal{E}_{u,v}(t)} e_{h,k}^{D}(t) \zeta_{u,s}^{c}(t) z_{u,k}^{c}(t). \tag{4.28}$$

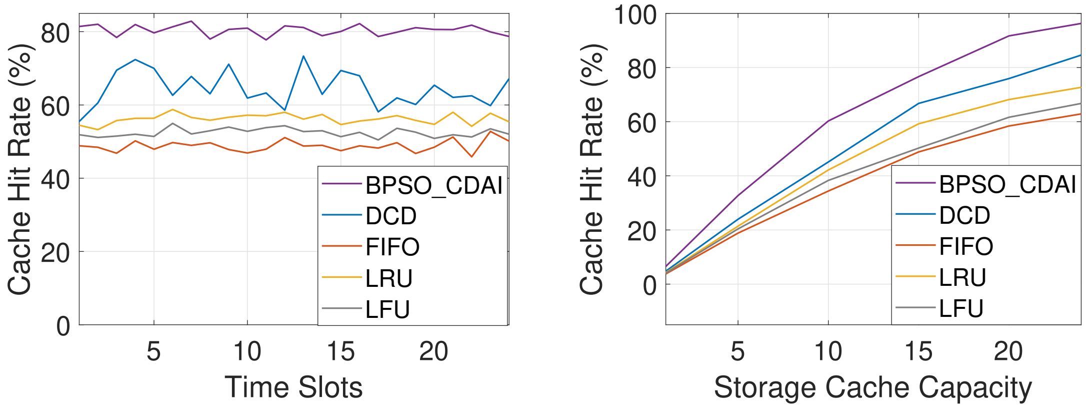
- ***Service Disruption Ratio:*** This metric measures the incompleteness of user requests due to two factors – (1) when a request continues into the next time slot, and the cached content is evicted due to a cache update and (2) when a user terminates a service because ad duration or frequency exceeds 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 servicesdue to either cache eviction or user disengagementto the total number of service requests in a given time slot, including active requests:

$$SDR(t) = \frac{1}{|\mathcal{S}_{cnt}(t)|} \sum_{i=1}^{|\mathcal{S}_{cnt}(t)|} \sum_{k=1}^{|\mathbb{N}|} |x_k^{'c}(t) - x_k^{'c}(t-1)| s^{\Upsilon}(t-1) \zeta_{u,s}^{c}(t) + s^{\eta}(t). \tag{4.29}$$

4.5.4 Results and Discussion

Cache Hit Rate

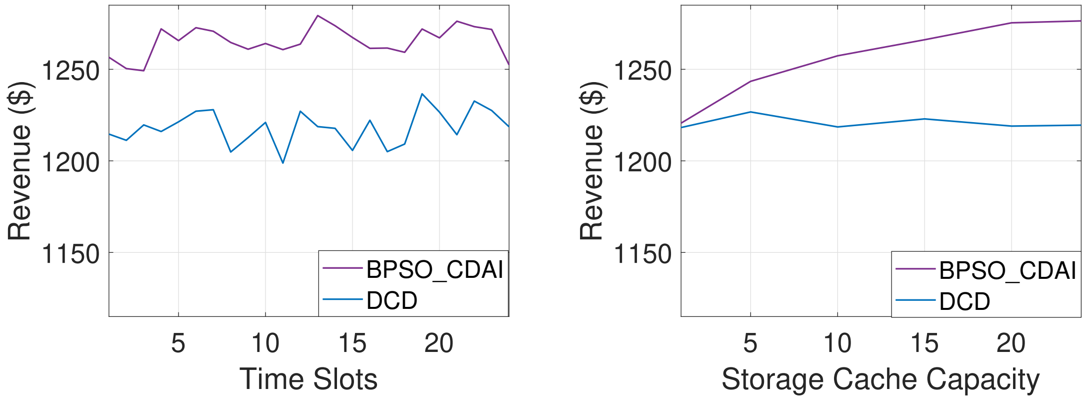
As illustrated in Figure 4.2a, the proposed solution, $BPSO\_DAI$, outperforms benchmarks in cache hit rate, achieving average improvements of 15%, 20%, 24%, and 27% over DCD, LRU, LFU, and $FIFO$, respectively, across all time slots. Figure 4.2b shows that the cache hit rate increases with cache size across all algorithms. A larger cache allows storing more content items, thereby improving the cache hit rate. For instance, the proposed solution outperforms the benchmark algorithms–DCD, LRU, LFU, and $FIFO$ by more than 10%, 15%, 20%, and 30%, respectively, when the cache size per node is 20GB. This performance gain stems from $BPSO\_DAI$'s ability to utilize more cache nodes, unlike DCD, which restricts caching based on satellite degree, reducing availability. Additionally, the proposed content popularity–based caching enhances resource utilization. While the cache hit rate fluctuates due to link availability and network topology, $BPSO\_DAI$ remains more stable than DCD, which lacks service continuity and topology awareness, causing higher fluctuations. In addition, unlike DCD, the proposed $BPSO\_DAI$ approach accounts for service continuity, allowing it to accept more content requests. In DCD, content requests that persist across time slots are considered cache misses if the content is not cached in the corresponding time slots, leading to a lower cache hit rate. By contrast, $BPSO\_DAI$ effectively handles such requests, resulting in a significantly higher cache hit rate. The main reason the proposed solution outperforms LRU, LFU, and $FIFO$ is that $BPSO\_DAI$, unlike LRU, LFU, and $FIFO$, follows a content popularity–based caching scheme, which prioritizes storing more popular content requests, thereby improving the 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 satellite cache capacity.

Figure 4.2: Average cache hit rate for different on–board content caching and distribution schemes.



(a) With varying time slots.

(b) With varying storage cache capacity.

Figure 4.3: Average content provider revenue for different on–board content caching and distribution schemes.

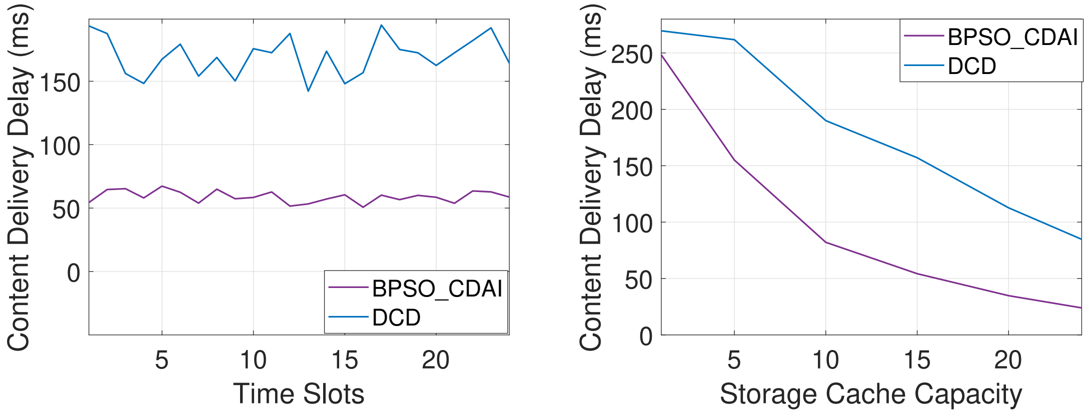
Content Provider Revenue

Figure 4.3 compares the average revenue generated by the content provider from content delivery and monetized advertising for the proposed solution and the benchmark DCD. Figure 4.3a shows that $BPSO\_DAI$ outperforms DCD by an average of more than 50$ in 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 DCD are attributed solely to statistical variations in content requests. As shown in Figure 4.3b, 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 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 DCD 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 4.3b, $BPSO\_DAI$ achieves an average revenue gain of more than 60$ compared to DCD with a cache capacity of 25 GB per node.

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.

Content Delivery Delay

Figure 4.4 presents the average content cache delivery delay, comparing the proposed solution with DCD. As shown in Figure 4.4a, $BPSO\_DAI$ achieves an average content delivery delay of 51 ms, significantly outperforming DCD, which experiences delays exceeding 150



(a) With varying time slots.

(b) With varying satellite cache capacity.

Figure 4.4: Average content delivery delay for different on–board content caching and distribution schemes.

ms across the time slots. Additionally, Figure 4.4b illustrates that the content delivery delay decreases as cache storage size increases in both $BPSO\_DAI$ and DCD. 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 4.4b, $BPSO\_DAI$ achieves a significantly lower content delivery delay than DCD, reducing the delay by 95 ms at a cache storage size of 25 GB. This improvement is attributed to the consideration of service continuity, which minimizes disruptions and dynamically routes requests to optimal nearby caches. In contrast, DCD restricts caching decisions based on node degree, which limits its flexibility in handling content requests and leads to higher delays. Moreover, the proposed solution exhibits more stable content delivery performance since DCD relies heavily on a dynamically changing node degree in a time–varying topology, introducing fluctuations in content delivery delay.

Service Disruption Ratio

Figure 4.5 compares the service disruption ratio of the proposed $BPSO\_DAI$ solution with the benchmark DCD over the first 5 time slots, with 50% of content requests continuing to

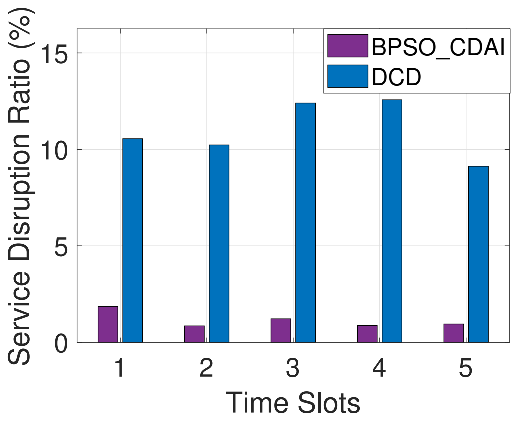


Figure 4.5: Service disruption ratio.

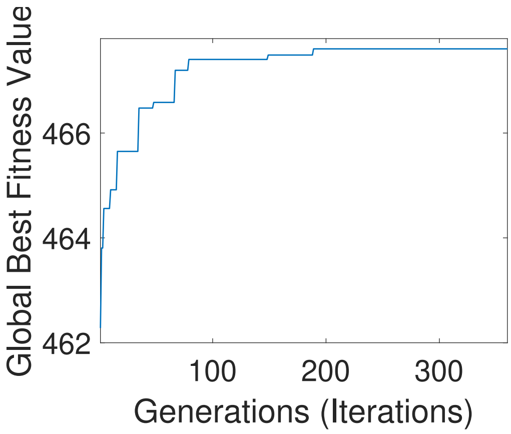
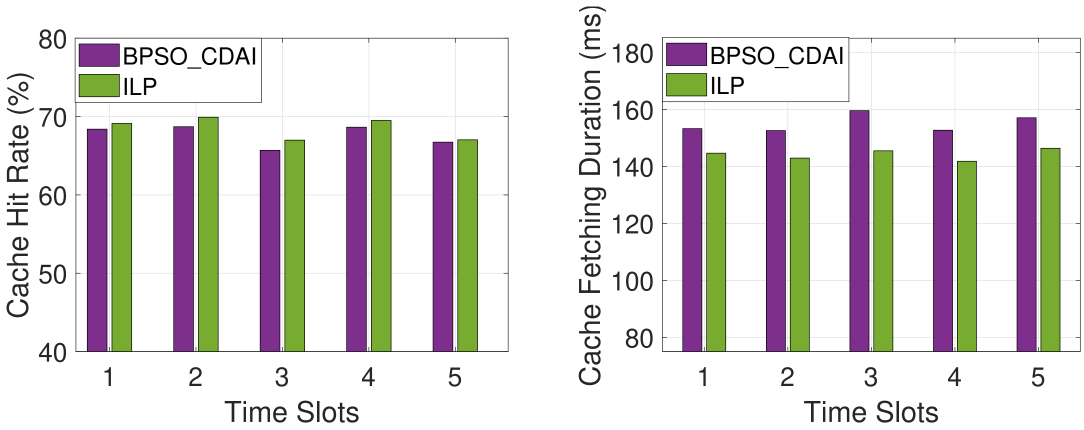


Figure 4.6: The convergence process of the proposed solution.



(a) Cache hit rate.

(b) Content delivery delay.

Figure 4.7: Performance comparison with optimal solution.

the next time slot. The proposed solution shows significantly lower disruption, with less than 2% of content requests disrupted, whereas *DCD* experiences over 10%. This improvement results from the topology–aware approach in *BPSO_DAI*, which ensures service continuity in time–varying topologies. However, a small fraction of content requests still face disruption due to resource limitations and user disengagement from ad insertions. Figure 4.6 shows that the *BPSO_DAI* converges in under 200 iterations.

Comparison of *BPSO_DAI* with the Optimal ILP

This section evaluates the optimality gap of the proposed *BPSO_DAI* and the optimal *ILP* solution. The *ILP* model formulated in Equations (4.16) and (4.17) was solved using the *CVX MOSEK* optimization solver [125] 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 4.7 illustrates the penalty gap between the proposed and the optimal *ILP* solutions. As shown in Figure 4.7a, $BPSO\_DAI$ achieves a maximum of 1.3% optimality gap in cache hit rate, demonstrating performance close to *ILP*. Similarly, Figure 4.7b 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 user terminals. 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.

4.6 Conclusion

In conclusion, this chapter 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 over a 5% increase in revenue and reduces cache retrieval duration by 33% compared to the benchmark DCD.

Chapter 5

Conclusion and Future Work

5.1 Conclusion

This thesis presents service provisioning in SEC-enabled satellite networks for next-generation applications through two core dimensions that address distinct use cases. The first part addresses the computing functionality of SEC, with a focus on VNF mapping and scheduling to efficiently process VNF components and complete SFC service requests for mission-critical applications. The second part focuses on content caching, distribution, and ad monetization, leveraging the storage functionality of SEC to enhance content delivery efficiency. We consider several challenges, such as capacity constraints, satellite topology dynamism, traffic demands, and heterogeneity of service requirements. In this thesis, we optimize specific aspects of network performance, user experiences, and service provider profits for different next-generation applications and use cases that range from mission-critical applications to VoD over OTT services. Furthermore, we consider every user profile with user distribution, and demand distribution is non-uniform. Dynamically changing nonuniform user demand, heterogeneity, and network topology force future service provisioning scheme techniques to adapt to the network's dynamic change.

In Chapter 2, we presented a VNF mapping and scheduling scheme for mission-critical applications, aiming to reduce E2E service delay and ensure fairness in service delay margins among competing services. The proposed approach meets the stringent requirements of mission-critical applications by leveraging VNF components of SFC service requests in orbit to facilitate and complete services with low latency while ensuring ubiquitous delivery enabled by the global coverage of satellite networks. We proposed a topology-aware SA-based algorithm that dynamically optimizes VNF mapping and scheduling in response to changing network topology and traffic demands, considering resource availability for seamless service completion. We compare the proposed approach with relevant benchmarks and a Greedy-based solution, which we designed primarily for low service arrival rates and small network sizes.

In Chapter 3, we proposed a novel on-board content cache reconfiguration overhead-aware placement strategy for SEC-enabled multi-layer satellite networks. Furthermore, we developed an on-board content caching algorithm that optimizes storage resource utilization through a novel content popularity model. The algorithm also includes a cache-to-cache update scheme to minimize cache reconfiguration overhead while ensuring content relevance. Cache reconfiguration overhead is defined in terms of content cache placement delay and feeder link load. This approach meets end-user requirements while adapting to dynamic content traffic demands and network topology to meet end-user requirements.

Chapter 4 presented a revenue-driven seamless content distribution framework with a DAI scheme in STINs. We focused on maximizing content provider revenue through content delivery and ad monetization while addressing key service disruption challenges. Specifically, we identified two primary causes of service disruption: (i) dynamic network topology, where links appear and disappear unpredictably, and (ii) excessive ad insertion, which can lead to user disengagement. To mitigate these challenges, we proposed a service continuity-aware content caching and distribution approach to prevent disruptions caused by network dynamics. Additionally, we introduced an ad user disengagement cost-aware scheme to balance monetization and user retention. Finally, we developed a BPSO-based joint content caching and distribution algorithm with a DAI mechanism, enabling collaborative cache node management to handle service requests efficiently. The proposed solution also accounts for dynamic content traffic patterns and network topology variations. To assess its effectiveness, we compared our approach against relevant benchmarks.

5.2 Future Work

The results presented in this thesis have shown the potential of the proposed service provisioning in SEC-enabled networks for next-generation applications. However, there are still many opportunities to extend its scope. This section discusses potential extensions of the thesis for future research.

- **Toward 3GPP-Compliant SEC – Enabling VNF-Based Service Provisioning in 5G/6G Networks**: In the latest technical specification release of 3GPP (Release 19) [162], support for regenerative payload architectures and enhanced compatibility with 5G Core (5GC) functions in NTNs has been introduced. A key future direction emerging from this development is the integration of 3GPP-compliant network functions within the SEC framework. The SEC framework can enable ubiquitous connectivity by orchestrating and provisioning services in orbit using core network components such as the gNB, User Plane Function (UPF), and Access and Mobility Management Function (AMF). This advancement presents significant opportunities for next-generation networks (5G/6G) and constitutes one of the potential extensions of this thesis. Integrating 3GPP functions into the SEC architecture ensures interoperability with future terrestrial networks and enhances the practical feasibility and deployment potential of SEC-enabled STINs.

- **Intelligent Virtual Network Embedding (VNE) in SEC-Enabled Networks**: Another promising direction for extending the VNF mapping and scheduling framework in SEC-enabled networks is incorporating Artificial Intelligence (AI) to enable predictive and intelligent decision-making during service provisioning. While this work primarily focuses on point-to-point SFC requests, real-world service demands are often more complex. These may involve multi-point-to-multi-point, multi-point-to-single-point, or other non-linear service topologies, where the set of requested VNFs and their interconnections form arbitrary graphs rather than simple chains. Future work should explore generic and intelligent VNE approaches to address such scenarios, leveraging AI/ML techniques to predict resource demand, optimize VNF placement under constraints, and dynamically adapt to varying service topologies. Integrating AI-driven VNE solutions within the SEC framework will enhance flexibility and scalability and improve the resilience and efficiency of service provisioning in complex and dynamic satellite networks.

- **Enhancing Security and Privacy in Content Distribution and Ad Insertion:** Security and privacy remain critical challenges in content caching, distribution, and DAI components of SEC-enabled networks. A promising future direction involves the integration of quantum communication technologies, particularly Quantum Key Distribution (QKD) [163], to ensure secure content delivery and protect against advanced cyber threats. Leveraging quantum-safe mechanisms within satellite-based content distribution systems can uphold end-to-end confidentiality and integrity, even under evolving threat landscapes. While this thesis introduces a distributed ad insertion scheme that inherently reduces privacy exposure compared to centralized approaches such as SSAI, strict compliance with data privacy regulations, most notably the GDPR, remains essential. Future research should explore privacy-preserving ad personalization techniques that uphold legal and ethical standards while supporting monetization goals. Key directions include edge-based anonymization, federated learning for decentralized user profiling, and user-consent-aware content adaptation. These approaches balance business objectives and robust privacy protections, marking an essential evolution of SEC-enabled content services.

- **AI-Driven Content Caching and Distribution with Proactive User Disengagement Handling**: Another promising extension of this research lies in developing an intelligent content caching and distribution framework capable of effectively handling service requests, particularly during cache update windows when content availability may be temporarily reduced. Integrating AI into the caching and distribution pipeline enables predictive and adaptive decision-making for content placement, update scheduling, and user association based on historical data and behavioral patterns. This direction is particularly valuable in addressing user disengagement costs during content delivery, a critical concern in DAI scenarios where poorly timed or irrelevant content may lead to user drop-off. An AI-driven caching and DAI system can proactively adapt to user preferences, contextual demand, and network dynamics to minimize disengagement, enhance QoE, and maximize monetization opportunities. By intelligently balancing content freshness, user behavior, and ad relevance, such a system would support more resilient, adaptive, and user-centric content delivery across satellite edge networks, paving the way for scalable and sustainable deployment in next-generation network infrastructures.

- **Orchestrator Placement for VNF Mapping and Content Caching in SEC**: Another valuable future research direction involves identifying and optimizing the placement of orchestration entities, such as SDN controllers, which coordinate both VNF mapping and scheduling and content caching and distribution decisions. The location and responsiveness of these orchestrators significantly impact service completion delay and the overall quality and correctness of the deployed service chains. Poor placement or delayed communication with the orchestrator may result in inefficient VNF deployments, increased latency, and even service failure in mission-critical scenarios. Similarly, in content caching systems, the placement of the orchestrator responsible for cache placement and updates is equally critical. Since this entity determines where and when content is cached or updated, its proximity to the nodes and decision latency directly affect the timeliness of cache updates and cache hit performance. Future research should, therefore, explore dynamic, hierarchical, or distributed placement strategies for orchestrators in SEC-enabled satellite networks, with optimization objectives that include delay minimization, load balancing, and resilience to link failures or congestion.

Appendix A

E2E Service Delay Model

The E2E service delay is the cumulative time taken to complete a service request via the selected physical path. The E2E service delay $d_s^{tot}(t)$ for a specific service request s at a given snapshot time t, is defined as follows:

$$d_s^{tot}(t) = \sum_{f_{s,j} \in s^{vnf}} \left(\sum_{k \in \mathbb{N}} \Big(\sum_{h \in \mathbb{N}} e^D_{h,k}(t)\ y^{s,j}_{h,k}(t) \Big) + f^{\ell,k}_{s,j}\ x^{s,j}_k(t) + f^{\tau,k}_{s,j} \right), \tag{1}$$

The E2E service delay, $d_s^{tot}(t)$, consists of three components.

1. The propagation delay of a service request includes the cumulative sum of the propagation delays of the physical links along the selected path required to fulfill the service request. The first component of the service delay $d_s^{tot}(t)$ for the service request s is the propagation delay $d_s^{prop}(t)$, defined as follows:

$$d_s^{prop}(t) = \sum_{f_{s,j} \in s^{vnf}} \left(\sum_{k \in \mathbb{N}} \Big(\sum_{h \in \mathbb{N}} y^{s,j}_{h,k}(t) e^D_{h,k}(t) \Big) \right) \tag{2}$$

   where $d_{h,k}(t)$ represents the propagation delay between the physical nodes h and k, and $y^{s,j}_{h,k}(t)$ is the virtual link decision variable indicating whether the virtual link between VNF $f_{s,j}$ and its preceding VNF $f_{s,j-1}$ is mapped onto the path containing the link $e_{h,k}$ at the given snapshot t, respectively.

2. The second component, termed processing delay $d_s^{proc}(t)$ within $d_s^{tot}(t)$, signifies the cumulative processing delay of the VNF components of s at their designated nodes. It is defined as:

$$d_s^{proc}(t) = \sum_{f_{s,j} \in s^{vnf}} f^{\ell,k}_{s,j}\ x^{s,j}_k(t) \tag{3}$$

Where $f^{\ell,k}_{s,j}$ stands for the processing delay necessary to handle the VNF component $f_{s,j}$ at node k, and $x^{s,j}_{k}(t)$ signifies the VNF mapping decision determining whether VNF component $f_{s,j}$ is executed at node k during the specified snapshot t.

3. We can define the queuing delay of a VNF as the total delay experienced by the VNF while it awaits processing at a designated node. This delay arises when the designated node is occupied with processing another VNF. The third component of $d^{tot}_{s}(t)$ is the total queuing delay for VNF $f_{s,j}$ at node k, denoted as $f^{\tau,k}_{s,j}$, and can be calculated as follows:

$$\begin{aligned} f^{\tau,k}_{s,j} = x^{s,j}_{k}(t) \sum_{g \in \boldsymbol{S}_{sfc}(t)} \sum_{f_{g,q} \in g^{vnf}} \Bigg(& \Big(f^{\ell,k}_{s,j} - (f^{A,k}_{g,q}(t) - f^{A,k}_{s,j}(t)) \Big) \theta^{k}_{sj,gq} \chi^{k}_{sj,gq}(t) \\ & + \Big(f^{\ell,k}_{g,q} - (f^{A,k}_{s,j}(t) - f^{A,k}_{g,q}(t)) \Big) \theta^{k}_{gq,sj} \chi^{k}_{gq,sj}(t) \Bigg), \forall \{g \neq s \ OR \ q \neq j\} \end{aligned} \tag{4}$$

Where $\theta^{k}_{sj,gq}$ is a binary variable that shows whether VNF $f_{s,j}$ arrives while VNF $f_{g,q}$ waits for processing or is in the middle of processing at node k and can be described as follows.

$$\theta^{k}_{sj,gq} = \begin{cases} 1, & \text{if } f_{s,j} \text{ arrived while } f_{g,q} \text{ at } k \\ 0, & \text{Otherwise.} \end{cases}$$

From Equation (1), the E2E delay of a request can be categorized into two main components: VNF mapping delay component and VNF scheduling delay component.

The delay associated with VNF mapping encompasses the processing delay and propagation delay components, and it can be defined as follows:

$$d^{vm}_{s}(t) = \sum_{f_{s,j} \in s^{vnf}} \Bigg(\sum_{k \in \mathbb{N}} \Big(\sum_{h \in \mathbb{N}} y^{s,j}_{h,k}(t) e^{D}_{h,k}(t) \Big) + x^{s,j}_{k}(t) f^{\ell,k}_{s,j} \Bigg) \tag{5}$$

The delay attributed to VNF scheduling encompasses the cumulative queuing delay incurred by the VNF components within a service request. This delay can be defined as:

$$d^{vs}_{s}(t) = \sum_{f_{s,j} \in s^{vnf}} f^{\tau,k}_{s,j} \tag{6}$$

Appendix B

Binary Relaxation

Note that we define $X = x_k^{s,j}(t)$, $Y = y_{h,k}^{s,j}(t)$ and $\chi = \chi_{sj,gq}^{k}(t)$ to simplify the solution of the optimization problem. We define the VNF mapping, ψ_{xy}, and VNF scheduling, ψ_{χ} penalized functions which include the penalty functions and their corresponding parameters as:

$$\psi_{xy} \;=\; \mu_x \sum_{f_{s,j} \in s^{vnf}} \sum_{k \in \mathbb{N}} (X - X^{i-1}) \nabla \mathcal{P}(X) \;+\; \mu_y \sum_{f_{s,j} \in s^{vnf}} \sum_{h \in \mathbb{N}} \sum_{k \in \mathbb{N}} (Y - Y^{i-1}) \nabla \mathcal{P}(Y) \quad (7)$$

$$\psi_{\chi} = \mu_{\chi} \sum_{s \in \boldsymbol{S}_{sfc}(t)} \sum_{f_{s,j} \in s^{vnf}} \sum_{g \in \boldsymbol{S}_{sfc}(t)} \sum_{f_{g,q} \in g^{vnf}} (\chi - \chi^{i-1}) \nabla \mathcal{P}(\chi) \quad (8)$$

Where $\mathcal{P}(X)$, $\mathcal{P}(Y)$, and $\mathcal{P}(\chi)$, are relaxation penalty functions associated with the decision variables for VNF node mapping (X), virtual link mapping (Y), and VNF scheduling (χ), respectively and defined as:

$$\mathcal{P}(X) = X^2 - X, \; \mathcal{P}(Y) = Y^2 - Y, \; and \; \mathcal{P}(\chi) = \chi^2 - \chi \quad (12)$$

The penalty functions are designed to encourage the relaxed problem to generate solution values within the range of 0 to 1. Additionally, we introduce corresponding penalty parameters, represented as μ_x, μ_y, and μ_{χ}, specifically for the decision variables $\mathbf{X}(t)$, $\mathbf{Y}(t)$, and $\boldsymbol{\chi}(t)$, respectively. These penalty parameters play a role in shaping the behavior of the relaxation to encourage binary-like solutions. We assume that X^{i-1} and Y^{i-1} are the $(i-1)^{th}$ VNF node and virtual link mapping solutions, respectively. Similarly, χ^{i-1} is considered the $(i-1)^{th}$ VNF scheduling solution.

Bibliography

[1] N. B. Ruparelia, *4 Cloud Native Foundations*, 2nd ed. MIT Press, 2023, pp. 61–75.

[2] N. B. Ruparelia, *1 Introduction*, 2nd ed. MIT Press, 2023, pp. 1–23.

[3] A. W. Services, “Amazon Web Services (AWS),” 2025, accessed: Apr. 2025. [Online]. Available: https://aws.amazon.com

[4] G. C. Platform, “Google Cloud Platform (GCP),” 2025, accessed: Apr. 2025. [Online]. Available: https://cloud.google.com

[5] I. Cloud, “IBM Cloud,” 2025, accessed: Apr. 2025. [Online]. Available: https://www.ibm.com/cloud

[6] M. Azure, “Microsoft Azure,” 2025, accessed: Apr. 2025. [Online]. Available: https://azure.microsoft.com

[7] X. Ning, C. Huang, C. Wang, and Y. Peng, “A Cloud Service Architecture for SDN-based Space-Terrestrial Integrated Network,” in *Wireless and Satellite Systems. WiSATS 2021*, ser. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, Q. Guo, W. Meng, M. Jia, and X. Wang, Eds., vol. 410. Springer, Cham, 2022.

[8] S. Liu, Z. Zhang, G. Han, and B. Shen, “Satellite-Air-Terrestrial Cloud Edge Collaborative Networks: Architecture, Multi-Node Task Processing and Computation,” *Intelligent Automation & Soft Computing*, vol. 37, no. 3, pp. 2651–2668, 2023.

[9] L. C. V. Mishra, “Satellite Edge Computing for Real-Time Data Processing and Connectivity,” *Defstrat*, vol. 18, no. 6, p. 40, 2025.

[10] R. Xie, Q. Tang, Q. Wang, X. Liu, F. R. Yu, and T. Huang, "Satellite-Terrestrial Integrated Edge Computing Networks: Architecture, Challenges, and Open Issues," *IEEE Network*, vol. 34, no. 3, pp. 224–231, 2020.

[11] Z. YIN, C. WU, C. GUO, Y. LI, M. XU, W. GAO, and C. CHI, "A Comprehensive Survey of Orbital Edge Computing: Systems, Applications, and Algorithms," *Chinese Journal of Aeronautics*, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1000936124004709

[12] N. Abbas, Y. Zhang, A. Taherkordi, and T. Skeie, "Mobile Edge Computing: A Survey," *IEEE Internet of Things Journal*, vol. 5, no. 1, pp. 450–465, 2018.

[13] B. Denby and B. Lucia, "Orbital Edge Computing: Nanosatellite Constellations as a New Class of Computer System," in *Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems*. ACM, 2020, pp. 939–954.

[14] C. Wu, Y. Li, M. Xu, C. Guo, Z. Yin, W. Gao, and C. Chi, "A Comprehensive Survey on Orbital Edge Computing: Systems, Applications, and Algorithms," *arXiv preprint arXiv:2306.00275*, 2023.

[15] Z. Zhang, W. Zhang, and F.-H. Tseng, "Satellite Mobile Edge Computing: Improving QoS of High-Speed Satellite-Terrestrial Networks Using Edge Computing Techniques," *IEEE Network*, vol. 33, no. 1, pp. 70–76, 2019.

[16] M. Centenaro, C. E. Costa, F. Granelli, C. Sacchi, and L. Vangelista, "A Survey on Technologies, Standards and Open Challenges in Satellite IoT," *IEEE Communications Surveys & Tutorials*, vol. 23, no. 3, pp. 1693–1720, 2021.

[17] H. Talebian, A. Gani, M. Sookhak, A. A. Abdelatif, A. Yousafzai, A. V. Vasilakos, and F. R. Yu, "Optimizing Virtual Machine Placement in IaaS Datacenters: Taxonomy, Review and Open Issues," *Cluster Computing*, vol. 23, no. 2, pp. 837–878, 2020.

[18] M. Cusumano, "Cloud Computing and SaaS as New Computing Platforms," *Communications of the ACM*, vol. 53, no. 4, pp. 27–29, 2010.

[19] C. Pahl, "Containerization and the PaaS cloud," *IEEE Cloud Computing*, vol. 2, no. 3, pp. 24–31, 2015.

[20] O. Ben Yahia, Z. Garroussi, B. Sans, J.-F. Frigon, S. Martel, A. Lesage-Landry, and G. Karabulut Kurt, “A Scalable Architecture for Future Regenerative Satellite Payloads,” *IEEE Wireless Communications Letters*, vol. 14, no. 2, pp. 514–518, 2025.

[21] Y. Wang, J. Yang, X. Guo, and Z. Qu, “Satellite Edge Computing for the Internet of Things in Aerospace,” *Sensors*, vol. 19, no. 20, p. 4375, 2019.

[22] L. Yan, S. Cao, Y. Gong, H. Han, J. Wei, Y. Zhao, and S. Yang, “SatEC: A 5G Satellite Edge Computing Framework based on Microservice Architecture,” *Sensors*, vol. 19, no. 4, p. 831, 2019.

[23] Airbus, “Satellite On-Board Computers (OBC),” 2025, accessed: Apr. 2025. [Online]. Available: https://www.satnow.com/search/on-board-computers

[24] Honeywell, “Spacecraft On-Board Computer (OBC),” 2025, accessed: Apr. 2025. [Online]. Available: https://aerospace.honeywell.com/us/en/products-and-services/products/emerging-technologies/space/space-electronics/spacecraft-on-board-computer-obc

[25] S. Strauss, “Storage in Space? Why Is It So Complicated and How Will Humanity Benefit from It?” 2022, accessed: Apr. 2025. [Online]. Available: https://www.calcalistech.com/ctech/articles/0,7340,L-3927958,00.html

[26] T. Technologies, “Satellite Orbits,” 2024, accessed: Apr. 2025. [Online]. Available: https://www.teledyne.com/digital-imaging-space-science-monthly/satellite-orbits

[27] E. Union, “Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the Protection of Natural Persons with Regard to the Processing of Personal Data and on the Free Movement of Such Data, and Repealing Directive 95/46/EC (General Data Protection Regulation),” 2016, accessed: Apr. 2025. [Online]. Available: https://eur-lex.europa.eu/eli/reg/2016/679/oj

[28] E. J. Kitindi, S. Fu, Y. Jia, A. Kabir, and Y. Wang, “Wireless Network Virtualization With SDN and C-RAN for 5G Networks: Requirements, Opportunities, and Challenges,” *IEEE Access*, vol. 5, pp. 19 099–19 115, 2017.

[29] D. Kreutz, F. M. V. Ramos, P. E. Verssimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, “Software-Defined Networking: A Comprehensive Survey,” *Proceedings of the IEEE*, vol. 103, no. 1, pp. 14–76, 2015.

[30] R. Mijumbi, J. Serrat, J.-L. Gorricho, N. Bouten, F. De Turck, and R. Boutaba, “Network Function Virtualization: State-of-the-Art and Research Challenges,” *IEEE Communications Surveys & Tutorials*, vol. 18, no. 1, pp. 236–262, 2016.

[31] O. N. Foundation, “Software-Defined Networking: The New Norm for Networks,” 2012, accessed: Apr. 2025. [Online]. Available: https://opennetworking.org/wp-content/uploads/2011/09/wp-sdn-newnorm.pdf

[32] A. Bhardwaj and C. R. Krishna, “Virtualization in Cloud Computing: Moving from Hypervisor to ContainerizationA Survey,” *Arabian Journal for Science and Engineering*, vol. 46, pp. 8585–8601, 2021. [Online]. Available: https://doi.org/10.1007/s13369-021-05553-3

[33] VMware, “VMware,” https://www.vmware.com/, 2025, accessed: Apr. 2025.

[34] Linux-KVM, “Kernel-based Virtual Machine (KVM),” https://linux-kvm.org/page/Main_Page, 2025, accessed: Apr. 2025.

[35] XEN, “Xen,” https://xenproject.org/, 2025, accessed: Apr. 2025.

[36] Microsoft A., “What Is a Virtual Machine and How Does It Work,” https://azure.microsoft.com/en-us/resources/cloud-computing-dictionary/what-is-a-virtual-machine/, 2025, accessed: Apr. 2025.

[37] Microsoft, “What Is a Container and How Does It Work,” https://azure.microsoft.com/en-us/resources/cloud-computing-dictionary/what-is-a-container/, 2025, accessed: Apr. 2025.

[38] “Docker: Accelerated Container Application Development,” https://www.docker.com/, 2025, accessed: Apr. 2025.

[39] Zabbly, “Container and Virtualization Tools,” https://linuxcontainers.org/, 2025, accessed: Apr. 2025.

[40] H. Yang, H. Pan, and L. Ma, “A Review on Software Defined Content Delivery Network: A Novel Combination of CDN and SDN,” *IEEE Access*, vol. 11, pp. 43 822–43 843, 2023.

[41] S. Yang, H. Li, Z. Lai, and J. Liu, “A Synergic Architecture for Content Distribution in Integrated Satellite and Terrestrial Networks,” in *2020 IEEE/CIC International Conference on Communications in China (ICCC)*, 2020, pp. 96–101.

[42] Ericsson, “Ericsson Mobility Report: November 2024,” 2024, accessed: July 2, 2025. [Online]. Available: https://www.ericsson.com/en/reports-and-papers/mobility-report/reports/november-2024

[43] S. Steinberg, “Business-Critical Apps Move into the Future,” *The Forecast By Nutanix*, 2023, accessed: July 2, 2025. [Online]. Available: https://www.nutanix.com/theforecastbynutanix/technology/business-critical-apps-move-into-the-future

[44] I. T. Union, “Facts and Figures 2023 - Internet Use,” 2023, accessed: July 2, 2025. [Online]. Available: https://www.itu.int/itu-d/reports/statistics/2023/10/10/ff23-internet-use/

[45] Y. Wang, J. Yang, X. Guo, and Z. Qu, “A Game-Theoretic Approach to Computation Offloading in Satellite Edge Computing,” *IEEE Access*, vol. 8, pp. 12 510–12 520, 2020.

[46] J. P. Choi and C. Joo, “Challenges for Efficient and Seamless Space-Terrestrial Heterogeneous Networks,” *IEEE Communications Magazine*, vol. 53, no. 5, pp. 156–162, 2015.

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

[48] H. Adoga and D. Pezaros, “Network Function Virtualization and Service Function Chaining Frameworks: A Comprehensive Review of Requirements, Objectives, Implementations, and Open Research Challenges,” *Future Internet*, vol. 14, no. 2, p. 59, 2022. [Online]. Available: https://doi.org/10.3390/fi14020059

[49] T. Kim, J. Kwak, and J. P. Choi, “Satellite Edge Computing Architecture and Network Slice Scheduling for IoT Support,” *IEEE Internet of Things Journal*, vol. 9, no. 16, pp. 14 938–14 951, 2022.

[50] Q. Li, S. Wang, X. Ma, Q. Sun, H. Wang, S. Cao, and F. Yang, "Service Coverage for Satellite Edge Computing," *IEEE Internet of Things Journal*, vol. 9, no. 1, pp. 695–705, 2022.

[51] Y. Qiu, J. Niu, X. Zhu, K. Zhu, Y. Yao, B. Ren, and T. Ren, "Mobile Edge Computing in Space-Air-Ground Integrated Networks: Architectures, Key Technologies and Challenges," *Journal of Sensor and Actuator Networks*, vol. 11, no. 4, p. 57, 2022.

[52] Y. Liao, K. Cheng, and H. Jin, "Towards Routing and Edge Computing in Satellite-Terrestrial Networks: A Column Generation Approach," *arXiv preprint arXiv:2502.14422*, 2025.

[53] E. S. Agency, "Types of Orbits," https://www.esa.int/Enabling_Support/Space_Transportation/Types_of_orbits, 2020, accessed: July 2, 2025.

[54] Y. Lin, W. Feng, T. Zhou, Y. Wang, Y. Chen, N. Ge, and C.-X. Wang, "Integrating Satellites and Mobile Edge Computing for 6G Wide-Area Edge Intelligence: Minimal Structures and Systematic Thinking," *IEEE Network*, vol. 37, no. 2, pp. 14–21, 2023.

[55] Y. Lin, W. Feng, Y. Wang, Y. Chen, Y. Zhu, X. Zhang, N. Ge, and Y. Gao, "Satellite-MEC Integration for 6G Internet of Things: Minimal Structures, Advances, and Prospects," *IEEE Open Journal of the Communications Society*, vol. 5, pp. 3886–3903, 2024.

[56] O. I. Now, "Orbit Ing Now," https://orbit.ing-now.com/, 2025, accessed: July 2, 2025.

[57] NanoAvionics, "How Many Satellites are in Space," https://nanoavionics.com/blog/how-many-satellites-are-in-space/, 2025, accessed: July 2, 2025.

[58] D. D. Tereza Pultarova, Adam Mann, "Starlink Satellites: Facts, Tracking and Impact on Astronomy," https://www.space.com/spacex-starlink-satellites.html, 2025, accessed: Apr. 2025.

[59] SpaceX, "Starlink Satellite Demisability," https://www.starlink.com/us/updates, 2025, accessed: Apr. 2025.

[60] R. Mijumbi, J. Serrat, J.-L. Gorricho, N. Bouten, F. De Turck, and S. Davy, "Design and Evaluation of Algorithms for Mapping and Scheduling of Virtual Network Functions,"

in *Proceedings of the 2015 1st IEEE Conference on Network Softwarization (NetSoft)*, 2015, pp. 1–9.

[61] J. Li, W. Shi, H. Wu, S. Zhang, and X. Shen, "Cost-Aware Dynamic SFC Mapping and Scheduling in SDN/NFV-Enabled Space-Air-Ground-Integrated Networks for Internet of Vehicles," *IEEE Internet of Things Journal*, vol. 9, no. 8, pp. 5824–5838, 2022.

[62] Q. T. Ngo, K. T. Phan, W. Xiang, A. Mahmood, and J. Slay, "On Edge Caching in Satellite IoT Networks," *IEEE Internet of Things Magazine*, vol. 4, no. 4, pp. 107–112, 2021.

[63] L. Li, G. Zhao, and R. S. Blum, "A Survey of Caching Techniques in Cellular Networks: Research Issues and Challenges in Content Placement and Delivery Strategies," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 3, pp. 1710–1732, 2018.

[64] T. Wang and L. Vandendorpe, "Successive Convex Approximation based Methods for Dynamic Spectrum Management," in *2012 IEEE International Conference on Communications (ICC)*, 2012, pp. 4061–4065.

[65] M. Grant and S. Boyd, "CVX: Matlab Software for Disciplined Convex Programming, Version 2.1," https://cvxr.com/cvx, Mar. 2014.

[66] X. You, C.-X. Wang, J. Huang, X. Gao, Z. Zhang, M. Wang, Y. Huang, C. Zhang, Y. Jiang, J. Wang *et al.*, "Towards 6G Wireless Communication Networks: Vision, Enabling Technologies, and New Paradigm Shifts," *Science China Information Sciences*, vol. 64, pp. 1–74, 2021.

[67] X. Hou, J. Wang, Z. Fang, Y. Ren, K.-C. Chen, and L. Hanzo, "Edge Intelligence for Mission-Critical 6G Services in Space-Air-Ground Integrated Networks," *IEEE Network*, vol. 36, no. 2, pp. 181–189, 2022.

[68] I. Leyva-Mayorga, B. Soret, M. Rper, D. Wbben, B. Matthiesen, A. Dekorsy, and P. Popovski, "LEO Small-Satellite Constellations for 5G and Beyond-5G Communications," *IEEE Access*, vol. 8, pp. 184 955–184 964, 2020.

[69] D. Chen, Z. Liu, L. Wang, M. Dou, J. Chen, and H. Li, "Natural Disaster Monitoring with Wireless Sensor Networks: A Case Study of Data-intensive Applications upon

Low-Cost Scalable Systems," *Mobile Networks and Applications*, vol. 18, pp. 651–663, 2013.

[70] L. Lei, M. Pinedo, L. Qi, S. Wang, and J. Yang, "Personnel Scheduling and Supplies Provisioning in Emergency Relief Operations," *Annals of Operations Research*, vol. 235, pp. 487–515, 2015.

[71] H. Farag, "Enabling Time-and Mission-Critical Applications in Industrial Wireless Sensor Networks," Ph.D. dissertation, Mid Sweden University, 2019.

[72] D. Bhattacherjee, S. Kassing, M. Licciardello, and A. Singla, "In-orbit Computing: An Outlandish thought Experiment?" in *Proceedings of the 19th ACM Workshop on Hot Topics in Networks*, 2020, pp. 197–204.

[73] G. Wang, S. Zhou, S. Zhang, Z. Niu, and X. Shen, "SFC-based Service Provisioning for Reconfigurable Space-Air-Ground Integrated Networks," *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 7, pp. 1478–1489, 2020.

[74] I. Maity, T. X. Vu, S. Chatzinotas, and M. Minardi, "D-ViNE: Dynamic Virtual Network Embedding in Non-Terrestrial Networks," in *2022 IEEE Wireless Communications and Networking Conference (WCNC)*, 2022, pp. 166–171.

[75] W. Qiao, H. Lu, Y. Lu, L. Meng, and Y. Liu, "A Dynamic Service Reconfiguration Method for Satellite-Terrestrial Integrated Networks," *Future Internet*, vol. 13, no. 10, p. 260, 2021.

[76] X. Gao, R. Liu, and A. Kaushik, "Virtual Network Function Placement in Satellite Edge Computing With a Potential Game Approach," *IEEE Transactions on Network and Service Management*, vol. 19, no. 2, pp. 1243–1259, 2022.

[77] X. Gao, R. Liu, A. Kaushik, and H. Zhang, "Dynamic Resource Allocation for Virtual Network Function Placement in Satellite Edge Clouds," *IEEE Transactions on Network Science and Engineering*, vol. 9, no. 4, pp. 2252–2265, 2022.

[78] Z. Jia, M. Sheng, J. Li, R. Liu, K. Guo, Y. Wang, D. Chen, and R. Ding, "Joint Optimization of VNF Deployment and Routing in Software Defined Satellite Networks," in *2018 IEEE 88th Vehicular Technology Conference (VTC-Fall)*, 2018, pp. 1–5.

[79] H. G. Abreha, H. Chougrani, I. Maity, V.-D. Nguyen, S. Chatzinotas, and C. Politis, “Fairness-Aware Dynamic VNF Mapping and Scheduling in SDN/NFV-Enabled Satellite Edge Networks,” in *ICC 2023 - IEEE International Conference on Communications*, 2023, pp. 4892–4898.

[80] H. Hawilo, M. Jammal, and A. Shami, “Orchestrating Network Function Virtualization Platform: Migration or Re-instantiation?” in *2017 IEEE 6th International Conference on Cloud Networking (CloudNet)*, 2017, pp. 1–6.

[81] G. Paschos, “Fairness in Resource Allocation: Foundation and Applications.” [Online]. Available: https://paschos.net/wp-content/uploads/2021/02/Paschos_cloud_computing_part2.pdf

[82] D. Lu, J. Ma, and N. Xi, “A universal fairness evaluation framework for resource allocation in cloud computing,” *China Communications*, vol. 12, no. 5, pp. 113–122, 2015.

[83] F. Ball and K. Basu, “Can Satellite Communications Support Time-Critical Applications,” (30 Agust 2023). [Online]. Available: https://radar.brookes.ac.uk/radar/file/59dee7f2-0842-468a-a4db-ea0ad2af3ba1/1/fulltext.pdf”

[84] M. E. Masum and M. J. Babu, “End-to-End Delay Performance Evaluation for VoIP in the LTE network,” 2011, (30 Agust 2023). [Online]. Available: https://www.diva-portal.org/smash/record.jsf?pid=diva2:831593

[85] M. Jia, L. Zhang, J. Wu, S. Meng, and Q. Guo, “Collaborative Satellite-Terrestrial Edge Computing Network for Everyone-Centric Customized Services,” *IEEE Network*, pp. 1–21, 2022.

[86] P. Korrai, E. Lagunas, S. K. Sharma, S. Chatzinotas, A. Bandi, and B. Ottersten, “A RAN Resource Slicing Mechanism for Multiplexing of eMBB and URLLC Services in OFDMA based 5G Wireless Networks,” *IEEE Access*, vol. 8, pp. 45 674–45 688, 2020.

[87] B. K. Sriperumbudur and G. R. Lanckriet, “On the Convergence of the Concave-Convex Procedure.” in *Nips*, vol. 9, 2009, pp. 1759–1767.

[88] B. K. Saha, L. Haab, and . Podleski, "SAFAR: Simulated Annealing-based Flow Allocation Rules for Industrial Networks," *IEEE Transactions on Network and Service Management*, vol. 18, no. 3, pp. 3771–3782, 2021.

[89] A. S. . D. M. E. Software. Ansys — Engineering Simulation Software. (25 Sept. 2022). [Online]. Available: https://www.ansys.com/products/missions/ansys-stk

[90] H. Pishro-Nik, *Introduction to Probability, Statistics, and Random Processes.* Kappa Research, LLC, 2014.

[91] D. Wu, J. Yan, H. Wang, and R. Wang, "User-Centric Edge Sharing Mechanism in Software-Defined Ultra-Dense Networks," *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 7, pp. 1531–1541, 2020.

[92] R. K. Jain, D.-M. W. Chiu, W. R. Hawe *et al.*, "A Quantitative Measure Of Fairness And Discrimination For Resource Allocation In Shared Computer Systems," *Eastern Research Laboratory, Digital Equipment Corporation, Hudson, MA*, vol. 21, 1984.

[93] J. Zu, G. Hu, Y. Wu, D. Shao, and J. Yan, "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.

[94] Ericsson, "Mobile Data Traffic Forecast," *Ericsson Mobility Report*, 2025. [Online]. Available: https://www.ericsson.com/en/reports-and-papers/mobility-report/dataforecasts/mobile-traffic-forecast

[95] X. Zhu, C. Jiang, L. Kuang, and Z. Zhao, "Cooperative Multilayer Edge Caching in Integrated Satellite-Terrestrial Networks," *IEEE Transactions on Wireless Communications*, vol. 21, no. 5, pp. 2924–2937, 2022.

[96] H. Al-Hraishawi, M. Minardi, H. Chougrani, O. Kodheli, J. F. M. Montoya, and S. Chatzinotas, "Multi-Layer Space Information Networks: Access Design and Softwarization," *IEEE Access*, vol. 9, pp. 158 587–158 598, 2021.

[97] Z. Chen, S. Gu, Q. Zhang, N. Zhang, and W. Xiang, "Energy-Efficient Coded Content Placement for Satellite-Terrestrial Cooperative Caching Network," in *2022 14th International Conference on Wireless Communications and Signal Processing (WCSP)*, 2022, pp. 871–876.

[98] L. Liu, Y. Li, Y. Xu, Q. Zhang, and Z. Yang, "Deep Learning-Enabled File Popularity-Aware Caching Replacement for Satellite-Integrated Content-Centric Networks," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 58, no. 5, pp. 4551–4565, 2022.

[99] R. D. Yates, Y. Sun, and S. K. Kaul, *Age of Information: Foundations and Applications.* Cham: Springer, 2024.

[100] J. Bao, X. Peng, C. Liu, B. Jiang, and J. Wu, "Multilayered Decentralized Coded Caching With Nonuniform Popularity and Multilevel Cache Capacity in SpaceAir-Ground Integrated Networks," *IEEE Internet of Things Journal*, vol. 11, no. 8, pp. 13 913–13 926, 2024.

[101] R. Xu, X. Di, J. Chen, H. Wang, H. Luo, H. Qi, X. He, W. Lei, and S. Zhang, "A Hybrid Caching Strategy for Information-Centric Satellite Networks based on Node Classification and Popular Content Awareness," *Computer Communications*, vol. 197, pp. 186–198, 2023.

[102] M. He, C. Zhou, H. Wu, and X. Sherman Shen, "Learning-based Cache Placement and Content Delivery for Satellite-Terrestrial Integrated Networks," in *2021 IEEE Global Communications Conference (GLOBECOM)*, 2021, pp. 1–6.

[103] Q. Liang, Y. Liu, and W. Tang, "Joint Cache Placement and Content Scheduling in Integrated LEO Satellite-Terrestrial Networks," in *2022 IEEE/CIC International Conference on Communications in China (ICCC)*, 2022, pp. 642–648.

[104] Z. Hu, Z. Jiang, J. Zhao, and Y. Li, "Joint Optimization of Energy and Delay in NGSO Satellite Constellations: Cached Data Sharing based on Spreading Dynamics," *IEEE Transactions on Vehicular Technology*, vol. 72, no. 6, pp. 7884–7899, 2023.

[105] W. Du, Z. Wei, C. Li, Y. Feng, Y. Liu, and B. Mao, "On A Deep Reinforcement Learning-based Content Caching Strategy in 6G Space-Air-Ground Integrated Networks," in *2023 15th International Conference on Communication Software and Networks (ICCSN)*, 2023, pp. 204–208.

[106] M.-H. T. Nguyen, T. T. Bui, L. D. Nguyen, E. Garcia-Palacios, H.-J. Zepernick, H. Shin, and T. Q. Duong, "Real-Time Optimized Clustering and Caching for 6G Satellite-

UAV-Terrestrial Networks," *IEEE Transactions on Intelligent Transportation Systems*, vol. 25, no. 3, pp. 3009–3019, 2024.

[107] D. Han, H. Peng, H. Wu, W. Liao, and X. S. Shen, "Joint Cache Placement and Content Delivery in Satellite-Terrestrial Integrated C-RANs," in *ICC 2021 - IEEE International Conference on Communications*, 2021, pp. 1–6.

[108] S. Gu, X. Sun, Z. Yang, T. Huang, W. Xiang, and K. Yu, "Energy-Aware Coded Caching Strategy Design With Resource Optimization for Satellite-UAV-Vehicle-Integrated Networks," *IEEE Internet of Things Journal*, vol. 9, no. 8, pp. 5799–5811, 2022.

[109] M. Ma and V. W. Wong, "Age of Information Driven Cache Content Update Scheduling for Dynamic Contents in Heterogeneous Networks," *IEEE Transactions on Wireless Communications*, vol. 19, no. 12, pp. 8427–8441, 2020.

[110] 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 Transactions on Mobile Computing*, vol. 22, no. 1, pp. 443–458, 2023.

[111] Z. Liu, Z. Liu, L. Wang, and X. Jin, "The Satellite Network Cache Placement Strategy based on Content Popularity and Node Collaboration," *PloS one*, vol. 19, no. 8, p. e0307280, 2024.

[112] J. Tang, J. Li, X. Chen, K. Xue, L. Zhang, Q. Sun, and J. Lu, "Cooperative Caching in Satellite-Terrestrial Integrated Networks: A Region Features Aware Approach," *IEEE Transactions on Vehicular Technology*, vol. 73, no. 7, pp. 10 602–10 616, 2024.

[113] X. Zhang, B. Zhang, K. An, G. Zheng, S. Chatzinotas, and D. Guo, "Stochastic Geometry-based Analysis of Cache-Enabled Hybrid Satellite-Aerial-Terrestrial Networks With Non-Orthogonal Multiple Access," *IEEE Transactions on Wireless Communications*, vol. 21, no. 2, pp. 1272–1287, 2022.

[114] 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.

[115] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker, “Web Caching and Zipf-like Distributions: Evidence and Implications,” in *IEEE INFOCOM ’99. Conference on Computer Communications. Proceedings. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. The Future is Now (Cat. No.99CH36320)*, vol. 1, 1999, pp. 126–134 vol.1.

[116] R. D. Yates, Y. Sun, D. R. Brown, S. K. Kaul, E. Modiano, and S. Ulukus, “Age of Information: An Introduction and Survey,” *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 5, pp. 1183–1210, 2021.

[117] M. Costa, M. Codreanu, and A. Ephremides, “On the Age of Information in Status Update Systems With Packet Management,” *IEEE Transactions on Information Theory*, vol. 62, no. 4, pp. 1897–1910, 2016.

[118] A. Kosta, *Age of Information Aware Communication Systems: Modeling and Performance Analysis*. Linköping University Electronic Press, 2020, vol. 2060.

[119] P. Lin, Q. Song, and A. Jamalipour, “Multidimensional Cooperative Caching in CoMP-Integrated Ultra-Dense Cellular Networks,” *IEEE Transactions on Wireless Communications*, vol. 19, no. 3, pp. 1977–1989, 2020.

[120] F. Yu, X. Fu, H. Li, and G. Dong, “Improved Roulette Wheel Selection-based Genetic Algorithm for TSP,” in *2016 International Conference on Network and Information Systems for Computers (ICNISC)*, 2016, pp. 151–154.

[121] P. Kora and P. Yadlapalli, “Crossover Operators in Genetic Algorithms: A Review,” *International Journal of Computer Applications*, vol. 162, no. 10, pp. 34–36, 2017.

[122] T. Kelso. Celestrak - NORAD Two-Line Element Sets. Accessed on July 2, 2025. [Online]. Available: Available:https://celestrak.org/NORAD/elements/

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

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

[125] I. CVX Research, “MOSEK Solver,” https://cvxr.com/cvx/doc/mosek.html, Accessed: Jan. 2025.

[126] 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/

[127] G. Megan and E. Jennifer, “How Does Google Make Money?” 2021, Accessed: Nov. 2024. [Online]. Available: https://www.cnbc.com/2021/05/18/how-does-google-make-money-advertising-business-breakdown-.html

[128] R. Dubin, A. Dvir, O. Hadar, T. Frid, and A. Vesker, “Novel Ad Insertion Technique for MPEG-DASH,” in *2015 12th Annual IEEE Consumer Communications and Networking Conference (CCNC)*, 2015, pp. 582–587.

[129] Bitmovin, “A guide to scte-35: What it is and how it works,” 2023, accessed: May 2025. [Online]. Available: https://bitmovin.com/blog/scte-35-guide/

[130] L. Bringuier, “Increasing Ad Personalization with Server-Side Ad Insertion,” in *IBC 2016 Conference*. The Institution of Engineering and Technology, 2016. [Online]. Available: https://digital-library.theiet.org/doi/10.1049/ibc.2016.0044

[131] X. Zhu, C. Jiang, Z. Yang, and H. Wang, “Delay-Optimized Edge Caching in Integrated SatelliteTerrestrial Networks With Diverse Content Popularity Distribution and User Access Modes,” *IEEE Internet of Things Journal*, vol. 11, no. 16, pp. 26 580–26 594, 2024.

[132] 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 *ICC 2024 - IEEE International Conference on Communications*, 2024, pp. 3943–3949.

[133] S. Bhandari, T. X. Vu, S. Chatzinotas, and B. Ottersten, “Efficient Content Caching for Delivery Time Minimization in the LEO Satellite Networks,” in *2023 IEEE International Conference on Communications Workshops (ICC Workshops)*, 2023, pp. 1246–1252.

[134] 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 *GLOBECOM 2022 - 2022 IEEE Global Communications Conference*, 2022, pp. 6541–6546.

[135] Y. Chen, X. Ma, A. Zhou, and S. Wang, "Cooperative Content Caching and Distribution for Satellite CDNs," in *2023 IEEE 31st International Conference on Network Protocols (ICNP)*, 2023, pp. 1–6.

[136] 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.

[137] 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.

[138] S. Pham, K. Hughes, and T. Lohmar, "Implementing Dynamic Ad Insertion in HTML5 using MPEG DASH," in *IBC 2016 Conference*. Institution of Engineering and Technology, 2016. [Online]. Available: https://digital-library.theiet.org/doi/10.1049/ibc.2016.0043

[139] R. Seeliger and L. Bassbouss, "Dynamic Ad Substitution in Hybrid Broadband Broadcast Environments," in *2022 IEEE 47th Conference on Local Computer Networks (LCN)*, 2022, pp. 248–250.

[140] "Convivas 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

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

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

[143] 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/

[144] V. Jatain, "Why Ad Viewability Matters and How to Improve It," *Blockthrough*, 2021. [Online]. Available: https://blockthrough.com/blog/why-ad-viewability-matters-and-how-to-improve-it/

[145] Digitalens, "Cultural Sensitivity in Content: Avoiding Pitfalls in Global Marketing," *Digitalens*, 2023. [Online]. Available: https://digitalens.co.uk/blog/cultural-sensitivity-in-content-avoiding-pitfalls-in-global-marketing

[146] 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.

[147] 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.

[148] A. Patil, S. Prabhu, and P. Ashok, "5G Monetization Strategies and Business Models: An insightful Analysis for Telecommunications Service Providers," in *2024 5th International Conference on Data Intelligence and Cognitive Informatics (ICDICI)*, 2024, pp. 16–23.

[149] K. Fridgeirsdottir and S. Najafi-Asadolahi, "Cost-Per-Impression Pricing For Display Advertising," *Operations Research*, vol. 66, no. 3, pp. 653–672, 2018.

[150] 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

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

[152] J. Kennedy and R. Eberhart, "Particle Swarm Optimization," in *Proceedings of ICNN'95 - International Conference on Neural Networks*, vol. 4, 1995, pp. 1942–1948 vol.4.

[153] A. Chakraborty and A. K. Kar, "Swarm Intelligence: A Review of Algorithms," in *Nature-Inspired Computing and Optimization.* Springer, 2017, pp. 475–494.

[154] SES S.A., "SES Teleport Map - Coverage Overview," 2024, Accessed: Nov. 2024. [Online]. Available: https://www.ses.com/our-coverage/teleport-map

[155] Iridium Communications Inc., "Iridium Communications Inc. - Official Website," 2024, Accessed: Nov. 2024. [Online]. Available: https://www.iridium.com/company/

[156] X. Jin and J. Han, "K-Means Clustering," in *Encyclopedia of Machine Learning*, C. Sammut and G. I. Webb, Eds. Boston, MA: Springer, 2011. [Online]. Available: https://doi.org/10.1007/978-0-387-30164-8_425

[157] 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 Transactions on Network and Service Management*, vol. 21, no. 6, pp. 6716–6730, 2024.

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

[159] 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/

[160] Google Ads, "Actual Cost-Per-Click (CPC): Definition," 2025. [Online]. Available: https://support.google.com/google-ads/answer/6297?hl=en&ref_topic=24937

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

[162] X. Lin, "The Bridge Toward 6G: 5G-Advanced Evolution in 3GPP Release I9," *IEEE Communications Standards Magazine*, vol. 9, no. 1, pp. 28–35, 2025.

[163] N. Alshaer and T. Ismail, "Exploring Quantum Key Distribution for Secure Communication in High-Altitude Platforms," in *2024 24th International Conference on Transparent Optical Networks (ICTON)*, 2024, pp. 1–5.