Priority-Based Service Provision Using Blockchain, Caching, Reputation and Duplication in Edge-Cloud Environments

Tarik CHANYOUR[1], Seddiq EL KASMI ALAOUI[2], Mohamed EL GHMARY[3]
CSS Lab-Science Faculty of Ain-Chock, Hassan II University, Maarif Casablanca, Morocco[1,2]
FSDM, Sidi Mohamed Ben Abdellah University, Atlas-Fez, Morocco[3]

***Abstract*—The integration of Multi-access Edge Computing (MEC) and Dense Small Cell (DSC) infrastructures within 5G and beyond networks marks a substantial leap forward in communication technologies. This convergence is critical for meeting the stringent low latency demands of services delivered to Smart Devices (SDs) through lightweight containers. This paper introduces a novel split-duplicate-cache technique seamlessly embedded within a secure blockchain-based edge-cloud architecture. Our primary objective is to significantly shorten the service initiation durations in high density conditions of SDs and ENs. This is executed by meticulously gathering, verifying, and combining the most optimal chunk candidates. Concurrently, we ensure that resource allocation for services within targeted ENs is meticulously evaluated for every service request. The system challenges and decisions are modeled then represented as a mixed-integer nonlinear optimization problem. To tackle this intricate problem, three solutions are developed and evaluated: the Brute-Force Search Algorithm (BFS-CDCA) for small-scale environments, the Simulated Annealing-Based Heuristic (SA-CDCA) and the Markov Approximation-Based Solution (MA-CDCA) for complex, high-dimensional environments. A comparative analysis of these methods is conducted in terms of solution quality, computational efficiency, and scalability to assess their performance and identify the most suitable approach for different problem instances.**

***Keywords*—Multi-access Edge-cloud Computing; container base image chunks; replication; fragmentation; service provision; blockchain; Markov approximation**

I. Introduction

A. Preliminary

Multi-access Edge Computing (MEC) [1], [2] represents a distributed computing framework, fostering a distributed computing environment at the network edge. By meticulously placing computational resources at access points, edge routers, gateways, base stations or dedicated edge servers, MEC enables the efficient deployment of high-speed, real-time systems and solutions for end-users or Smart Devices (SDs). This approach is optimized for latency-critical, high-capacity data processing, making it ideal for emerging mobile apps and IoT devices.

MEC presents a transformative range of applications, notably within the Internet of Things (IoT) domain, alongside augmented and virtual reality, autonomous vehicles, and smart factories [3]. It empowers localized data processing and analysis, reducing reliance on centralized cloud data centers and effectively addressing latency and bandwidth challenges. Moreover, MEC architectures seamlessly integrate with a wide spectrum of wireless technologies, including next-generation cellular networks, legacy cellular and wireless local area networks. This integration ensures that MEC solutions are versatile and adaptable across different network environments, whether public or private.

Dense Small Cell (DSC) networks, characterized by a high density of small cell base stations within a limited geographic area, have emerged as a critical component of modern cellular architectures. These networks are designed to deliver high-capacity, high-speed connectivity to users. However, the surge in user demands necessitates rapid service provisioning and virtualization [4] to maintain seamless and superior user experiences. Virtualization offers substantial benefits, including enhanced service delivery, scalability, simplified management, and improved performance. This is especially advantageous in the constrained physical environments and dynamic traffic patterns of Dense Small Cell (DSC) networks. By optimizing resource utilization and enabling on-demand provisioning, virtualization helps DSC networks effectively meet the growing demands of users [5]. In addition, containerized applications enable service virtualization, which provides a more sophisticated method for effective service deployment and management in resource-constrained environments. Containerization enables to create lightweight, self-contained service instances that improve operational flexibility and streamline management procedures. This method not only makes resource allocation easier but also supports advanced interference mitigation techniques like network slicing, which guarantee optimal performance and scalability in dynamic network conditions. Nevertheless, Delivering swift service provisioning in DSC environments necessitates a combination of high-speed network connectivity for rapid data transfer, advanced service orchestration to optimize network performance, and low-latency data processing to support real-time applications. These essential components work together to ensure timely and reliable service accessibility for end-users.

Furthermore, by integrating blockchain technology with caching mechanisms [6], [7], [8], containers data can be fragmented and dispersed within multiple ENs. This distribution not only enhances data availability but also reinforces data integrity through distributed verification. Thus, initiating container-based MEC services using splitting/duplicating/caching entails distributing container image fragments across multiple secure nodes using the blockchain technology [9]. Subsequent verification, orchestration, and

service delivery to the user ensure privacy, confidentiality, reliability, and fault tolerance. This decentralized fragmentation approach also alleviates network congestion and improves performance by caching and replicating frequently requested service data.

B. Motivation

This paper aims to optimize service delivery by minimizing container-based service data collection time, bandwidth consumption, and network hops between the user and the target Edge Node (EN). The service data comprises base image chunks of service containers, which require optimal caching, replication, and transfer to the designated target EN. To address this, we formulate an optimization problem considering chunks transfer paths and the availability of diverse resources at each Edge Node. Three solution approaches are proposed: a brute-force search algorithm (BFS-CDCA) for small-scale environments, a simulated annealing-based heuristic (SA-CDCA), and a Markov approximation-based solution for larger, more complex scenarios.

C. Contributions

The distinctive features of this paper can be enumerated as follows:

- A method based on reputation and blockchain for secure service provision, emphasizing the strategic optimized collection of containers' base image from multiple edge nodes, was introduced within a multi-user MEC network.
- An adaptive approach for Containers' Base Image Chunks (CBIC) collection is proposed, ensuring efficient service provision by strategically considering CBIC with the possibility of duplication across Edge Nodes (ENs). The focus is placed on security and efficiency within a dense small cell network environment.
- An optimization problem was formulated to minimize a derived cost function, considering constraints imposed by network bandwidth, cached chunk availability, and strict service initiation deadlines. Notably, the availability of critical resources was enhanced through the implementation of service penalization based on priority.
- Given the NP-hard nature of the formulated optimization problem, a time-efficient heuristic scheme was proposed, incorporating a simulated annealing-based algorithm and a Markov approximation-based solution, with a thorough evaluation of their respective performances.

D. Paper Organization

The remaining sections of this paper are detailed following the structure : Section II presents relevant works related to the study. Section III elaborates on the framework under examination. Section V outlines the formulation of the optimization problem. Following that, Section VII provides an overview of resolution methods, and Section VIII offers an overview of the main evaluation results. Finally, Section IX serves as the conclusion, offering insights into future directions.

II. Related Works

Recent studies, such as [10], have shown a rising interest in using edge computing for the Internet of Things (IoT). Many, like the authors in [11], have investigated how it can enhance performance, primarily by optimizing resources. The authors of [10] focused on reducing costs related to task completion, and employed specialized techniques to lower energy use during tasks. Meanwhile, in [11], wireless charging and task offloading were combined to save energy. Liu et al. developed a new strategy for data storage in edge computing, aiming to boost service providers' profits [12]. Yet, studies such as [13], [14] took a broader view, improving system performance through task offloading and data storage techniques. However, these studies highlight the limitations of individual Edge Nodes (EN) in providing services.

Addressing these shortcomings, the authors of [15]-[16] advocated for EN collaboration to enhance resource utilization and equitably distribute workloads. Specifically, [15] and [17] centered on diminishing task completion durations. While [15] explored collaborative methodologies, [17] added resource allocation to its examination. Feng et al. in [18] concentrated on minimizing user delays and conserving energy. On a different note, [16] embarked on ascertaining the optimal count of collaborating ENs. However, a shared assumption across these studies is that providers freely extend their services, leading to potential reservations about their spontaneous participation absent of incentives.

The subsequent works have primarily centered on enhancing data caching and data sharing by mobile devices. In this context, a plethora of contemporary studies have delved into issues related to edge caching for content sharing. For example, Huang et al. [19] tackled the specific issue of ensuring fairness in data sharing for caching within MEC environments. In a related vein, Asheralieva et al. [20] probed the data caching dilemma using D2D-based method, allowing users to transparently reveal their content sharing costs, diminish average network expenses, and stabilize the queuing framework. Yin et al. [21] put forth a hierarchical strategy for data sharing applications within the same environments, focusing on the challenge of efficient sharing management in MEC networks to enhance user device mobility and heterogeneity.

Progressing further, the authors of [22], [23] amalgamated machine learning methodologies with secure computing techniques to redress the limitations observed in antecedent models. Specifically, in [22], devices were harnessed to execute federated calculations, with an accent on fine-tuning task offloading choices. Concurrently, Cui et al., in their study [23], synergistically integrated secure data storage mechanisms with a blockchain framework, solidifying data integrity. Notwithstanding these innovative strides, it's worth noting that conventional mathematical approaches might grapple when faced with intricate network configurations. To elevate system performance to its zenith, striking a harmonious equilibrium between competitive and collaborative paradigms is paramount. In recent work by [24], a model was proposed concentrating on a singular container-based service. However, this study overlooked the significance of considering the blockchain block size as a crucial parameter, limiting its scope. In contrast, our investigation delves into the integration of multiple container-based services, addressing their specificities

and intricacies. Additionally, we explore the impact of the blockchain operational delay as a critical parameter, considering its implications on the overall system dynamics and performance.

III. The Multi-tier Framework Under Study

In this section, the foundational framework upon which our study is built is outlined. A comprehensive overview of the framework's architecture, components, and key methodologies is provided. The framework aims to provide Container-based Services (CSs) to smart devices (SDs) by offering Container Base Image (CBI) caching with intelligent fragmentation and duplication. It employs a blockchain network to secure the exchange of the resulting Container Base Image Chunks (CBICs) among the ENs constituting a DSC network situated within a two-dimensional geographical area.

Afterwards, the main components of the proposed framework, as illustrated in Fig. 1, are presented in this section.

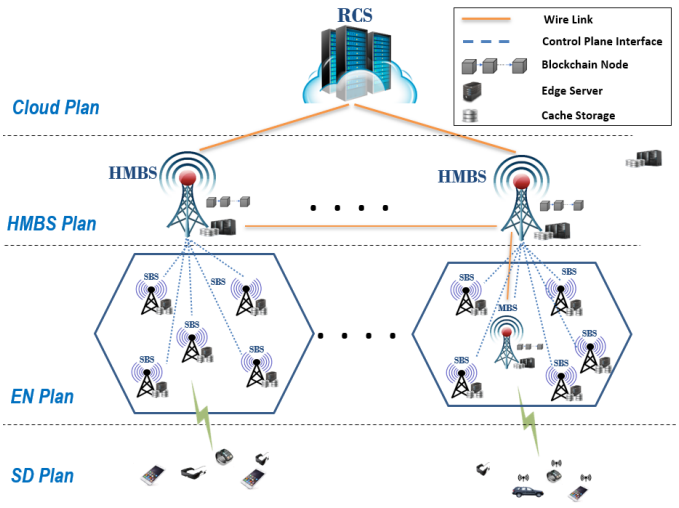

Fig. 1. Overview of the framework's main components.

A. Cloud Server and Edge Nodes

The framework handles a Remote Cloud Server (RCS) and several Small/Macro Base Stations. The RCS is expected to have extensive resources, including virtually unlimited capacities for disk storage, memory, CPU, GPU, etc. The base stations, acting as Edge Nodes (EN), provide wireless network access for smart devices (SDs) within their coverage areas. Additionally, interconnections between ENs are established through wired connections, utilizing high-bandwidth Ethernet cables or fiber-optic connections. Furthermore, each EN supports many users/subscribers and is provisioned with a dedicated edge server that has finite resources, thereby enhancing their capacity to perform localized data processing and service provisioning.

The set of all ENs is organized into multiple regions, each orchestrated by a designated Head Macro Base Station (HMBS) serving as the region head. The clustering of ENs into distinct regions, overseen by HMBSs, enables streamlined network operation and facilitates the delivery of services to end-users, resulting in improved reliability and performance. Additionally, HMBSs take on the role of caching nodes, overseeing image and transaction management to ensure the reliability and security of CBICs while disseminating them across the network. They orchestrate the geographical distribution of published containers' CBICs and handle service initiation requests by refining collection procedures to reduce service initiation time.

B. Container-based Service and Reputation

Each EN can provide only a limited set of independent CSs, due to limited capacities, where each running service uses a container instance and serves one SD only. Advancing further, the base image of the i-th CS, is partitioned into multiple chunks according to the specified chunk size, and subsequently, each CBIC is replicated D_i times.

Furthermore, a reputation management model is integrated, playing a crucial role in fostering trust among entities and enabling secure and informed caching decisions. This model dynamically allocates a reputation score to each CS i within each region r based on its historical usage and interactions. This reputation reflects the size of the demand for the service within that specific region. As a result, the degree of duplication of a CS within a region is closely tied to its reputation. Essentially, the more reputable a CS is perceived to be, the higher the likelihood of it being duplicated within that region. This approach is driven by the principle that services with a stronger reputation are in higher demand and therefore benefit from having more duplicates available to meet user requests promptly and efficiently.

Practically, the reputation scores are computed based on all pertinent trust information and feedback provided by the network entities. These scores serve to establish trust relationships among caching entities, including service providers and edge nodes.

C. Blockchain Network

The blockchain network operations in the proposed secure caching framework comprises the following four phases:

1) System initialization: During this phase, the Trusted Authority (TA) executes a Setup procedure to compute public parameters. TA generates cyclic groups with a prime order, selects random exponents and hash functions, computes essential parameters to release the public parameters to all involved entities of the blockchain network.

2) Entity registration: Upon joining the blockchain network, both the Service Provider (SP) and the ENs undergo a registration phase, during which the Trusted Authority (TA) authenticates their identities. TA executes a key generation procedure to generate random values and computes secret keys for encryption, signing, and verification. TA assigns the signing key to SP and the decryption key to edge nodes (EN) through a secure communications channel.

3) Block creation and validation: Registered SPs encrypt message chunks with a randomly generated symmetric secret key and define access structures to control access to encrypted messages by target edge nodes (ENs). SPs then signcrypt the secret key under the access structure and send the resulting ciphertext to the blockchain network for validation. This record

includes the SP's public key, pseudo identity, block hash, and signcrypted ciphertext along with the SP's signature. All registered SPs are considered authority candidates for validation. A genesis block is created at the initiation of the permissioned chain, and time intervals are allocated for affixing single blocks to the chain. In the case of multiple authorities, one leader per interval collects and validates records before passing them to other candidate authorities. Validated records are included in new blocks along with blockheaders containing relevant metadata.

4) Chunks distribution: Upon receiving a new block, the network may opt to forward the blockchain header to the edge nodes (ENs), allowing them to decide whether to request the signed ciphertext (ST) and signature (π) from specific blocks via a pull request. Upon receiving ST and π from the blockchain, ENs decrypt to recover the symmetric key (keysym), then decrypt the chunk message and verify its integrity and signature. After obtaining keysym, ENs use it to decrypt the message and calculate verification parameters, comparing them to π to confirm that the message has not been altered.

IV. System Model

This section offers a comprehensive overview of the modelization of the primary components essential to our study. Here, to simplify our notation, we will assign the variables i, j, k, n, p, r and m to refer container-based services, chunks, chunks' duplicates, edge nodes, paths, regions and resources, respectively.

A. Nodes and Resources

The CSs are denoted by the set $\mathcal{S} = \{S_1, S_2, ..., i, ..., S_{\sigma_s}\}$, distributed among the Edge Nodes (ENs) represented by the set $\mathcal{N} = \{N_1, N_2, ..., n, ..., N_{\sigma_n}\}$. Each EN is equipped with a edge server which in turn offers a diverse array of resources across multiple categories, encompassing CPU, GPU, TPU, FPGA, memory, storage, network bandwidth, etc. Here, the set of possible σ_z resources is denoted $\mathcal{Z} = \{r_1, r_2, ..., r_{\sigma_z}\}$. Accordingly, every node n is characterized by its capacity set in terms of resources which we denote $\mathcal{Z}_n^{cap} = \{Z_{n,1}^c, Z_{n,2}^c, ..., Z_{n,\sigma_z}^c\}$. Here, $Z_{n,m}^c$ denotes the maximum quantity of resource r_m that node n can provide. Furthermore, its current resource utilization is represented by: $\mathcal{Z}_n^{use} = \{Z_{n,1}^u, Z_{n,2}^u, ..., Z_{n,\sigma_z}^u\}$, where $Z_{n,m}^u$ indicates the amount of resource r_m utilized in node n.

B. Container-based Service and Chunks

For convenience, we will interchangeably use the term Container-based Service (CS) or its user, and denote CS S_i as i. Then, to summarize the operational parameters related to CS i, we use the notation Ω_i, defined as :

$$\Omega_i \triangleq \langle R_i, \pi_i, \pi_i^{min}, \pi_i^{max}, \rho_i, N_i^t, \mathcal{M}_i, B_i^{ser}, \mathcal{Z}_i^{dem}, D_i, t_i^{trans}, A_i \rangle.$$

As shown in Table I, the operational parameters for CS i encompass essential details regarding its initiation and execution.

TABLE I. Summary of Operational Parameters for CS i

Parameter	Description
Ω_i	The operating parameters of CS i
R_i	Region receiving the service initiation request for CS i.
$\mathcal{C}_i$	The set of chunks related to CS i with the size σ_i
D_i	Number of duplicates of the CS i.
A_i	Total data amount of CS i.
π_i	Priority score of CS i.
π_i^{min}, π_i^{max}	Priority score bounds of CS i.
ρ_i	Reputation score of CS i.
N_i^t	EN receiving the service initiation request for CS i.
$\mathcal{M}_i$	Localization matrix of all available duplicates of chunks associated with CS i.
B_i^{ser}	Minimum permissible data rate for the available bandwidth between the node hosting CS i and N_i^t.
$\mathcal{Z}_i^{dem}$	Resource demand of CS i, quantified by the number of standardized virtual resource units.
t_i^{trans}	Maximum permissible deployment delay for CS i.

Specifically, $\mathcal{Z}_i^{dem} = \{Z_{i,1}^d, Z_{i,2}^d, ..., Z_{i,\sigma_z}^d\}$ denotes the resource demand of CS i in terms of resources, where the resources are quantified by the number of standardized virtual resource units. Here, σ_z denotes the number of resource types, and $z_{i,m}^d$ indicates the required quantity of resource r_m.

Afterwards, the set of chunks of CS i is denoted $C_i = \{C_{i,1}, C_{i,2}, ..., C_{i,\sigma_i}\}$ where σ_i is the chunks count of CS i. For ease of use the j chunk of C_i is denoted as $C_{i,j}$ ($j \in \mathcal{C}_i = [\![1; \sigma_i]\!]$) and defined by the following key parameters: $\Omega_{i,j} \triangleq \langle I_{i,j}, D_{i,j} \rangle$. Here, $I_{i,j}$ is the identifier used to determine the order of the chunk within the CS i, and $D_{i,j}$ denotes the total data size of the chunk, measured in bytes. Moreover, in region R_i the σ_i duplicates of all chunks belonging to CS i are distributed across its ENs. The location information is provided by a set of σ_i matrices that are continually updated by the HMBS. This set specifies, for each CS i, the identifiers of the ENs caching all its chunk duplicates. The matrix $\mathcal{M}_i$ linked to CS i is of dimensions $(\sigma_i \times D_i)$. For convenience, the k-th duplicate ($k \in \mathcal{K} = \{1, 2, ..., D_i\}$) of the j-th chunk of CS i is denoted as $C_{i,j,k}$ and located in EN $\mathcal{M}_{i,j,k}$.

C. Service Priority and Reputation

Penalization of services in dense small cell networks is an unavoidable challenge, often stemming from users diversity and resource constraints such as limited bandwidth, computational capabilities, or storage/memory availability. Capacity constraints within a system may lead to the prioritization of certain users over others, achieved through a deliberate reduction in the pace at which their service requests are fulfilled. This selective prioritization strategy is implemented to effectively manage resource limitations, ensuring that critical users or tasks receive timely attention while acknowledging and potentially delaying less urgent requests. Such prioritization mechanisms play a crucial role in optimizing resource utilization and maintaining system stability under high demand scenarios. In response to these challenges, we adopt a prioritization model wherein users with higher priorities are subjected to lesser penalties, if necessary. Accordingly, a priority π_i is attributed to CS i such that:

$$\pi_i \in \left\{\pi_i^{min}, ..., \pi_i^{max}\right\} \cup \{M, M + \rho_i\} \quad ; i \in \mathcal{S} \qquad (1)$$

Here, π_i^{min} and π_i^{max} respectively represent the minimum and maximum allowable priority scores for CS i. M is a very

big number such that $\pi_i^{max} \ll M$, $\forall i \in \mathcal{S}$. The priority factors are used to favour the transfer of CS. In particular, CS with priority at least $\pi_i = M$ are high priority containers that are solicited for urgent transfer. Other priorities are determined according to the service continuity requirement in terms of downtime or the service level agreement.

Simultaneously, users who have been penalized in one round may be perceived as potential priority users in subsequent rounds. This dynamic approach recognizes that users experiencing penalties in a time slot might require special attention or resource allocation in subsequent slots to ensure fair access and equitable long term treatment. By adapting priorities based on historical user interactions, the system aims to efficiently handle specific needs and requirements of individual users over time. As a result, within our model, the penalization score undergoes adjustments based on the following criteria:

- If user i holds high-priority score $M + \rho_i$, its priority score remains unchanged. Here, ρ_i, referred the reputation score which is introduced to establish a hierarchical order among high-priority users to ensuring their precedence.
- Alternatively, if user i does not possess high-priority status:
  - If its request is fulfilled during the current time slot, its priority π_i is reset to its initial value π_i^{min}.
  - If its request remains unsatisfied during the current slot, its priority in the subsequent time slot is either incremented by 1 or set to M: if π_i reaches the maximum allowable value π_i^{max}, π_i is set to M, otherwise it is elevated to $\pi_i + 1$.

D. Network Model

The network model is given by $\mathcal{P} = \{\mathcal{P}_{n\to n'} \,/\, n \in \mathcal{N}; n' \in \mathcal{N} \setminus \{n\}\}$ composed of all sufficient paths connecting all pairs of distinct nodes (n, n'). Also, we use $\mathcal{P}_{i,j,k}^n$ to denote the set $\mathcal{P}^{\mathcal{M}_{i,j,k}\to n}$ of paths connecting EN $\mathcal{M}_{i,j,k}$ housing chunk $C_{i,j,k}$ to node n. Without loss of generality, we assume that the set $\mathcal{P}_{i,j,k}^n$ is precalculated and given at the decision time slot.

Moreover, each available path $p \in \mathcal{P}_{n\to n'}$ that is used for multi-hop data transmission is characterized by [5], [25]:

- its hop count $\mathcal{H}_{n\to n'}^p$
- its allocatable bandwidth $\mathcal{B}_{n\to n'}^p$
- traversing delay $\mathcal{D}_{n\to n'}^p$

Thus, if EN n is the transfer destination of the CS i's chunks, the backhaul bandwidth between n and the target node N_i^t, serving the user of CS i's, is given by:

$$\mathcal{B}_i^n = \begin{cases} \infty & ;n = N_i^t \\ \max\limits_{p\in\mathcal{P}_{n\to N_i^t}} \left\{\mathcal{B}_{n\to N_i^t}^p\right\} & ;n \neq N_i^t \end{cases} \quad ;i \in \mathcal{S}; n \in \mathcal{N} \qquad (2)$$

E. Delays

The cumulative delay while transferring a data byte of chunk $C_{i,j,k}$ using path p in the set $\mathcal{P}_{i,j,k}^n$ is denoted $\mathcal{D}_{i,j,k}^{n,p}$.

In this study, the service collection process involves transferring the base image of its container from the most suitable caching ENs to the nearest node possible to node N_i^t to cater to the user's request. According to the specific service being requested, the provider's CBI might be exclusively stored in the RCS, completely located on an EN, or distributed as multiple chunks across regional ENs. Upon receiving a new service request, the associated node acquires necessary instructions from the HMBS. The MBS selects the most appropriate procedure from three potential options based on specific circumstances. Notably, the focus of this study is the third option, which encompasses the characteristics of the other two scenarios. Consequently, the transfer latency experienced by a specific chunk, denoted as $C_{i,j,k}$, to EN n is determined by the cumulative transfer delays of all selected chunks from their respective caching nodes. These delay are influenced by the underlying blockchain network architecture and prevailing transfer conditions. As such, they can be decomposed into the following components:

1) Blockchain-related operational delays: [26], [27] this delay encompasses the time it takes for a node to respond and send a requested block back to the requester. This delay is intricately tied to factors like node processing power, data volume, network bandwidth, and the blockchain protocol employed. In our approach, we extend this modeling by incorporating the block size as an additional parameter, recognizing its influence on delay. Specifically, we characterize this delay as a linear function of both the chunk size and block size, introducing two parameters associated with the hosting EN and the HMBS. This refinement allows for a more comprehensive representation of the operational delay in the blockchain network.

2) Network-related transfer delays: which refers to the time it takes for the transfer of the CBIC to propagate through the network from its holding node to its final decided node. This duration is influenced by several factors, including network topology, congestion, and the number of hops.

The first delay, denoted ${}^{BC}T_{i,j,k}$, is given in the next formula where $a_{i,j,k}$, $b_{i,j,k}$ and $c_{i,j,k}$ are the delay-related parameters associated with the hosting EN $\mathcal{M}_{i,j,k}$:

$${}^{BC}T_{i,j,k} = a_{i,j,k} * D_{i,j} + b_{i,j,k} * L^{Bloc} + c_{i,j,k} \qquad (3)$$

Here, $i \in \mathcal{S}$, $j \in \mathcal{C}_i$, $k \in \mathcal{K}$, $D_{i,j}$ is the total data amount of chunk $C_{i,j}$, the terme $b_{i,j,k} * L^{Bloc}$ represents the delay overhead related to the adopted block-chain block size L^{Bloc}.

Achieving precise estimates for the parameters $a_{i,j,k}$, $b_{i,j,k}$ and $c_{i,j,k}$ within the blockchain edge-cloud network involves a comprehensive strategy. Initial empirical experiments provide a foundational dataset for response time with varying chunk sizes. Employing regression analysis offers initial parameter estimates. To enhance precision, integrate machine learning (ML) techniques, utilizing supervised learning algorithms and considering features like hosting EN attributes, blockchain protocol, and chunk characteristics. Advanced ML methods, including neural networks, contribute to capturing intricate

relationships. Continuous monitoring and adaptation based on real-world performance data ensure ongoing accuracy. This hybrid empirical-ML approach establishes a robust framework for dynamic and precise estimation of the delay-related parameters in the blockchain edge-cloud network.

The second delay depends on the decided target node n and it is given by:

$$^{NET}T^{n,p}_{i,j,k}=\begin{cases} 0 & ;n=M_{i,j,k} \\ D_{i,j}\times\mathcal{D}^{n,p}_{i,j,k} & ;n\neq M_{i,j,k} \end{cases} \quad (4)$$

where $i\in\mathcal{S}; j\in\mathcal{C}_i; k\in\mathcal{K}; n\in\mathcal{N}; p\in\mathcal{P}^n_{i,j,k}$.

Deploying a CS in this work refers to the transfer of all its CBICs from the distributed hosting nodes to the decided MEC server in order to make it ready to start serving the requester user. Thus, a new incoming service request from a user trigger the service deployment from their hosting nodes to the best available nearby EN.

Hence, the transfer process delay of duplicate $C_{i,j,k}$ to EN n is composed of the blockchain operational delay as well as the transfer delay from the storing nodes to the decided hosting node n. Using Eq. 3 and 4 we can formulate the overall transfer delay $T^{n,p}_{i,j,k}$ related to the duplicate $C_{i,j,k}$ as follows:

$$T^{n,p}_{i,j,k}={}^{BC}T_{i,j,k}+{}^{NET}T^{n,p}_{i,j,k} \quad (5)$$

Lastly, Table II provides an inventory of the primary notations employed in this paper.

TABLE II. MAIN NOTATIONS

Notation	Definition
$\mathcal{N}$	The set of edge-cloud nodes
$\mathcal{S}$	The set of container-based services
σ_n, σ_c, σ_r	The the total number of nodes, CS and resources
$\mathcal{P}_{n\to n'}$	The set of available paths connecting nodes N_n and $N_{n'}$
$\mathcal{B}^p_{n\to n'}$	The bandwidth of path $p\in\mathcal{P}_{n\to n'}$
$\mathcal{H}^p_{n\to n'}$	The hop count of path $p\in\mathcal{P}_{n\to n'}$
$D_{i,j}$	The total data amount of the container, chunk $C_{i,j}$
$\mathcal{P}^n_{i,j,k}$	The set of available paths connecting nodes N_n and $N_{n'}$
$\mathcal{D}^n_{i,j,k}$	The transfer cumulative delay using path $p\in\mathcal{P}^n_{i,j,k}$
$Z^d_{i,m}$	The CS i's demand in terms of resource r_m
$Z^u_{n,m}$	The available resource of node N_n in terms of resource r_m

V. THE OPTIMIZATION PROBLEM

In this section, the fundamental problem that serves as the focal point of our study is articulated. The variables, objectives, and constraints are clearly defined to establish a comprehensive formulation and delineate the overarching objective that requires optimization.

A. Decision Variables

To accurately represent the operations within our system, the decision variables employed in our model are introduced as follows:

The transfer binary decision variable of CS i to node n is denoted α^n_i where $\alpha^n_i=1$ refers to the decision to deploy C_i to node n, otherwise, $\alpha^n_i=0$.

$$\alpha^n_i\in\{0;1\} \qquad ;i\in\mathcal{S}; n\in\mathcal{N} \quad (6)$$

The duplicate choice binary decision variable of chunk duplicate $C_{i,j,k}$ is denoted $\beta_{i,j,k}$ where $\beta_{i,j,k}=1$ refers to the decision to transfer $C_{i,j,k}$, otherwise, $\beta_{i,j,k}=0$.

$$\beta_{i,j,k}\in\{0;1\} \qquad ;i\in\mathcal{S}; j\in\mathcal{C}_i; k\in\mathcal{K} \quad (7)$$

Additionally, when transferring chunk $C_{i,j,k}$ from its caching node $M_{i,j,k}$ to node n the decision variable to select the migration path p among the possible paths set $\mathcal{P}^n_{i,j,k}=\mathcal{P}_{M_{i,j,k}\to n}$ is the binary variable $\gamma^{n,p}_{i,j,k}$ where $\gamma^{n,p}_{i,j,k}=1$ refers to the decision to use the p-th path in $\mathcal{P}^n_{i,j,k}$ to migrate $C_{i,j,k}$ from node $M_{i,j,k}$ to n, otherwise $\gamma^{n,p}_{i,j,k}=0$.

$$\gamma^{n,p}_{i,j,k}\in\{0;1\} \qquad ;i\in\mathcal{S}; j\in\mathcal{C}_i; k\in\mathcal{K}; n\in\mathcal{N}; p\in\mathcal{P}^n_{i,j,k} \quad (8)$$

In situations where the available system resources are insufficient to support the initiation of a CS, our proposed model offers the option to defer the initiation of this CS. To represent this penalization decision for C_i, we utilize the binary variable δ_i, where $\delta_i=0$ indicates penalization, while $\delta_i=1$ denotes no penalization.

$$\delta_i\in\{0;1\} \qquad ;i\in\mathcal{S} \quad (9)$$

B. The Cost Model

In this section, we present the proposed Cost Model, a fundamental component of our system design aimed at comprehensively assessing and optimizing the framework decisions.

1) Delay cost: In our model, when deploying CSs to the designated hosting nodes, the delay cost incurred during the transmission of data transfer flows is contingent on the congestion levels of the network's links. Consequently, with a placement decision vector α, a duplicates selection vector β, and a path selection vector γ as well as using Eq. 5 we can formulate the transfer delay related to the CS i s as follows:

$$T_i(\alpha,\beta,\gamma)=\sum_{j\in\mathcal{C}_i}\sum_{k\in\mathcal{K}}\sum_{n\in\mathcal{N}}\sum_{p\in\mathcal{P}^n_{i,j,k}}\alpha^n_i\beta_{i,j,k}\gamma^{n,p}_{i,j,k}\left(T^{n,p}_{i,j,k}\right) \quad (10)$$

As a result, the comprehensive delay-related cost can be calculated as follows:

$$\Psi^{delay}(\alpha,\beta,\gamma,\delta)=\sum_{i\in\mathcal{S}}\delta_i T_i(\alpha,\beta,\gamma) \quad (11)$$

Additionally, for normalization purpose we use the total maximal allowable transfer delay for all CS given by:

$$\Psi^{delay}_{max}=\sum_{i\in\mathcal{S}}t^{trans}_i \quad (12)$$

2) Backhaul cost: Once all the required container chunks related to S_i have been transferred to destination EN n, the backhaul bandwidth between the connected EN N_i^t and n plays a crucial role in determining the service-related user's perceived delay. Ideally, the user's experience is best when $n = N_i^t$, and we aim to prioritize such decisions. To achieve this, our model takes into account two main costs related to EN n: the available bandwidth between EN n and N_i^t as well as the hop count between them. Utilizing a placement decision vector α and employing Eq. 2, the serving bandwidth for the user linked to CS S_i can be expressed as:

$$\mathcal{B}_i(\alpha) = \sum_{n \in \mathcal{N}} \alpha_i^n \mathcal{B}_i^n \qquad ; i \in \mathcal{S} \tag{13}$$

The resulting backhaul cost function, denoted as $\Psi_{i,n}^{back}$, is defined in such a way that it equals 0 when $n = N_i^t$. However, in all other cases, it takes on a value as follows:

$$\Psi_{i,n}^{back} = \min_{p \in \mathcal{P}_{n \to N_i^t}} \left\{ W_r \frac{\min\limits_{p' \in \mathcal{P}_{n \to N_i^t}} \mathcal{B}^{p'}_{n \to N_i^t}}{\mathcal{B}^{p}_{n \to N_i^t}} + W_h \frac{\mathcal{H}^{p}_{n \to N_i^t}}{\max\limits_{p' \in \mathcal{P}_{n \to N_i^t}} \mathcal{H}^{p'}_{n \to N_i^t}} \right\} \tag{14}$$

Here, $i \in \mathcal{S}$, $n \in \mathcal{N}$ and the weights W_r and W_h are two adjustable weights to fine-tune the optimization process for different scenarios, where W_r represents the weight associated with available bandwidth, and W_h represents the weight associated with hop count costs, with the constraint that $W_r + W_h = 1$. Moreover, the cost function $\Psi_{i,n}^{back}$ falls within the range [0,1], and the use of the max and min expressions for fractions serves the purpose of normalization.

Therefore, with the decision vector α, the overall user backhaul cost can be obtained as:

$$\Psi^{back}(\alpha, \delta) = \sum_{i \in \mathcal{S}} \left\{ \delta_i \sum_{n \in \mathcal{N}} \alpha_i^n \Psi_{i,n}^{back} \right\} \tag{15}$$

3) Load balancing cost: The load balancing procedure targets an equitable distribution of the incoming service workloads, aiming to minimize deviations from the average value. Within our model, the associated cost must consider the processing workloads across all ENs. This includes the current workload of running services on each EN, along with anticipating the additional load expected from services that will be selected to run on them.

Initially, following the CS deployment, the anticipated resource utilization $Z_{n,m}^u$ of processing resource r_m on node n can be determined using the following equation:

$$Z_{n,m}^a(\alpha, \delta) = Z_{n,m}^u + \textstyle\sum_{i \in \mathcal{S}} \alpha_i^n \delta_i Z_{i,m}^d \qquad ; n \in \mathcal{N}; m \in \mathcal{Z}^p. \tag{16}$$

Next, we establish the processing load ratios $\theta_{n,m}$ associated with resource r_m in EN n, along with their mean value $\bar{\theta}_m$, defined as follows:

$$\theta_{n,m}(\alpha, \delta) = \frac{Z_{n,m}^a(\alpha, \delta)}{Z_{n,m}^c} \in [0,1] \qquad ; n \in \mathcal{N}; m \in \mathcal{Z}^p \tag{17}$$

$$\overline{\theta_m}(\alpha, \delta) = \sum_{n \in \mathcal{N}} \frac{\theta_{n,m}(\alpha, \delta)}{\sigma_n} \in [0,1] \qquad ; m \in \mathcal{Z}^p \tag{18}$$

Subsequently, the processing load concerning EN n across all processing resource types in $\mathcal{Z}^p$ is defined as:

$$\Psi_n^{load}(\alpha, \delta) = \sum_{m \in \mathcal{Z}^p} \frac{|\theta_{n,m}(\alpha, \delta) - \overline{\theta_m}(\alpha, \delta)|}{\sigma_z} \qquad ; n \in \mathcal{N} \tag{19}$$

Finally, the resulting overall processing load is as follows:

$$\Psi^{load}(\alpha, \delta) = \sum_{n \in \mathcal{N}} \sum_{m \in \mathcal{Z}^p} \frac{|\theta_{n,m}(\alpha, \delta) - \overline{\theta_m}(\alpha, \delta)|}{\sigma_n \sigma_z} \in [0,1] \tag{20}$$

C. The Multi-Objective Function

The adopted multi-objective function, to be elaborated upon, consists of two components: the overall cost and the degree of penalization for users. We begin by introducing the overall cost model below. It is constructed using a multi-objective function, which combines the detailed cost metrics into a weighted sum using the weight aggregation approach:

$$\Psi(\alpha,\beta,\gamma,\delta) = W_d \frac{\Psi^{delay}(\alpha,\beta,\gamma,\delta)}{\Psi_{max}^{delay}} + W_b \frac{\Psi^{back}(\alpha,\delta)}{\sigma_c} + W_l \Psi^{load}(\alpha,\delta) \tag{21}$$

Here, W_d, W_b, and W_l serve as regulatory weight constants that determine the priority attributed to each cost. Their values range between 0 and 1, satisfying the condition $W_d + W_b + W_l = 1$. Furthermore, the denominators in this function act as normalization factors for each cost function. The metrics normalizing procedure involves transforming the values of the three studied metrics into dimensionless costs between 0 and 1. This ensures the ability to addition and comparison within the cost function by standardizing the metrics' scales.

Next, a penalty function is introduced in order to minimize the number of penalized users. It is proposed such that high-priority users could be penalized, if necessary, only if all non priority users are penalized. Accordingly, we adopt the following penalty function given by:

$$\Pi(\delta) = \sum_{i \in \mathcal{S}} (1 - \delta_i) \pi_i \tag{22}$$

The formulation of this function establishes a penalization hierarchy among the users, where the decision to penalize any CS guarantees that high-priority containers are penalized last, if required.

At this point, we can finally state the following equation, which defines the adopted overall multi-objective function, referring to the objective function targeted for minimization:

$$\Theta(\alpha, \beta, \gamma, \delta) = \Psi(\alpha, \beta, \gamma, \delta) + \Pi(\delta) \tag{23}$$

D. The Constraints

In our model, the case when CS i is not transferred (penalized by setting $\delta_i = 0$) is represented by setting $\alpha_i^n = 0$ for $n \in \mathcal{N}$ and if transferred, only one target node is selected. Accordingly, the transfer decision of CS i has to meet the following constraint:

$$\sum_{n\in\mathcal{N}} \alpha_i^n = \delta_i \qquad ; i \in \mathcal{S} \tag{24}$$

Additionally, if i is not penalized, only one duplicate of chunk $C_{i,j,k}$ must be selected, resulting in the following constraint:

$$\sum_{k\in\mathcal{K}} \beta_{i,j,k} = \delta_i \qquad ; i \in \mathcal{S}; j \in \mathcal{C}_i \tag{25}$$

Also, all the σ_i chunks of CS i, if not penalized, must be transferred to the selected node which lead to the following constraint:

$$\sum_{j\in\mathcal{C}_i}\sum_{k\in\mathcal{K}} \beta_{i,j,k} = \delta_i\sigma_i \qquad ; i \in \mathcal{S} \tag{26}$$

By selecting only one path p in $\mathcal{P}^n_{i,j,k}$ to serve the transfer flow of chunk $C_{i,j,k}$, if i is not penalized, from its node $\mathcal{M}_{i,j,k}$ to target node n, the next constraint has to be satisfied:

$$\sum_{p\in\mathcal{P}^n_{i,j,k}} \gamma^{n,p}_{i,j,k} = \delta_i \qquad ; i \in \mathcal{S}; j \in \mathcal{C}_i; k \in \mathcal{K}; n \in \mathcal{N} \tag{27}$$

Therefore, the anticipated demand for each resource r_m on each node n must fulfill every resource type requirement. This can be formally formulated as:

$$Z^a_{n,m}(\alpha,\delta) \leq Z^c_{n,m} \qquad ; n \in \mathcal{N}; m \in \mathcal{Z}. \tag{28}$$

The subsequent constraint ensures that the total delay incurred during the collection of all chunks associated with CS i does not surpass its tolerated transfer delay t_i^{trans}.

$$\delta_i T_i(\alpha,\beta,\gamma) \leqslant t_i^{trans} \qquad ; i \in \mathcal{S} \tag{29}$$

Finally, the bandwidth constraint associated with CS i, utilizing the maximum available bandwidth specified in Eq. (13), is formalized as follows:

$$\mathcal{B}_i(\alpha) \geqslant \delta_i B_i^{ser} \qquad ; i \in \mathcal{S} \tag{30}$$

E. The Problem Formulation

Based on the elucidated problem context, the following optimization problem, designated as $\mathscr{P}1$, aims to optimize the aforementioned dual objectives. The solution aims to maximize user benefits while respecting resource constraints, meeting transfer delay and bandwidth requirements, and minimizing penalties associated with priority.

$$\mathscr{P}1 : \underset{\{\alpha,\beta,\gamma,\delta\}}{\text{minimize}}\ \Theta(\alpha,\beta,\gamma,\delta)$$
$$\text{s.t. } (6),(7),(8),(9),(24),(25),(26),(27),(28),(29),(30)$$

VI. Problem Decomposition

The general problem $\mathscr{P}1$ involves making decisions regarding CSs placement (α decision variables), selecting CBICs along with path determination (β and γ decision variables) as well as the final penalization decision (δ decision variables). Next, $\mathscr{P}1$ is decomposed into sub-problem as well as a general problem. The sub-problem (CSPDP) deals exclusively with the aspect concerning the selection of CBICs along with path determination and penalization decisions. It is assumed that the decisions regarding container placement have been made. Subsequently, the general problem (GCP) iterates over the placement possibilities and employs the solution from the CSPDP to derive the overall solution.

A. CSPDP Sub-Problem

The formulation of this problem can be presented as follows:

$$\mathcal{CSPDP} : \underset{\{\beta,\gamma,\delta^\alpha\}}{\text{minimize}}\ \Theta^\alpha(\beta,\gamma,\delta^\alpha)$$
$$\text{s.t. } (7),(8),(9),(25),(26),(27),(29)$$

In this formulation, with fixed decisions α, the objective function value is obtained using Eq. (23). Additionally, the penalty vector δ^α pertains only to containers for which resource and bandwidth constraints ((28) and (30) respectively) are satisfied based on decisions α. That is to say, if a certain container i is penalized with decisions α ($\delta_i = 0$), then δ_i^α always equals 0 and is not considered a variable for the CSPDP problem. The remaining penalty decisions of δ^α are related to transfer time constraints that depend on selecting CBICs along with path determination (β and γ decision variables).

Subsequently, a proposition related to the set of feasible solutions of the problem will be demonstrated.

Proposition 1. The set $\mathcal{S}_f^\alpha$ of feasible solutions of $\mathcal{CSPDP}$ is non-empty.

Proof: By utilizing binary decisions, constraints $(7),(8),(9)$ are fulfilled. Additionally, $\forall\alpha$, with a penalty vector $\delta^+ = \mathbf{0}$, resource constraints ((28) and (30)) will be satisfied. Let the decision vectors β^+ and γ^+ be constructed such that $\beta^+ = \mathbf{0}$ and $\gamma^+ = \mathbf{0}$. Since $\delta_i^+ = 0\ \forall i$, all remaining constraints $(25),(26),(27),(29)$ are satisfied. Therefore, $(\beta^+,\gamma^+,\delta^+) \in \mathcal{S}_f^\alpha \Rightarrow \mathcal{S}_f^\alpha \neq \emptyset$. ■

B. GCP Global Problem

The general problem $\mathscr{P}1$ involves traversing all possible decisions given by the containers placement α and solving the sub-problem CSPDP at each iteration. This means that for each potential configuration α, problem CSPDP is solved to obtain a corresponding optimal solution. This iterative process is repeated until all possibilities of α are explored, thereby determining the best global solution for problem $\mathscr{P}1$.

Thus, by using the next constraint, this challenging placement problem, denoted $\mathcal{GCP}$, can be further formulated:

$$(\beta,\gamma,\delta^\alpha) \in \mathcal{S}_f^\alpha \tag{31}$$

and the GCP Global Problem is as follows:

$$\mathcal{GCP} : \underset{\{\alpha\}}{\text{minimize}}\ \Theta^{\alpha}\left(\beta, \gamma, \delta^{\alpha}\right)$$
$$\text{s.t. } (6), (24), (28), (30), (31)$$

However, this problem's placement decisions lies in optimally placing σ_s containers within σ_n edge nodes. This placement must consider resource constraints, including available resources and bandwidth capacity, which contribute to the inherent complexity of the problem. Consequently, the next proposition 2 is derived.

Proposition 2. The optimization problem $\mathcal{GCP}$ is NP-hard.

Proof: Given σ_s services to place within σ_n ENs. To find the best placement solution for problem $\mathcal{GCP}$, even when all resources are sufficient, the problem remains one of determining the optimal placement of σ_s services among σ_n nodes, the computation complexity of $\mathcal{GCP}$ can reach $O(\sigma_n^{\sigma_c})$. Therefore, problem $\mathcal{GCP}$ is NP-hard and cannot be well solved in polynomial time. ■

Subsequently, an efficient resolution of $\mathscr{P}1$ (or its equivalent form $\mathcal{GCP}$) will be presented in the next section.

VII. Problems Resolution

Now, the resolution of the obtained optimization problem $\mathscr{P}1$ is explored. The identified optimization-related challenges will be tackled through a detailed approach, and the derived solutions will be presented. The end of this section will provide insights into our efforts to effectively address the problem and demonstrate our commitment to achieving optimal results.

A. GCP Problem Resolution

Initially, two conditions are established regarding the ENs that can potentially serve as candidates for the placement of service i. The first condition (32) ensures that each EN must satisfy the resource constraints (28) and the service bandwidth constraints (30).

$$\begin{cases} Z_{i,z}^{d} + \sum\limits_{i' \in \mathcal{C} \setminus i} \alpha_{i'}^{n} Z_{i',z}^{d} \leq Z_{n,z}^{c} \quad z \in \mathcal{Z} \\ \max\limits_{p \in \mathcal{P}_{n \to N_i^t}} \left\{ \mathcal{B}_{n \to N_i^t}^{p} \right\} \geqslant B_i^{ser} \end{cases} \tag{32}$$

Based on this first condition (32), we can define the next set $E_i(\alpha)$ associated with CS i as the collection of candidate ENs with their minimum hop count to target node N_i^t .

$$E_i(\alpha) = \left\{ \left(n, \min_{p \in \mathcal{P}_{n \to N_i^t}} \left\{ \mathcal{H}_{n \to N_i^t}^{p} \right\} \right) ; n \in \mathcal{N} \text{ and (32) is satisfied} \right\} \tag{33}$$

A tuple (n, h) is included in this set if EN n is reachable from target node N_i^t with a hop count h and satisfies the conditions in (32).

Then, using $E_i(\alpha)$, we define a second condition to identify a refined set $\mathcal{E}_i(\alpha, \Delta_h)$ of candidate placement ENs relevant to CS i. This condition use a threshold Δ_h to ensure that the minimum number of hops between each potential node in $E_i(\alpha)$ and the target node N_i^t is less than Δ_h. If no nodes satisfy this condition, the set will instead comprise the node from the set $E_i(\alpha)$ that is closest to N_i^t, regardless of this condition, provided such node exist. Since an empty subset signifies an unavoidable penalty for the associated service, this condition guarantees that the set $\mathcal{E}_i(\alpha, \Delta_h)$ is empty only when $E_i(\alpha)$ is also empty. Moreover, the use of the threshold Δ_h is primarily driven by the need to restrict the solution search space, thus excluding trivially non-feasible combinations. Additionally, it serves the purpose of maintaining control over the execution time, particularly in scenarios involving non-feasible configurations.

Accordingly, and in relation to all services, we define a general vector $\mathcal{E}(\alpha, \Delta_h)$ containing all subsets $\mathcal{E}_i(\alpha, \Delta_h)$ as follows:

$$\mathcal{E}(\alpha, \Delta_h) = \{ \mathcal{E}_i(\alpha, \Delta_h) \ / \ i \in \mathcal{S} \} \tag{34}$$

Subsequently, the total number $\chi(\alpha, \Delta_h)$ of placement possibilities for "not yet penalized" services can be derived as:

$$\chi(\alpha, \Delta_h) \leftarrow \prod_{\{x \in \mathcal{E}(\alpha, \Delta_h) \text{ and } x \neq \emptyset\}} (|x|) \tag{35}$$

Now that we've established these key concepts, let's delve into the solution implementation. Here's the Algorithm 1, named GSPA, that outlines the steps involved:

Algorithm 1 : Global Service Placement Algorithm (GSPA)

Require: $\mathcal{S}$, $\mathcal{N}$, $\mathcal{C}$, $\mathcal{K}$, $\mathcal{P}$, $\mathcal{M}$, Ω, $\mathcal{D}$ and Δ_h.
Ensure: Global solution α^*, β^*, γ^*, δ^* with cost $Cost^*$

1: $Cost^* \leftarrow \infty$;
2: build $\mathbf{E}^0 = \mathcal{E}(\mathbf{0}, \Delta_h)$ using (34);
3: $N \leftarrow \chi(\mathbf{0}, \Delta_h)$ using (35);
4: **for** $l = 0$ à $N - 1$ **do**
5: **for** each service i in $\mathcal{S}$ **do**
6: $\alpha_i \leftarrow \mathbf{0}$;
7: **if** $|\mathbf{E}_i^0| = 0$ **then**
8: $\delta_i \leftarrow 0$; //service i penalization
9: **else**
10: $\delta_i \leftarrow 1$; //no penalization of service i
11: n_i is the node within $\mathbf{E}_i^0$ at index $\left(l \ mod \ |\mathbf{E}_i^0|\right)$;
12: $\alpha_i^{n_i} \leftarrow 1$; // service i placement at node n_i
13: $l \leftarrow \ l \ div \ |\mathbf{E}_i^0|$;
14: **end if**
15: **end for**
16: $(\beta, \gamma, \delta, X) \leftarrow subSolution(\alpha, \delta)$;
17: **if** $X < Cost^*$ **then**
18: $(\alpha^*, \beta^*, \gamma^*, \delta^*, Cost^*) \leftarrow (\alpha, \beta, \gamma, \delta, X)$
19: **end if**
20: **end for**
21: **return** $(\alpha^*, \beta^*, \gamma^*, \delta^*, Cost^*)$

In the outer for loop (line 4), all feasible placements are iterated over, with each iteration involving the construction of the placement vector α and the initial penalization decisions δ (lines 5 to 15). Subsequently, the CSPDP sub-problem is resolved, and the current solution along with its cost are updated (lines 16 to 19).

B. CSPDP Sub-Problem Resolution

In this subsection, the process of solving this sub-problem is demonstrated. Three solutions are proposed and detailed: The first involves an exact solution based on a Brute Force Search (BFS) method, while the remaining two utilize approximate solutions based on Simulated Annealing (SA) and Markov Approximation (MA) methods, respectively.

1) Brute-force-search-based scheme: To determine the optimal chunks duplicate Transfer decisions provided by the placement decisions α, we conduct an exhaustive search across all potential solutions using a Brute Force Search for Chunks Duplicates Collection, denoted as BFS-CDCA. Algorithm 2 presents this solution. An exhaustive search over all possible combinations of chunk duplicates and paths is conducted with respect to the placement decision α. This search aims to identify the optimal solution $(\beta^*, \gamma^*, \delta^*)$ with the minimum cost S^*. The cost function is computed based on the decisions made for α, β, γ, and δ, considering the constraints of problem $\mathcal{CSPDP}$. The algorithm iterates, evaluates and updates the optimal solution whenever a lower cost is encountered. The function $minPenalisation$ facilitates the determination of the penalty vector that minimizes the overall objective function Θ, considering that the decisions α, β, and γ have already been established. Finally, the algorithm returns the intermediate optimal solution $(\beta^*, \gamma^*, \delta^*)$ along with the corresponding cost S^*.

Algorithm 2 Brute Force Search based CBIC Distributed Collection Algorithm (BFS-CDCA)

Require: $\mathcal{S}$, $\mathcal{N}$, $\mathcal{C}$, $\mathcal{K}$, $\mathcal{P}$, $\mathcal{M}$, Ω, $\mathcal{D}$ and α.
Ensure: Intermediate solution $(\beta^*, \gamma^*, \delta^*)$ with cost S^*;
1: $S^* \leftarrow \infty$
2: $N \leftarrow$ combinations count;
3: **for** $l = 0$ à $N - 1$ **do**
4: construct decisions vectors β, γ from l;
5: $\delta \leftarrow minPenalisation(\alpha, \beta, \gamma)$
6: **if** constraints of $\mathcal{CSPDP}$ are satisfied **then**
7: $S \leftarrow \Theta(\alpha, \beta, \gamma, \delta)$ according to (23);
8: **if** $S < S^*$ **then**
9: $(\beta^*, \gamma^*, \delta^*, S^*) \leftarrow (\beta, \gamma, \delta, S)$
10: **end if**
11: **end if**
12: **end for**
13: **return** $(\beta^*, \gamma^*, \delta^*, S^*)$

2) Simulated-annealing-based scheme: In Algorithm 3, the proposed heuristic solution based on simulated annealing is introduced. Simulated annealing, a widely utilized optimization technique, is renowned for its simplicity, general applicability, and efficiency compared to alternative methods.

Algorithm 3 uses a probabilistic approach that allows for potential degradation in cost to prevent being trapped in local minima. It utilizes a cost function as an analogy to the energy of a thermodynamic system. Throughout the iteration in the solution space, the acceptance of the current state depends on whether the new state has lower energy. If the new state has higher energy, acceptance is probabilistic, determined by a temperature parameter and the Boltzmann distribution. As the temperature decreases, the system becomes less likely to accept higher energy states. This adjustment in the probability of accepting a penalizing transition seeks a balance between exploring new solutions and exploiting known solutions in the space.

Algorithm 3 : Simulated Annealing based CBIC Distributed Collection Algorithm (SA-CDCA)

Require: $\mathcal{S}$, $\mathcal{N}$, $\mathcal{C}$, $\mathcal{K}$, $\mathcal{P}$, $\mathcal{M}$, Ω, $\mathcal{D}$, Δ, L^{max}, T_0 and α.
Ensure: Intermediate solution $(\beta^*,\gamma^*,\delta^*)$ with cost S^*;
1: Generate initial decisions (β,γ)
2: $\delta \leftarrow minPenalisation(\alpha,\beta,\gamma)$;
3: $S^* \leftarrow \Theta(\alpha,\beta,\gamma,\delta)$ according to (23);
4: **for** l=1 to L^{max} **do**
5: $T \leftarrow T_0 e^{-0.5 l \frac{1}{2\Delta}}$;
6: $\beta' \leftarrow rand\_neighbour(\beta)$;
7: $(\gamma',\delta') \leftarrow bestTransfer(\alpha,\beta')$;
8: **if** constraints of $\mathcal{CSPDP}$ are satisfied **then**
9: Calculate $S'=\Theta(\alpha,\beta',\gamma',\delta')$ using 23;
10: $\Delta_S \leftarrow S' - S$
11: **if** $\Delta_S<0$ or $e^{\frac{-|\Delta_S|}{T}} \geqslant random(0,1)$ **then**
12: $(\beta,\gamma,\delta,S) \leftarrow \left(\beta',\gamma',\delta',S'\right)$
13: **if** $S<S^*$ **then**
14: $(\beta^*,\gamma^*,\delta^*,S^*) \leftarrow (\beta,\gamma,\delta,S)$
15: **end if**
16: **end if**
17: **end if**
18: **end for**
19: **return** $X^*=(\beta^*,\gamma^*,\delta^*,S^*)$

The algorithm enters a simulated annealing loop (line 4), where it iterates over a specified number of iterations L^{max}. In each iteration, the temperature T is updated based on the current iteration l, and a random neighbor solution β' is generated. The corresponding best γ' is then determined using β'. The penalty vector δ' for the new solution is obtained by minimizing penalization. If the constraints of problem $\mathcal{CSPDP}$ are satisfied, the cost S' is calculated, and a decision is made based on the Metropolis criterion to accept or reject the new solution. If accepted, the current solution is updated, and if it improves the global solution, it is recorded. Finally, the algorithm returns the global intermediate solution $X^* = (\beta^*, \gamma^*, \delta^*)$ along with the corresponding cost S^*.

3) Markov approximation-based scheme: Markov Approximation is a technique used in optimization to approximate complex problems by simplifying them into a Markov chain model. It is particularly useful for problems with large solution spaces where exact methods become computationally infeasible. In this method, the problem is represented as a Markov decision process, where each state corresponds to a possible solution, and transitions between states are determined by a stochastic process based on the problem's constraints and objectives. By iteratively updating the probabilities of transitioning between states, the algorithm converges towards an optimal or near-optimal solution.

a) Log-sum-exp approximation: Now, we delve into a Log-sum-exp method aimed at approximating the mathematical expression of the obtained problem. Let the configuration $c = \{\alpha; \beta; \gamma; \delta\} \in \mathcal{S}_f^\alpha$ be a solution, where $\mathcal{S}_f^\alpha$ is the Feasible Solution Set (FSS) of problem $\mathcal{CSPDP}$ with objective

function Θ^{α}. Sure, problem $\mathcal{CSPDP}$ is equivalent to:

$$\min_{c\in\mathcal{S}_f^{\alpha}} \Theta^{\alpha}(c) \tag{36}$$

According to Appendix A of [28], by associating a probability p_c with the adoption of a configuration c, the optimal solution of the problem $\max_{c\in\mathcal{S}_f^{\alpha}} \Theta^{\alpha}(c)$ is the same as that of the problem $\max_{p\geq 0} \sum_{c\in\mathcal{S}_f^{\alpha}} p_c\Theta^{\alpha}(c)$ where $\sum_{c\in\mathcal{S}_f^{\alpha}} p_c = 1$. It follows that the optimal solution of problem $\mathcal{CSPDP}$ is the same as that of the following problem:

$$\min_{p\geq 0} \sum_{c\in\mathcal{S}_f^{\alpha}} p_c\Theta^{\alpha}(c) \text{ subject to } \sum_{c\in\mathcal{S}_f^{\alpha}} p_c = 1. \tag{37}$$

Here $p = (p_c)_{c\in\mathcal{S}_f^{\alpha}}$ is a probability distribution associated with the possibility universe $\mathcal{S}_f^{\alpha}$.

Subsequently, the formulation of the log-sum-exp approximation can be derived such that [29]:

- Firstly, for any strictly positive constant τ, we have:

$$0 \leq \min_{c\in\mathcal{S}_f^{\alpha}} \Theta^{\alpha}(c) + \tfrac{1}{\tau}\ln\left(\sum_{c\in\mathcal{S}_f^{\alpha}} e^{-\tau\Theta^{\alpha}(c)}\right) \leq \tfrac{\ln\left|\mathcal{S}_f^{\alpha}\right|}{\tau} \tag{38}$$

- Secondly, let g_{τ} be the log-sum-exp function defined on $\mathbb{R}^m$ by:

$$g_{\tau}(x_1;...;x_m) = \frac{1}{\tau} ln(\sum_{1\leq i\leq m} e^{\tau x_i}) \tag{39}$$

  where $m = |\mathcal{S}_f^{\alpha}|$. As $\tau \to \infty$ and according to Eq. (38), the approximation gap $\frac{\ln|\mathcal{S}_f^{\alpha}|}{\tau} \to 0$, and thus the proposed approximation in Eq. 38 becomes exact.

- Thirdly, by using the conjugate function of g_{τ} and according to [30]-p.93, we can find that:

$$g_{\tau}(x) = \max_{\left\{p\geq 0 \text{ s.t. } \sum_{1\leq i\leq m} p_i=1\right\}} \sum_{1\leq i\leq m} p_i x_i - \tfrac{1}{\tau} p_i ln(p_i) \tag{40}$$

- Finally, the approximation defined by Eq. 40 is also the solution to the following optimization problem:

$$\min_{\left\{p\geq 0 \text{ s.t. } \sum_{c\in\mathcal{S}_f^{\alpha}} p_c=1\right\}} \sum_{c\in\mathcal{S}_f^{\alpha}} p_c\Theta^{\alpha}(c) + \frac{1}{\tau}\sum_{c\in\mathcal{S}_f^{\alpha}} p_c ln(p_c) \tag{41}$$

b) The Markov chaine: : In this subsection, the Metropolis-Hastings algorithm will be employed to construct an irreducible Markov chain $(X_n)_{n\geq 0}$ with state space S_f. This chain will have p^* as its reversible probability distribution, thereby ensuring its stationarity.

Firstly, the Lagrangian of problem (41) is given by:

$$L(p,\lambda) = \sum_{c\in\mathcal{S}_f^{\alpha}} p_c\Theta^{\alpha}(c) + \frac{1}{\tau}\sum_{c\in\mathcal{S}_f^{\alpha}} p_c\ln(p_c) + \lambda\left(\sum_{c\in\mathcal{S}_f^{\alpha}} p_c - 1\right) \tag{42}$$

where λ is the Lagrange multiplier [29].

Secondly, solving the Karush-Kuhn-Tucker conditions yields the following two equations $\sum_{c\in\mathcal{S}_f^{\alpha}} p_c^* = 1$ and $\Theta^{\alpha}(c) + \frac{1}{\tau}(\ln({p_c}^*) + 1) + \lambda^* = 0\left(\forall c \in \mathcal{S}_f^{\alpha}\right)$, where $({p_c}^*)_{c\in S_{\alpha}}$ is the optimal solution of the primal problem and λ^* is the optimal solution of the dual problem. Thus, from the second equation we can get:

$$p_c^* = e^{-\tau(\Theta^{\alpha}(c)+\lambda^*)-1}; \forall c \in \mathcal{S}_f^{\alpha} \tag{43}$$

This result and the condition $\sum_{c\in\mathcal{S}_f^{\alpha}} p_c^* = 1$, give the result:

$$\lambda^* = \frac{1}{\tau}\left(\ln\left(\sum_{c\in\mathcal{S}_f^{\alpha}} e^{-\tau\Theta^{\alpha}(c)}\right) - 1\right) \tag{44}$$

Finally, combining Eq. (43) and (44) leads to the following probability distribution:

$$p_c^* = \frac{e^{-\tau\Theta^{\alpha}(c)}}{\sum_{u\in\mathcal{S}_f^{\alpha}} e^{-\tau\Theta^{\alpha}(u)}}; \forall c \in \mathcal{S}_f^{\alpha} \tag{45}$$

here p_c^* represents the optimal solution of problem (Eq. 41) and thus a quasi-optimal solution of problem (Eq. 36).

c) MA-CDCA algorithm: Our problem-solving process starts with constructing an initial configuration, denoted as c_0. Once c_0 is established, we employ the Metropolis-Hastings algorithm to create an irreducible Markov chain, represented by $(X_n)_{n\geq 0}$. This Markov chain has the desirable property of converging to a specific stationary probability distribution, denoted by p^*. The core of the algorithm lies in a probabilistic approach based on the Metropolis-Hastings principle, which is described in detail next as follows :

if at time t, we have $X_t = c \in \mathcal{S}_f^{\alpha}$, a state c' is randomly drawn from $\mathcal{S}_f^{\alpha}$ according to the distribution $q_{c,c'}$ given by:

$$q_{c,c'} = \frac{p_c^* + p_{c'}^*}{|\mathcal{S}_f^{\alpha}| - 1} \tag{46}$$

then we calculate the acceptance probability $A_{c,c'} = min\left\{\frac{p_{c'}^* q_{c',c}}{p_c^* q_{c,c'}}; 1\right\}$ simplified as:

$$A_{c,c'} = min\left\{e^{-\tau(\Theta^{\alpha}(c')-\Theta^{\alpha}(c))}; 1\right\} \tag{47}$$

Hence, we accept the transition $X_{t+1} = c'$ with probability $A_{c,c'}$ and reject it with probability $1 - A_{c,c'}$.

The pseudo-code of the solution is presented in Algorithm 4.

Algorithm 4 : Markov Approximation based CBIC Distributed Collection Algorithm (MA-CDCA)

Require: $\mathcal{N}$, $\mathcal{C}$, $\mathcal{K}$, $\mathcal{P}$, $\mathcal{M}$, Ω, $\mathcal{D}$, T^{max} and α.
Ensure: Intermediate solution X^*=(β^*,γ^*,δ^*);
1: Generate an initial chunks selection (β_0) based on α;
2: (γ_0,δ_0)←$bestTransfer(\alpha,\beta_0)$;
3: Build the initial solution X_0=(α,β_0,γ_0,δ_0);
4: **for** t=0 to T^{max} **do**
5: Select CS i such that $CS\_Gain(X_t,i)$>0;
6: Generate a chunks selection $\beta_i(t+1)$ related to CS i such that $Chunk\_Gain(X_t,i,\beta_i(t+1))$>0;
7: Update $\beta^{'}$ according to shift $\beta_i(t)\rightarrow\beta_i(t+1)$;
8: ($\gamma^{'}$,$\delta^{'}$)←$bestTransfer(\alpha,\beta^{'})$;
9: Build the new configuration $c^{'}\leftarrow(\alpha,\beta^{'},\gamma^{'},\delta^{'})$
10: Calculate acceptation probability $A_{X_t,c'}$ using 47;
11: **if** $A_{X_t,c'}$>random(0,1) **then**
12: Accept the new transition $X_t\rightarrow c'$ and set X_{t+1}=c'
13: **else**
14: Reject the new transition $X_t\rightarrow c'$ and set X_{t+1}=X_t
15: **end if**
16: **end for**
17: **return** X^*=$X_{T^{max}+1}$

VIII. Evaluation

To evaluate the proposed solutions, we conducted a series of experiments using a range of metrics. These experiments were meticulously designed to test each solution under various conditions, ensuring a comprehensive assessment.

A. Simulation Setup

All the experiments were conducted on a personal computer running Windows 10, equipped with a 2.4GHz Intel Core i5 processor and 16GB of RAM. This setup ensured a consistent and controlled environment for testing, minimizing the potential influence of hardware variability on the performance metrics.

The parameters of the simulation experiments are detailed in Table III. These parameters were meticulously chosen to reflect realistic and challenging conditions for the proposed solutions.

TABLE III. Simulation Parameters

Parameter	Values
σ_s;σ_n;σ_i; D_i	variable
σ_r	3 resource types: CPU, RAM, Storage
$\lvert\mathcal{P}_{n\rightarrow n'}\rvert$	$\in [\![3;5]\!]$
τ	10^{10}
W_r; W_h	0.5; 0.5
W_d; W_b; W_l	0.5; 0.5; 0.5
L^{Bloc}	10.0 MB
$a_{i,j}$	$\in [5,30] \times 10^{-3}$ s/MB
$b_{i,j}$	$\in [1,5] \times 10^{-3}$ s/MB
$c_{i,j}$	$\in [1,10] \times 10^{-4}$ s

B. Cost and Execution Time Analysis

This section presents a comparative analysis of the three solutions: BFS-CDCA, MA-CDCA, and SA-CDCA focusing on their normalized cost and execution time under varying conditions. In this analysis, we consider a scenario with 8 services (users) and a variable count of edge nodes σ_n ranging from 2 to 9. For all services i, a duplication factor $D_i = 2$ is used and two settings for chunks fragmentation are considered, $\sigma_i \in \{3;5\}$, and these values are used as suffixes in the names of each solution to denote their specific configurations. For example, MA-CDCA-3 and MA-CDCA-5 refer to the Markov Approximation-based solution with a chunk fragmentation settings of $\sigma_i = 3$ and $\sigma_i = 5$, respectively.

Fig. 2 shows how the average total cost changes with the number of edge nodes.

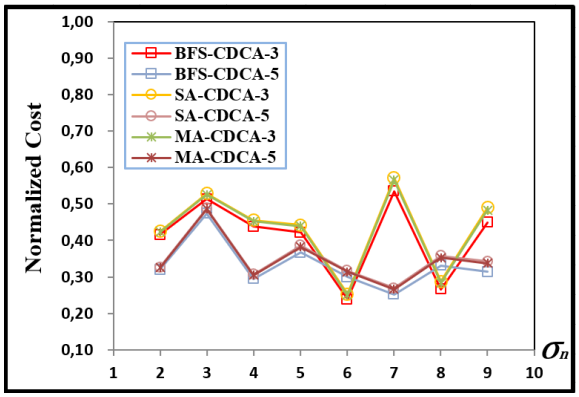

Fig. 2. Normalized Cost; $\sigma_s = 8$, $D_i = 2$, $\sigma_i \in \{3;5\}$.

The results indicate a significant similarity between the three solutions' performance, particularly for $\sigma_n \in \{2;3;4\}$ across both settings of σ_i. Furthermore, for $\sigma_c \in \{5;6;7;8;9\}$, the experiment demonstrates that both MA-CDCA and SA-CDCA solutions achieve costs within 1.1% and 1.3% margin of difference compared to the exact BFS-CDCA solution, respectively.

Likewise, Fig. 3 shows how average execution time changes with the number of edge nodes σ_n.

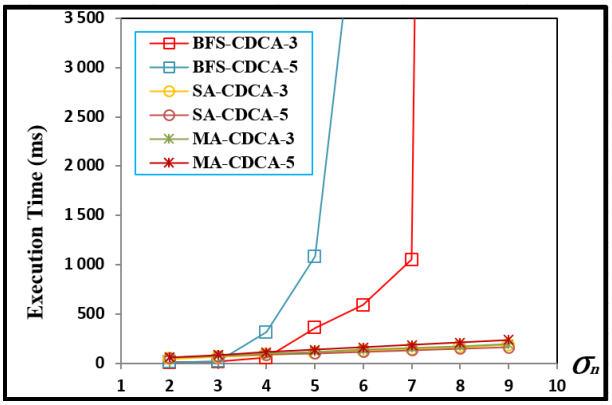

Fig. 3. Execution Time; $\sigma_s = 8$, $D_i = 2$, $\sigma_i \in \{3;5\}$.

The trend in Fig. 3 is confirmed by the data. Execution time increases with the number of edge nodes σ_n, particularly for the BFS-CDCA solution, which exhibits exponential growth. In contrast, the MA-CDCA and SA-CDCA solutions demonstrate stable execution times. For example, with a fragmenta-

tion parameter σ_i of 3 and 9 edge nodes, MA-CDCA-3 and SA-CDCA-3 require only $210.0ms$ and $229.0ms$, respectively, while the exact BFS-CDCA solution reaches impractically high execution times of $6452677.1ms$. Similarly, with a fragmentation parameter of 5 and 9 edge nodes, MA-CDCA-5 and SA-CDCA-5 require only $320.0ms$ and $389.0ms$, respectively, whereas the exact BFS-CDCA solution reaches impractically high execution times of $10877142.4ms$.

While the number of edge nodes σ_n has the most significant impact on execution time, the results also reveal the influence of the fragmentation parameter σ_i. As σ_i increases, all solutions show a trend of requiring more execution time. This suggests that careful consideration must be given to the fragmentation parameter σ_i to avoid additional processing overhead for the adopted solutions.

C. Service Provision Analysis

Now, this section examines the satisfaction rate for service provision requests, focusing on three main factors: fragmentation, duplication, and service priority. The influence of these factors is analyzed to determine their effects on the overall performance of both the system and the proposed solutions.

1) Chunks count and duplication factors: In this experiment, the impact of fragmentation on request satisfaction rates is investigated. This is achieved by controlling two factors related to each service: the per-service duplication factor D_i and The chunk count factor σ_i. In essence, the experiment examines how dividing services into smaller chunks (fragmentation) affects service requests satisfaction. For each service i, the D_i factor is set in $\{1; 3\}$) while the chunk count was varied from $\sigma_i = 1$ (where only a single piece of service data is considered) to $\sigma_i = 8$. Also, the number of services is set to either $\sigma_s = 8$ or $\sigma_s = 16$. These services are deployed across 10 edge nodes ($\sigma_n = 10$) and are assigned equal priority. Moreover, considering the significant impact of data size on the studied timing metric, the CSs' data sizes A_i are generated within the range of [1, 8] MB. For the configuration where $\sigma_s = 8$, the obtained average data size A_i is 4.6 MB, whereas for the $\sigma_s = 16$ configuration, the average is 5.7 MB.

Fig. 4 shows how the Average Collection Time of the selected chunks changes with the number of chunks.

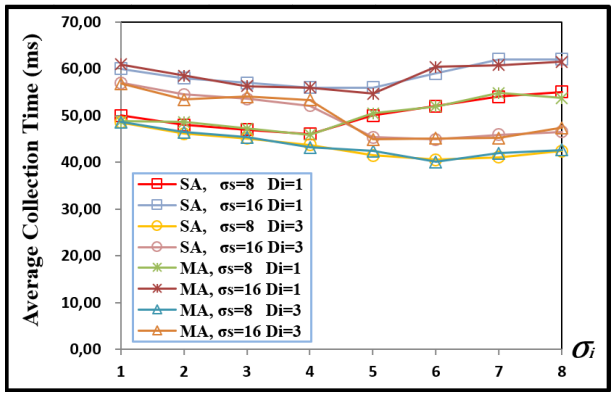

Fig. 4. Average Chunks Collection Time; $\sigma_s \in \{8; 16\}$, $\sigma_n = 10$, $D_i \in \{1; 3\}$.

A preliminary analysis of the results reveals a high degree of similarity between the SA-CDCA and MA-CDCA solutions across most configurations examined. This suggests that both approaches achieve comparable efficiency. For fragmentation levels (σ_i) ranging from 1 to 4, the average collection time consistently decreases across all combinations of service count (σ_s) and duplication factor (D_i). This suggests that a moderate level of fragmentation may improve efficiency in collecting data. In fact, regardless of the service count (σ_s), increasing the duplication factor (D_i) from 1 to 3 consistently reduces the average collection time. This implies that data redundancy introduced by duplication might be beneficial for faster collection. Interestingly, the behavior changes beyond a fragmentation level of 4. Indeed, when $D_i = 1$ (no duplication), the collection time tends to increase regardless of the service count (σ_s). This suggests that excessive fragmentation without redundancy becomes detrimental for collection efficiency. Conversely, with three duplicates of each chunk ($D_i = 3$), the collection time continues to decrease until a fragmentation level of 6, after which it slightly increases. This implies that a higher level of redundancy can tolerate a wider range of fragmentation levels before seeing a performance drop. These observations highlight the importance of considering both fragmentation and duplication when optimizing data collection strategies.

2) Service priority factor: The next experiment investigates the influence of service priority under critical resource limitations, where the available resources can meet the requirements of only two CS. Twenty services, all requiring the same resources, were divided into two groups of ten. The first group (GU) received a uniform priority ($\pi_i = 1$). The second group (GG) received graded individual priorities assigned from 10 (highest) to 1 (lowest). The experiment was conducted over five rounds, with served services removed from the pool for subsequent rounds. This setup allowed the study of the global penalization function, which evaluates the overall penalty incurred due to unmet service requests based on priority levels. Additionally, the obtained average normalized cost was analyzed to understand how the changing priority of the variable service affects the efficiency of service delivery.

Fig. 5 illustrates how the Average Normalized Cost and the overall penalization evolve as the rounds progress.

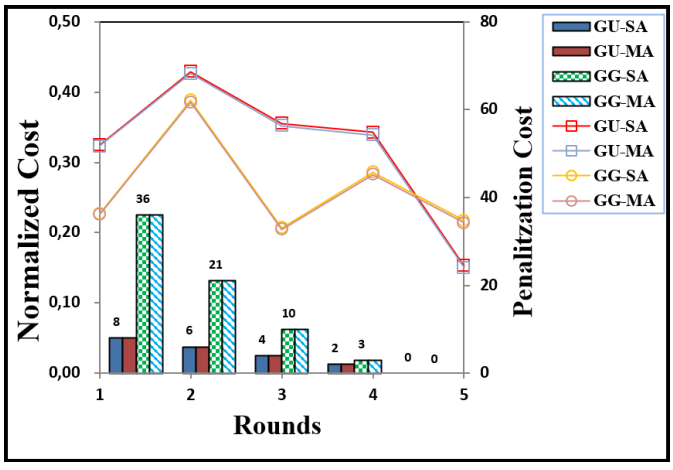

Fig. 5. Normalized cost (line chart), penalization cost (bar chart); $\sigma_s = 20$, $\sigma_n = 12$, $D_i = 2$, $\sigma_i = 2$.

In terms of Normalized Cost, the two algorithms, SA-CDCA and MA-CDCA, yield nearly identical costs for the cost related objective function. However, the MA-CDCA algorithm has a minor edge in terms of cost making it slightly more effective. Furthermore, the penalization cost for all priority groups (GU, GG) and algorithms (SA, MA) consistently decreases across the rounds. This indicates a general improvement in performance over execution rounds in terms of penalized CS. Indeed, in the GU group, where all equipment possesses a uniform priority ($\pi_i = 1$), the consistent decrease in penalization cost by 2 units per round implies that two CS are being served consistently each round. Similarly, in the GG group, the decreasing rate in penalization cost implies that the highest priority equipment is being served consistently, leading to a reduction in penalties. In other words, the adopted prioritization strategy effectively prioritizes high-priority CS.

D. Blockchain-based Caching Analysis

The third experiment investigates how replicating services influences time overhead. To understand the effect of varying fragmentation levels, four different chunks' count scenarios ($\sigma_i = 1$ to $\sigma_i = 4$) are considered, each with 1 and 3 duplicate scenarios for each of the σ_s services ($\sigma_s \in \{8, 16\}$). This experiment aims to analyze how different levels of fragmentation and duplication impact the overall time overhead. Throughout the experiment, all services are assigned the same priority, and a fixed number of edge nodes ($\sigma_n = 10$) is maintained. For each service, the Blockchain-related (BC) Time Overhead is calculated as the maximum operational time across all the involved blockchain nodes, as given by equation 3. The maximum is taken because BC operations are performed in parallel. Then, the total time overhead for the system is obtained by summing these maximum times for all σ_s services.

Fig. 6 shows how the Overall Blockchain-related Time Overhead changes as the fragmentation progresses.

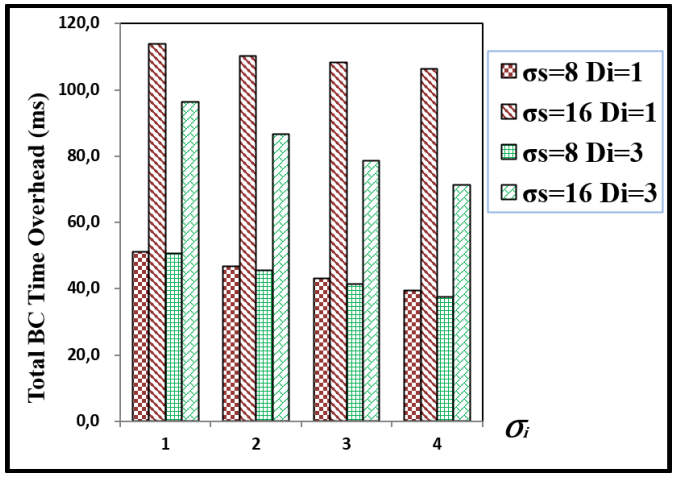

Fig. 6. Overall blockchain-related time overhead; $\sigma_s \in \{8; 16\}$, $\sigma_i \in \{1; 2; 3; 4\}$, $\sigma_n = 10$, $D_i \in \{1; 3\}$.

As the services are divided into an increasing number of chunks (σ_i), the time overhead associated with the blockchain operations decrease. Indeed, with lower fragmentation levels, the time overhead is relatively high due to fewer parallel operations being performed. However, as the fragmentation level increases, more chunks are processed in parallel across the blockchain nodes, resulting in a reduction in processing time. This trend is observed consistently, regardless of the number of services or duplicates involved. Additionally, the total time overhead clearly increases with a higher number of services, as more cache blocs are processed. Conversely, the total time overhead decreases with an increase in the number of duplicates, as the redundant data allows for more flexible decisions regarding the timing characteristics of the caching blockchain nodes. Hence, the results clearly highlight the positive impact of service fragmentation and replication on reducing the time overhead caused by blockchain-related operations. These insights are essential for the development of efficient cache management mechanisms within the investigated blockchain-based systems.

IX. Conclusions and Perspectives

This paper investigates the fusion of Multi-access Edge Computing (MEC) capabilities with blockchain technology to address the low-latency and safety requirements of Smart Devices (SDs). The emphasis was placed on examining the role of a split-duplicate-cache method within a secure blockchain-enhanced MEC system, with the objective of enhancing service provision. An optimization problem was derived and solved with the objective to reduce the backhaul bandwidth and the number of hops between the serving node and the selected deployment node. The experimentation assessed the performance of the three proposed solutions: MA-CDCA, SA-CDCA, and BFS-CDCA. The analysis revealed that BFS-CDCA performed best in smaller-scale settings, while MA-CDCA and SA-CDCA showed efficient execution times, especially in configurations with a high number of CBICs, replicas, ENs, and CSs. Additionally, it was demonstrated that the fragmentation-replication methodology positively impacts the Blockchain operational time overhead.

Future research could focus on evaluating the scalability of MA-CDCA and SA-CDCA in larger, more complex scenarios and assessing their real-world deployment in smart cities and IoT networks. Additionally, exploring the security implications, energy efficiency, and impact on end-user experience could provide valuable insights into the practical benefits and limitations of these solutions.

References

[1] Q.-V. Pham, F. Fang, V. N. Ha, M. J. Piran, M. Le, L. B. Le, W.-J. Hwang, and Z. Ding, "A survey of multi-access edge computing in 5g and beyond: Fundamentals, technology integration, and state-of-the-art," *IEEE access*, vol. 8, pp. 116 974–117 017, 2020.

[2] Y. Hmimz, T. Chanyour, M. El Ghmary, and M. O. Cherkaoui Malik, "Energy efficient and devices priority aware computation offloading to a mobile edge computing server," in *2019 5th International Conference on Optimization and Applications (ICOA)*, 2019, pp. 1–6.

[3] T. Chanyour, Y. Hmimz, M. El Ghmary, and M. O. Cherkaoui Malki, "Delay-aware and user-adaptive offloading of computation-intensive applications with per-task delay in mobile edge computing networks," *International Journal of Advanced Computer Science and Applications*, vol. 11, no. 1, 2020. [Online]. Available: http://dx.doi.org/10.14569/IJACSA.2020.0110190

[4] V. K. Kaliappan, S. Yu, R. Soundararajan, S. Jeon, D. Min, and E. Choi, "High-secured data communication for cloud enabled secure docker image sharing technique using blockchain-based homomorphic encryption," *Energies*, vol. 15, no. 15, p. 5544, 2022.

[5] T. Chanyour and M. O. Cherkaoui Malki, "Deployment and migration of virtualized services with joint optimization of backhaul bandwidth and load balancing in mobile edge-cloud environments," *International Journal of Advanced Computer Science and Applications*, vol. 12, no. 3, 2021. [Online]. Available: http://dx.doi.org/10.14569/IJACSA.2021.0120368

[6] G. Wang, C. Li, Y. Huang, X. Wang, and Y. Luo, "Smart contract-based caching and data transaction optimization in mobile edge computing," *Knowledge-Based Systems*, vol. 252, p. 109344, 2022.

[7] J. Guo, C. Li, and Y. Luo, "Blockchain-assisted caching optimization and data storage methods in edge environment," *The Journal of Supercomputing*, vol. 78, no. 16, pp. 18 225–18 257, 2022.

[8] R. Aghazadeh, A. Shahidinejad, and M. Ghobaei-Arani, "Proactive content caching in edge computing environment: A review," *Software: Practice and Experience*, vol. 53, no. 3, pp. 811–855, 2023.

[9] H. Chai, S. Leng, M. Zeng, and H. Liang, "A hierarchical blockchain aided proactive caching scheme for internet of vehicles," in *ICC 2019-2019 IEEE International Conference on Communications (ICC)*. IEEE, 2019, pp. 1–6.

[10] S. Chen, Y. Zheng, W. Lu, V. Varadarajan, and K. Wang, "Energy-optimal dynamic computation offloading for industrial iot in fog computing," *IEEE Transactions on Green Communications and Networking*, vol. 4, no. 2, pp. 566–576, 2019.

[11] R. Malik and M. Vu, "On-request wireless charging and partial computation offloading in multi-access edge computing systems," *IEEE Transactions on Wireless Communications*, vol. 20, no. 10, pp. 6665–6679, 2021.

[12] Y. Liu, Q. He, D. Zheng, X. Xia, F. Chen, and B. Zhang, "Data caching optimization in the edge computing environment," *IEEE Transactions on Services Computing*, vol. 15, no. 4, pp. 2074–2085, 2020.

[13] Z. Chen and Z. Zhou, "Dynamic task caching and computation offloading for mobile edge computing," in *GLOBECOM 2020-2020 IEEE Global Communications Conference*. IEEE, 2020, pp. 1–6.

[14] G. Zhang, S. Zhang, W. Zhang, Z. Shen, and L. Wang, "Joint service caching, computation offloading and resource allocation in mobile edge computing systems," *IEEE Transactions on Wireless Communications*, vol. 20, no. 8, pp. 5288–5300, 2021.

[15] X. Ma, A. Zhou, S. Zhang, and S. Wang, "Cooperative service caching and workload scheduling in mobile edge computing," in *IEEE INFOCOM 2020-IEEE Conference on Computer Communications*. IEEE, 2020, pp. 2076–2085.

[16] P. Yuan, S. Shao, L. Geng, and X. Zhao, "Caching hit ratio maximization in mobile edge computing with node cooperation," *Computer Networks*, vol. 200, p. 108507, 2021.

[17] S. Zhong, S. Guo, H. Yu, and Q. Wang, "Cooperative service caching and computation offloading in multi-access edge computing," *Computer Networks*, vol. 189, p. 107916, 2021.

[18] H. Feng, S. Guo, L. Yang, and Y. Yang, "Collaborative data caching and computation offloading for multi-service mobile edge computing," *IEEE Transactions on Vehicular Technology*, vol. 70, no. 9, pp. 9408–9422, 2021.

[19] Y. Huang, X. Song, F. Ye, Y. Yang, and X. Li, "Fair and efficient caching algorithms and strategies for peer data sharing in pervasive edge computing environments," *IEEE Transactions on Mobile Computing*, vol. 19, no. 4, pp. 852–864, 2019.

[20] A. Asheralieva and D. Niyato, "Combining contract theory and lyapunov optimization for content sharing with edge caching and device-to-device communications," *IEEE/ACM Transactions on Networking*, vol. 28, no. 3, pp. 1213–1226, 2020.

[21] J. Yin, M. Zhan, Z. Zhang, L. Wang, D. Zhang, and X. Xiao, "Research on the content sharing system for mobile edge caching networks: a hierarchical architecture," in *2022 15th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI)*. IEEE, 2022, pp. 1–6.

[22] S. Zarandi and H. Tabassum, "Federated double deep q-learning for joint delay and energy minimization in iot networks," in *2021 IEEE International Conference on Communications Workshops (ICC Workshops)*. IEEE, 2021, pp. 1–6.

[23] L. Cui, X. Su, Z. Ming, Z. Chen, S. Yang, Y. Zhou, and W. Xiao, "Creat: Blockchain-assisted compression algorithm of federated learning for content caching in edge computing," *IEEE Internet of Things Journal*, vol. 9, no. 16, pp. 14 151–14 161, 2020.

[24] T. Chanyour and A. Kaddari, "Blockchain-based distributed caching with replication for efficient service provision in edge-cloud environments," in *Proceedings of the 6th International Conference on Networking, Intelligent Systems & Security*, ser. NISS '23. Larache, Morocco: Association for Computing Machinery, 2023. [Online]. Available: https://doi.org/10.1145/3607720.3607768

[25] Z. Ma, S. Shao, S. Guo, Z. Wang, F. Qi, and A. Xiong, "Container migration mechanism for load balancing in edge network under power internet of things," *IEEE Access*, vol. 8, pp. 118 405–118 416, 2020.

[26] F. Wilhelmi, S. Barrachina-Muñoz, and P. Dini, "End-to-end latency analysis and optimal block size of proof-of-work blockchain applications," *IEEE Communications Letters*, vol. 26, no. 10, pp. 2332–2335, 2022.

[27] T. Pflanzner, H. Baniata, and A. Kertesz, "Latency analysis of blockchain-based ssi applications," *Future Internet*, vol. 14, no. 10, p. 282, 2022.

[28] W. Pu, X. Li, J. Yuan, and X. Yang, "Resource allocation for millimeter wave self-backhaul network using markov approximation," *IEEE Access*, vol. 7, pp. 61 283–61 295, 2019.

[29] X. Li and C. Zhang, "Semi-dynamic markov approximation-based base station sleep with user association for heterogeneous networks," *IET Communications*, vol. 17, no. 6, pp. 704–711, 2023.

[30] S. P. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004.