Received April 9, 2021, accepted June 20, 2021, date of publication July 7, 2021, date of current version July 19, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3095356

Cooperative Content Caching in MEC-Enabled Heterogeneous Cellular Networks

TADEGE MIHRETU AYENEW, DIONYSIS XENAKIS, NIKOS PASSAS, AND LAZAROS MERAKOS
Department of Informatics and Telecommunications, National and Kapodistrian University of Athens, 157 84 Athens, Greece

Corresponding author: Dionysis Xenakis (nio@di.uoa.gr)

This work was supported by the European Union's Horizon 2020 Research and Innovation Programme through the Marie Skłodowska Curie Grant under Agreement 722788.

ABSTRACT Multimedia content delivery via the cellular infrastructure increases fast due to the very high volumes of mobile video traffic generated by the billions of end devices populating the mobile data network. A critical mass of mobile video content requests refers to the consumption of the same popular video content, which is consumed by different end terminals spanning small geographical regions. Such content requests place a great burden on the backhaul of content-agnostic cellular networks, which fail to exploit the correlation of video requests to decongest their backhaul links. This creates redundant retransmissions while fetching the same video content from a central server to the network edge, using the bandwidth-limited backhaul at peak-time periods. With the integration of multi-access edge computing (MEC) capabilities in 5G mobile cellular networks, mobile network operators can place popular video content closer to the network edge at off-peak time periods, predicting user requests exhibiting a high correlation for a given time interval over smaller geographical regions. In this paper, we investigate popular content placement in multi-tier heterogeneous cellular networks where the edge network infrastructure can cooperate to create content delivery (and placement) clusters to effectively serve correlated video requests. To this end, we model the cooperative content placement problem using the multiple knapsack problem (MKP) formulation and present an exact (optimal) bound-and-bound strategy to solve it. The performance of the proposed strategy is evaluated in-depth using extensive system-level simulations and is compared against that of other state-of-the-art algorithms. Valuable design guidelines and key performance trade-offs are discussed, paving the way towards cluster-based cooperative caching in MEC-enabled cellular network setups.

INDEX TERMS Content caching, cellular networks, HetNets, MEC, multiple knapsack problem, branch-and-bound, dynamic programming.

I. INTRODUCTION

The emergence of new technologies, such as the Internet of things (IoT), machine to machine communications (M2M), e-health, and vehicular communications has pushed the cellular traffic much higher than anticipated [1]. Before the outbreak of the COVID-19 pandemic in 2020, recent reports [2] on cellular network traffic predicted a five-fold increase of the monthly mobile traffic, from 33 exabytes in 2019 to 164 exabytes by 2025, exhibiting an annual growth of 31%. Mobile data traffic is dominated by mobile video content, which in 2020 accounts for 63% of the total traffic, while it is predicted to exceed 76% by 2025, exhibiting an annual growth rate of 30% [2]. The ever-increasing mobile video traffic stems from the fast growth of high-bandwidth demanding multimedia services, such as video on demand (VoD), augmented reality, immersive media formats and the multi-billions of mobile broadband subscriptions. On top of that, online streaming is extensively used to support the daily activities and working routine of humans around the globe, especially after the outbreak of the COVID-19 pandemic, which has accelerated the extensive use of online video streaming services and teleworking.

Although mobile cellular networks are being upgraded to accommodate the offered mobile data traffic load, e.g., through the deployment of new generation fronthaul and backhaul technologies, the exponential increase of mobile video traffic still brings the utilization of existing backhaul

The associate editor coordinating the review of this manuscript and approving it for publication was Giuseppe Araniti.

links to their capacity limits, questioning the scalability of existing systems in light of the ever-increasing demand for mobile video content delivery. Backhaul congestion is aggravated by factors, such as the size of contents, the overall number of video consumers and the characteristics governing video consumption (e.g., screen size, quality of experience - QoE - requirements, video resolution). Mobile video traffic is dominated by multiple requests of the same popular video content, creating overlapping end-to-end video content delivery chains, starting from the edge network terminals and ending to the servers hosting the requested content in the far Internet.

Popular video segments are redundantly re-transmitted from the central content server (residing in the far Internet) to the end user equipment (UE) (residing at the network edge) creating unnecessary utilization of intermediate network resources, especially in heterogeneous cellular networks (HCN) with multiple network tiers and high deployment density [3]. This problem will be further exacerbated in the fifth generation (5G) of mobile cellular networks, where low latency requirements of less than 1ms are targeted for specific service types co-utilizing the wireless medium [4]. To alleviate redundancy upon popular content transmission throughout the cellular infrastructure and meet the high-end performance targets set for 5G and Beyond mobile data networks, the utilization of *edge network caching* has been proposed as a cost-effective capacity-boosting network-layer solution [5], [6].

Edge network caching consists of the *edge content placement* and the *edge content delivery* phases. Edge content placement refers to the process by which the storage of edge network nodes is filled with popular content to reduce the delay and backhaul overheads of fetching popular content from central servers outside the network service domain to the network edge. Edge content delivery refers to the process by which the cached content is delivered to the content consumers (edge users), utilizing multiple wireless links and radio access technologies (RATs). The performance of cache-enabled mobile cellular networks strongly depends on the optimization techniques used to place popular video contents in the edge network and on the mixture of technologies used to deliver the respective content to the end terminals [7]–[9].

Edge content placement and delivery can utilize storage resources available in different types of network elements, including macrocell base stations (MBS), small base stations (SBS), femtocell base stations (FBS) and UEs. Edge content placement and delivery can utilize multi-access edge computing (MEC) capabilities at the network edge, a technology that has been long-viewed as the cornerstone for seamless delivery of personalized video content. For example, MEC nodes can [10] i) predict content popularity to locally store popular video chunks in the near area of mobile users and offload the radio access network (RAN) during on-peak periods (reducing thus the user-perceived latency), ii) employ local transcoding on high-resolution videos to match the screen resolution of mobile terminals (thus, enhancing the user-perceived rate), and iii) move relevant content and context closer to the end user (increasing thus the user-perceived 5G service availability) [11]. In the sequel, we will refer to the cache-enabled (infrastructure) nodes, whose main role is to cache popular video content and relay it to the end users without transcoding, as *mobile helpers* (MHs).

In this paper, we are concerned with popular content placement at the network edge and focus on the challenging scenario where the edge network nodes of the HCN, which form cache-enabled MEC clusters, coordinate the placement of popular video contents and cooperate to deliver the respective content to the edge users within their coverage. On the one hand, node coordination refers to the employment of joint decision making towards an optimized placement of popular video contents over the entire MEC cluster (i.e. creating a joint virtual storage pool of individual caches) without allowing content partitioning (i.e. the entire content is stored per node). On the other hand, cooperation in content delivery assumes that if a requested content is found in the cache of a node belonging to the MEC cluster, within which the user request has been recorded, then the MEC cluster nodes shall cooperate to relay the content to the serving node of the user (i.e. the node where the original request has been made by the user). Accordingly, a cache hit is considered to take place if i) the content request of the user is made to a node belonging to a tagged MEC cluster c and ii) any cache-enabled node belonging to c has cached the requested content.

Without focusing on how the requested content is relayed from a MEC node hosting the content to the MEC node serving the user request, in the sequel, we formulate the MEC-enabled cooperative content placement problem and present an exact algorithm that enables joint content placement into the individual caches of the MEC-enabled cluster nodes. To this end, we use the *cache hit probability* (CHP) as the optimization metric at the cluster level subject to a set of constraints, which includes the placement of entire video files per cluster with no repetition, no content partitioning, and the limited cache sizes of the nodes.

The remainder of the paper is organized as follows. Section II summarizes related works in the area and highlights the key differences of our modeling and optimization approach in light of current state-of-the-art. Section III introduces the system description and the problem formulation for cooperative cluster-based content caching in MEC-enabled HCNs. The proposed content placement strategy is presented in section IV. In section V, we evaluate the performance of the proposed content placement strategy using extensive system-level simulations and provide detailed comparisons with other state-of-the-art algorithms. Section VI contains our conclusions.

II. RELATED WORK AND MOTIVATION

Existing works on modeling and performance analysis of edge network caching may be categorized into two broad classes: *non-cooperative*, which assume that each MH acts autonomously and selects the content to be placed into its cache using its own logic, and *cooperative*, which assumes centrally planned content placement and delivery across multiple network nodes that cooperate to this end. Assuming equal file size for popular contents and random request distribution, the authors in [12], [13] have shown that cooperative caching enables significant improvements on network performance due to the enhanced utilization of edge computing and storage resources. Under the same set of assumptions, a lower download latency and better utilization of storage resources has also been reported in [3]. In our previous work in [14] we have formulated the non-cooperative content placement problem for a single MH using a 0/1 single knapsack problem formulation given a generic file popularity and size distribution for popular video contents.

The authors in [12] have formulated the joint content placement and delivery, showing that it is an NP-hard problem. Accordingly, they have relaxed the original nested dual problem to a mixed integer non-linear program (MINLP) and employed a branch-and-bound method to enable MHs decide on their own the content to be placed in their cache. Although the respective formulation and methodology used to solve the problem is promising, the proposed system-wide centralized optimization of the joint content placement and delivery phases faces significant scalability challenges due to the centralized aggregation of radio network information from all nodes in the system and the required scaling of the proposed solution to a very large number of nodes in the HCN. The authors in [3] have decoupled the original joint content placement and delivery problem by using i) integer linear programming for solving the content placement problem and ii) employing unbalanced assignments for solving the content delivery problem.

Content placement is typically formulated as a constrained problem, which is modeled using different mathematical frameworks. Depending on the proposed placement model, different algorithms are used to solve the proposed formulation. The survey work in [15] reviews the most widely used modeling methods including, among others, game-theoretic, stochastic, and predictive approaches. The authors in [16] have used multiple-choice knapsack problems to model the cooperative content placement problem in large instances. The emphasis was given on caching layered videos using a fully polynomial time approximation (FPTA) algorithm. The authors in [17] have modeled content placement as a reward maximization problem and solved it by reducing to solvable linear programs. Similarly, the authors in [18] have used mixed integer programming to model the content placement problem and employed greedy algorithms to solve it. In [3], the authors have used linear integer programming to model the content placement problem and applied Lagrangian relaxation with hierarchical prime-dual decomposition to decouple its structure into two-levels, enabling solutions using the sub-gradient method.

Greedy and heuristic algorithms are computationally feasible in large-scale networks but exhibit sub-optimal performance in the general case. In [19], the authors describe a cooperative multi-tier caching system that is decomposed into a series of independent knapsack sub-problems, which are solved by using greedy methods. Focusing on adaptive streaming, the authors in [20] have used a polynomial time greedy method to solve a series of knapsack problems, in order to cache different versions of a video at the network edge. Authors in [21], have modeled the content placement problem using stochastic approach and have applied a fully polynomial time approximation method where a random set of contents is cached to the network edge, and the placement probabilities within a tier are considered to be the same. In [22], the authors employ stochastic geometry under fixed cache size and bandwidth constraints, assuming a greedy method to solve the proposed content placement problem.

The authors in [23] have used uncoded caching for the content placement phase (file partitioning without transcoding) and index coding for the content delivery phase, by broadcasting the coded messages. The Lyapunov function is employed in [24] to allow hybrid cloud and edge content caching using greedy and heuristic content placement algorithms. In [25], content placement is modeled as a multiple knapsack problem and dynamic programming (DP) is used to solve the uncoded caching case. Nevertheless, the profit maximization problem still depends on approximations of scaling, before the DP is applied. Additional effort is required to deal with the inhomogeneous content popularity over large geographical areas, the limited storage and network capacity as well as the heterogeneous characteristics of the cache-enabled edge network nodes.

Different from the vast majority of existing approaches, which typically formulate the problem of network-wide content placement given a content popularity distribution, in this work we focus on the emerging scenario where the mobile network operator (MNO) can assign the heterogeneous edge network nodes of the HCN to MEC clusters of different sizes and capabilities, e.g. different number of infrastructure nodes per MEC cluster, different mixture of MEC node types and cache sizes. MEC integration into the Radio Access Network (RAN) will create an overlay layer of MEC service areas, enabling MNOs not only to cope with the different video popularities met across neighbor geographical regions but also to adapt MEC service coverage to the actual topology and capabilities of HCN nodes.

In light of the forthcoming MEC/RAN integration, current literature includes a noteworthy amount of MEC clustering techniques that incorporate different optimization criteria, such as minimizing end-to-end delay of MEC services [26], reducing traffic congestion within the MEC cluster [27], enhancing MEC service coverage [28], or offloading core network traffic to the edge MEC nodes [29]. Given the rich literature in the area, in this work, we choose not to focus on how to form a MEC cluster. Instead, we focus on how to optimize content placement within a given MEC cluster of known size (in terms of number of nodes) assuming potentially different cache sizes across the MEC nodes. To this end, we consider that our strategy is used after the employment of a MEC clustering technique. A key requirement is that MEC clustering is performed such that intra-cluster content exchange is performed at a low transmission cost, forming a robust content delivery path between i) the MEC node serving the video request directly to the end user and ii) the MEC node that hosts the requested video content within the MEC cluster.

Without loss of generality, we consider that content placement and intra-cluster content delivery is centrally coordinated by a MEC cluster head. We further consider that intra-cluster content delivery is performed in response to the video requests, whereas content placement is performed on an *epoch-by-epoch* basis. We define an epoch as the time interval within which the list of popular video files is valid for the entire MEC service area. The size and popularity of popular files are assumed to be known to the MEC cluster head and remain fixed for a given epoch. Current literature includes a large volume of techniques for identifying popular video contents and estimating their popularity distribution within a target area [30].

Different from current state-of-the-art, we formulate and solve a 0/1 multiple knapsack problem (ZOMKP) that integrates the following two practical constraints i) MEC nodes store complete files only (*no file partitioning*) and ii) only one MEC node is allowed to store a given file within the same MEC cluster (*no file repetition*). Both these constraints result in a mathematically more involved problem but are of high practical interest in realistic MEC-enabled setups as explained below. Firstly, adding/removing cached segments of a given file from the cache of multiple MEC nodes comes with increased communication and cache monitoring overheads, whereas the released cache may not be fully utilized (or be sufficient) to cache a new video file. Also, distributed (coded) caching in different nodes of the MEC cluster requires perfect tracking of cached segments and sophisticated multi-source transmission schemes to be implemented within the MEC cluster. Secondly, even though redundant caching of the same file at different MEC nodes increases the availability of popular contents within the same MEC cluster, reducing the intra-cluster transmission cost given a cache hit, also degrades the probability of having a cache hit event in the MEC cluster due to the under-utilization of the available storage resources. Accordingly, the two requirements of *no file partitioning* and *no file repetition* are in line with MEC-enabled service provisioning, which primarily aims to leverage edge network resources and avoid the utilization of end-to-end links to the far Internet through the backhaul network.

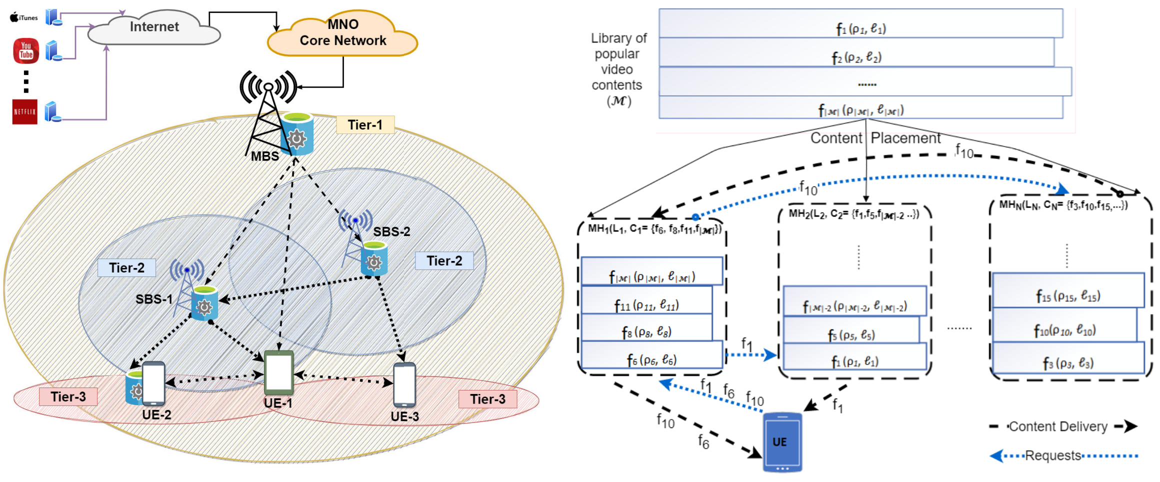
Different from the vast majority of existing works, which typically relax the original problem formulation constraints to strike a good performance trade-off between cache hit efficiency and computation time, in our work we present an exact (optimal) content placement strategy that employs a highly-effective bound-and-bound methodology that explores the full-state space of the problem in a smart fashion. Compared to traditional branch-and-bound methodologies, the proposed bound-and-bound strategy exploits both an upper and a lower performance bound to quickly eliminate sub-optimal solution branches without going deeper into the evaluation of sub-optimal solution branches. Using extensive system level simulations we validate that under realistic MEC system setups, the proposed bound-and-bound content placement strategy is computationally feasible. The main contributions of this paper are summarized as follows:

- We investigate how content placement can be formulated and optimized under the emerging MEC/RAN integration scenario, where the MNO can bundle edge network, computation and storage resources to form joint MEC/RAN service clusters (areas). The formation of MEC service areas shall enable the MNOs to better adapt to i) the local spatiotemporal file popularity in smaller geographical areas and ii) the actual topology and capabilities of HCN nodes.
- We provide a ZOMKP formulation of the cooperative content placement problem within a given MEC cluster, under practical constraints with regard to the heterogeneity of cache sizes of cluster nodes, the no-repetition of popular video files within the same cluster, the non-uniformity of content popularity and the un-coded placement of video files (no file repetition is allowed within the same MEC cluster).
- We propose an exact bound-and-bound search strategy to solve the proposed ZOMKP formulation. A lower bound (LB) is obtained by fixing sequentially files to the caches of MHs using the 0/1 single knapsack problem (ZOSKP) formulation in [14]. An upper bound (UB) is obtained by assuming a virtual aggregate cache pool of size equal to the sum of individual cache sizes of the MHs and employing the ZOSKP solution in [14]. The LB is used to avoid exploration of content placement solution branches exhibiting sub-optimal performance, whereas the UB is used to mitigate unnecessary computations towards the exploration of additional solutions.
- We provide extensive system-level simulations to evaluate the performance of the proposed strategy in MEC-enabled multi-tier HCN, while we also compare its performance to that of the widely-used greedy and random content placement strategies. Moreover, we highlight the key performance trade-offs governing the content placement phase in MEC clusters, while we also derive valuable design guidelines and discuss best practices for MEC cluster formation and content placement in multi-tier MEC-enabled HCNs.

III. SYSTEM MODEL AND PROBLEM FORMULATION

A. SYSTEM DESCRIPTION

We focus on the downlink direction of a multi-tier MEC-enabled HCN, where each tier consists of HCN nodes of similar networking capabilities. We consider that the MNO employs a cluster formation algorithm to group


Internet
MNO Core Network
iTunes
Tier-1
MBS
SBS-2
Tier-2
Tier-2
SBS-1
Tier-3
UE-2
UE-1
UE-3
Tier-3
Library of popular video contents ($\mathcal{M}$)
$f_1(\rho_1, \ell_1)$
$f_2(\rho_2, \ell_2)$
$f_{|\mathcal{M}|}(\rho_{|\mathcal{M}|}, \ell_{|\mathcal{M}|})$
Content Placement
f_{10}
$MH_1(L_1, C_1 = \{f_6, f_8, f_{11}, f_{|\mathcal{M}|}\})$
$MH_2(L_2, C_2 = \{f_1, f_5, f_{|\mathcal{M}|-2} ..\})$
$MH_N(L_N, C_N = \{f_3, f_{10}, f_{15}, ...\})$
$f_{|\mathcal{M}|}(\rho_{|\mathcal{M}|}, \ell_{|\mathcal{M}|})$
$f_{11}(\rho_{11}, \ell_{11})$
$f_8(\rho_8, \ell_8)$
$f_6(\rho_6, \ell_6)$
$f_{|\mathcal{M}|-2}(\rho_{|\mathcal{M}|-2}, \ell_{|\mathcal{M}|-2})$
$f_5(\rho_5, \ell_5)$
$f_1(\rho_1, \ell_1)$
$f_{15}(\rho_{15}, \ell_{15})$
$f_{10}(\rho_{10}, \ell_{10})$
$f_3(\rho_3, \ell_3)$
UE
Content Delivery
Requests

(a) Three-tier system model example. A MEC cluster formed by MBS, SBS-1, SBS-2, UE-2 and UE-3. UE-1 can receive content from MBS, SBS-1, UE-2 and UE-3 (but not from SBS-2 which is out of coverage).

(b) UE issues a video request to MH1 for files f_1, f_6 and f_{10}. MH_1 forwards the request for f_{10} to MH_N, which relays the file to MH_1 using in-cluster content delivery. UE receives f_6 and f_{10} from MH_1, and file f_1 directly from MH_2

FIGURE 1. (a) A system model example with three-tier HCN, (b) A content placement and delivery example.

a heterogeneous set of cache-enabled HCN nodes into MEC clusters that will carry out the edge network caching process (both content placement and delivery). A cache-enabled MEC cluster can be composed by HCN nodes belonging to different network tiers. In Fig. 1(a), we provide an illustrative example of a three-tier system model instance, where UE-1 is out of coverage of SBS-2. Without loss of generality, for each MEC-enabled cluster, we consider that the content placement is centrally planned by a cluster head (e.g. an MBS). Cache-enabled nodes belonging to the same MEC cluster may have different cache sizes (i.e. storage capacity), while they are considered to follow the instructions of the MEC cluster head to fill their cache with the allocated content during off-peak periods.

We focus on the content placement process for a given time period, which we term as the cache epoch. For a tagged cache epoch, we consider that the cluster head concludes on a target list of popular video files with known popularity and file size. The popularity is evaluated on a per epoch and cluster basis, while it can match the user request ratio on a per cluster basis for the given popular video file. MEC cluster heads are considered to deploy their own policy to infer the list of popular files and the popularity distribution governing the list within their coverage area, potentially encompassing information from the MNO's core network (e.g., using techniques similar to those in [30]).

At any time, end users are considered to associate with a serving HCN node, following the network discovery and attachment protocols of the Radio Access Technology employed by the HCN. If the serving HCN is not a cache-enabled node belonging to a MEC cluster, it serves the video request as usual (i.e., end-to-end connection to the content delivery server at the Internet). However, if the serving HCN node is a MEC node, it first examines whether the requested video can be found in its local cache. If a *local cache hit* takes place, the serving MEC node will serve the video request without establishing an end-to-end connection to the content server. If not, the serving MEC node will subsequently seek the requested video content within the MEC cluster it belongs to, e.g., by querying the cluster head, or by utilizing a cache map available to all MHs belonging to the same MEC cluster.

If the content is found in the cache of another MH belonging to the same MEC cluster (*cluster cache hit*), in-cluster content delivery methods shall be deployed to relay the requested video content to the serving HCN node and, finally, to the end user. Provided that files are not partitioned and that a single copy of a popular file can be cached within a given MEC cluster, it follows that in-cluster content delivery comes down to the implementation of a direct, or a multi-hop link between the serving MEC node and the MEC node (MH) hosting the content, depending on the technologies available in the MEC cluster. If the file cannot be found in the cache of any cluster node (including the serving HCN node) then the serving HCN node shall establish an end-to-end connection to the content delivery server, hosting the requested video content. Assuming that intra-cluster MEC content delivery requires significantly lower delay and network overheads compared to an end-to-end connection to the content delivery server in the far Internet, in the sequel, we consider a cache hit event to take place if i) the serving HCN node is part of a MEC cluster and ii) one of the MHs forming the respective MEC cluster has a copy of the requested video content in its cache. Note that the proposed problem formulation and solution apply also when the user requests are served by multiple MEC nodes, i.e. the CHP performance of the cluster is not affected by the methods used for content delivery.

Let us now focus on the content placement process of a given cache epoch and a tagged cache-enabled MEC cluster of interest. Let N denote the number of nodes in the tagged MEC cluster and let L_n denote the cache size (in bits) of the n-th node belonging to the MEC cluster with $n \in \{1, \ldots, N\}$. Omitting indexes relating to the cache epoch, let $\mathcal{M} = \{f_m : 1 \leq m \leq |\mathcal{M}|\}$ denote the set (list of size $|\mathcal{M}|$) of popular video files, where f_m is the identifier of the m-th file belonging to the list of popular video files. We also let ρ_m and s_m denote the (fixed) popularity and size of a tagged file $f_m \in \mathcal{M}$, respectively. Accordingly, we define the set of file sizes $\mathcal{S} = \{s_1, s_2, \ldots s_{|\mathcal{M}|}\}$ and the set of popularity values (i.e. the popularity distribution) $\mathcal{P} = \{\rho_1, \rho_2, \ldots \rho_{|\mathcal{M}|}\}$, where $\sum_{m=1}^{|\mathcal{M}|} \rho_m = 1$. The content popularity values are assumed to be normalized over the set $\mathcal{M}$.

At the beginning of each cache epoch, the MEC cluster head is assumed to be aware of the list of popular files $\mathcal{M}$, the size of popular files in $\mathcal{S}$, the popularity values in $\mathcal{P}$ as well as the number N and cache size values L_n of all mobile helpers in $n \in \{1, \ldots, N\}$. Using those parameters as an input, the cluster head implements its content placement algorithm to decide on the popular files that should be cached in the buffers of all N cluster nodes belonging to its cache-enabled MEC cluster. For a given epoch, the content placement strategy should take into consideration that i) the number of MHs in the cluster is fixed to N, ii) the cache size available per cluster node $n \in \{1, .., N\}$ is fixed and known (L_n), iii) the size and popularity of files in $\mathcal{M}$ are fixed and known (using sets $\mathcal{S}$ and $\mathcal{P}$), iv) no repetition of popular files is allowed across the caches of individual MHs forming the MEC cluster, v) cluster nodes are considered capable to relay content from another cluster node to another to server user requests, and vi) file partitioning is not allowed (i.e. cluster nodes cache entire files and not video chunks).

Assuming that the content placement procedure has concluded, let $\mathcal{C}_n$ denote the set of cached popular files at the n^{th} MH and $\mathcal{C} = \bigcup_{n=1}^{N} \mathcal{C}_n$ the full set of cached files in the MEC cluster, where $\mathcal{C}_n \subseteq \mathcal{M}$ and $\mathcal{C} \subseteq \mathcal{M}$. Given the system model constraints mentioned above, it readily follows that $\mathcal{C}_i \cap \mathcal{C}_j = \emptyset$ for $i, j \in \{1, \ldots, N\}$ $(i \neq j)$ and that $\sum_{c \in \mathcal{C}_n} s_c \leq L_n$. Accordingly, the cluster head is required to maximize the cluster cache hit probability (CHP), a metric that we define on the basis of the popularity distribution characterizing the library $\mathcal{M}$ as follows:

$$\Psi_{\mathcal{C}} = \sum_{n=1}^{N} \sum_{c \in \mathcal{C}_n} \rho_c \tag{1}$$

B. 0/1 MULTIPLE KNAPSACK FORMULATION

Recall that the CHP is defined as the probability that a requested video content is found in the cache of any MH belonging to the MEC cluster where the original video request has been made. In the sequel, we define the index parameter $x_{n,m}$ to denote the realization of the event where the n-the MH of the MEC cluster has cached the popular video file $f_m \in \mathcal{M}$ with $n \in \{1, \ldots, N\}$. Accordingly, $x_{n,m} = 1$ if the event $f_m \in \mathcal{C}_n$ holds true and $x_{n,m} = 0$, otherwise. Provided that file partitioning is not allowed and that MHs belonging to the same cluster cannot cache the same popular video file, it readily follows that $\sum_{n=1}^{N} x_{n,m} \leq 1$ for $1 \leq m \leq |\mathcal{M}|$. Accordingly, we present the ZOMKP formulation in the context of MEC-enabled cluster-based edge content placement in HCNs.

$$\Psi_{\mathcal{C}} = \underset{x_{n,m}, 1 \leq n \leq N, 1 \leq m \leq |\mathcal{M}|}{\arg\max} \sum_{n=1}^{N} \sum_{m=1}^{|\mathcal{M}|} \rho_m x_{n,m} \tag{2a}$$

$$\text{subject to:} \sum_{m=1}^{|\mathcal{M}|} s_m x_{n,m} \leq L_n, \quad 1 \leq n \leq N \tag{2b}$$

$$\sum_{n=1}^{N} x_{n,m} \leq 1, \quad 1 \leq m \leq |\mathcal{M}| \tag{2c}$$

$$x_{n,m} \in \{0,\ 1\},\ 1 \leq n \leq N, 1 \leq m \leq |\mathcal{M}| \tag{2d}$$

$$\sum_{m=1}^{|\mathcal{M}|} \rho_m \leq 1 \tag{2e}$$

In (2a), the $x_{n,m}$ values $(1 \leq n \leq N,\ 1 \leq m \leq |\mathcal{M}|)$ should be adapted by the content placement strategy so as to maximize the cluster-wide CHP $\Psi_{\mathcal{C}}$ under the constraints of Eqs.(2b)-(2e). Eq. (2b) formalizes the constraint that the total size of video files placed per MH cannot exceed its cache size limit. Eq. (2c) formalizes the constraint that a popular video file cannot be cached in more than one MHs belonging to the same MEC cluster (no file repetition). Eq. (2d) ensures that file partitioning is not allowed (i.e. no intermediate values are enabled). Eq. (2e) follows by construction of the file popularity (i.e. the ρ_m values are probability values summing up to one for all $f_m \in \mathcal{M}$).

Note that by removing the constraint of Eq. (2e) we can allow the popularity values of ρ_m $(m \in \{1, \ldots, |\mathcal{M}|\})$ to take arbitrary values and not be normalized as probabilities in view of a popularity distribution over the state space $\mathcal{M}$. However, in such an occasion, the proposed CHP metric should be considered as a utility (reward) function rather than a probability and be revised accordingly. For easy reference, Table 1 summarizes the notation used for the main system model parameters.

IV. PROPOSED SOLUTION

The ZOMKP is an NP-hard problem [8]. In this section, we modify an exact *bound-and-bound* method that was originally proposed in [31] and adapt it to the context of our formulation in Eq. (2). The proposed bound-and-bound content placement strategy is a modification of a typical branch-and-bound state space exploration technique that progressively fills the cache of a tagged MH taking into consideration both an upper and a lower performance bound to

TABLE 1. Paper notation.

Symbol(s)	Definition
f_m	File index of the m^{th} content in $\mathcal{M}$
$\mathcal{M}$	Library of popular video files for the target cache epoch
ρ_m	Popularity of file $f_m \in \mathcal{M}$ [0,1]
$\mathcal{P}$	Popularity vector of all contents in $\mathcal{M}$
s_m	File size of m^{th} content (bits)
N	Total number of MHs constituting the cluster
L_n, L	Cache size of the n^{th} MH (bits)
L	Cache size of the cluster $L = \sum_1^N L_n$ (bits)
$\mathcal{C}_n$	Cached contents at the n^{th} MH
$\mathcal{C}$	Cached contents in the tagged cluster $\mathcal{C} = \cup_1^N \mathcal{C}_n$
$\|\mathcal{M}\|$, $\|\mathcal{C}\|$	Number of contents in $\mathcal{M}$ and $\mathcal{C}$, respectively
Ψ_C	Cache hit probability

avoid the evaluation of inefficient solution branches. As a branch we consider a content placement chain where i) the placement of a given number of files has been fixed to the cache of some MHs and ii) multiple options exist on fixing the remaining files to the cache of some MHs with non-full buffer. In the sequel, we consider that the MHs of the MEC cluster under scope are sorted in increasing order of their cache size, i.e., $L_1 \leq L_2 \leq \ldots \leq L_N$, and that the popular videos in library $|\mathcal{M}|$ are sorted in decreasing order of their 'popularity per size unit', i.e., $\rho_1/s_1 \geq \rho_2/s_2 \geq \ldots. \geq \rho_{|\mathcal{M}|}/s_{|\mathcal{M}|}$.

The proposed strategy calculates the upper CHP performance bound of the original problem assuming the ideal scenario where files are placed into a single knapsack of size $L = \sum_{n=1}^{N} L_n$. This scenario corresponds to the case where all MHs of the MEC cluster form a virtual aggregate pool of their cache resources, where files can be partitioned within the underlying physical caches of MHs but only full files will be cached in the aggregate MEC cache. The aforementioned problem is equivalent to the ZOSKP for which optimal solution algorithms exist [14], [32]. A lower performance bound is also calculated by independently fixing files into the caches of MHs progressively (i.e. the MHs are filled one by one, on increasing order of cache sizes) by using an algorithm solving the individual ZOSKP formulations assuming that popular files of the previous iterations are removed from $\mathcal{M}$.

Based on the lower and upper performance bounds, the proposed strategy is capable of rejecting specific solutions by i) progressively placings files into the cache of MHs, ii) calculate the corresponding upper and lower performance bounds conditioned on the aforementioned placements, and iii) rejecting solution branches that exhibit poor CHP performance through the use of a backtracking solution process. A different stack data structure is used per MH to enable effective backtracking of the partial solution and the exploration of new solution branches (including the re-calculation of upper and lower performance bounds).

In the following subsections, we explain the building blocks of the proposed bound-and-bound (BaB) content placement strategy that aims to solve the ZOMKP. Section IV-A discusses an exact content placement strategy of the ZOSKP, based on our previous work in [14]. The respective strategy is used to derive upper and lower bound for the CHP in section IV-B, where the proposed content placement strategy for the ZOMKP formulation is also presented. More details on why the proposed content placement strategy is exact and optimal can be found in [31], [32].

A. EXACT STRATEGY FOR THE 0/1 SINGLE KNAPSACK PROBLEM

Content placement to the cache of a single MH can be modeled by the widely known 0/1-Knapsack problem and optimally solved using dynamic programming (DP) [14], [32]. A DP method breaks one problem into multiple sub-problems and solves them sequentially. In [14], we have presented an exact algorithm that utilizes a two-dimensional array structure V of size $(|\mathcal{M}| + 1)$ x $(L + 1)$, which is used to track the different solution branches and infer on the optimal solution to the problem. Each column of V corresponds to a 'unit size' of the available MH's cache, whereas each row of V to the evaluation of a given file $f_m \in \mathcal{M}$. The values of $V[m, j]$ record the highest CHP value that can be attained for the sub-problem where the first m popular files are to be placed into a cache of a single MH j cache with size j units (e.g. MBs). Accordingly, the optimal CHP value is given by $V[|\mathcal{M}| + 1, L + 1]$.

Apart from obtaining the optimal CHP value, a backtracking process should be also performed on V to identify the optimal content placement solution, i.e. which files should be cached per MH. Starting from $V[|\mathcal{M}| + 1, L + 1]$, the backtracking process skips rows (of the current column) that have the same value with $V[|\mathcal{M}|+1, L+1]$. When a different value is found, the file corresponding the respective row is included in the solution and the current column is updated by shifting an equal number of columns with the size of the respective file. The process continues using the same methodology and concludes when no different CHP values can be found in V.

Algorithm 1 provides the pseudocode of the content placement strategy used to solve the ZOSKP. The algorithm uses as an input a library of popular videos $\mathcal{M}$, a vector of corresponding popularities $\mathcal{P}$, a vector of corresponding file sizes $\mathcal{S}$ and the available cache size L. The output of the algorithm is the optimal content placement for the ZOSKP problem, which we denote by Ψ^* and an index vector x of size $|\mathcal{M}|$ that indicates whether file m is selected ($x_m = 1$), or not ($x_m = 0$). Steps 4-5 are used to initialize the algorithm parameters, assuming that the call $X = Zeros(x_1, \ldots, x_k)$ initializes the k-dimensional vector X of size $x_1\ xx_2 \ldots xx_k$ with zeros. Steps 5-15 calculate the optimal CHP value in a content-by-content fashion, whereas steps 15-27 conclude the algorithm by backtracking the two-dimensional array V to identify the file allocation x_m leading to the derived optimal CHP Ψ^*.

B. EXACT STRATEGY FOR THE 0/1 MULTIPLE KNAPSACK PROBLEM

In this subsection we present an enumerative strategy where the caches of MHs are filled progressively, evaluating the

Algorithm 1: Exact Solution for a Single MH

```
Input: $\mathcal{M}, \mathcal{P}, \mathcal{S}, L$
Output: $\Psi^*_{\mathcal{C}_n}, x$
Function: DP-ZOSKP ($\mathcal{M}, \mathcal{P}, \mathcal{S}, L$)
    $V = Zeros(|\mathcal{M}|, N)$;
    $x = Zeros(|\mathcal{M}|)$;
    for $1 \leq m \leq |\mathcal{M}|$ do
        for $0 \leq j \leq L$ do
            if $j \geq s_m$ then
                $V[m, j] = max(V[m-1, j], \rho_m + V[m-1, j-s_m])$;
            else
                $V[m, j] = V[m-1, j]$;
            end
        end
    end
    $\Psi^* = V[|\mathcal{M}|, L]$;
    $m = |\mathcal{M}| + 1, j = L + 1$;
    $temp = V[m, j]$;
    while $m > 0$ && $j > 0$ do
        if $temp \neq V[m-1, j]$ then
            $x_m = 1$;
            $j = j - s_m$;
            $m = m - 1$;
            $temp = V[m, j]$;
        else
            $m = m - 1$;
        end
    end
    return $\Psi^*, x$
end
```

MHs in an ascending order based on their available cache size (i.e. $L_1 \leq L_2 \leq \ldots \leq L_N$). In each iteration, the proposed strategy evaluates whether a popular video file m, which has not yet been assigned to a previous MH, will be cached to the current MH, or not. In each iteration, the strategy evaluates the files to be placed in the cache of MH n. Files that have been currently evaluated for MH n are stacked into a data structure D_n. A two-dimensional vector $\hat{x}$ is used to store the current (partial) solution of the ZOMKP instance, where $\hat{x}_{n,m} = 1$ if file m is (temporarily considered to be) cached in MH n and $\hat{x}_{n,m} = 0$, if not. When the content placement strategy evaluates MH n, the caches of MHs $1, \ldots, n-1$ are assumed to be fully filled while the caches of MHs $n, \ldots, N$ will be empty.

The upper performance bound is calculated using Algorithm 2 (section IV-B1), whereas the lower performance bound by using Algorithm 3 (section IV-B2). The proposed content placement strategy is implemented by Algorithm IV-B (section), which combines the outputs of Algorithms 1-3. The calculation of both upper and lower performance bounds is based on the use of single ZOSKP implemented by Algorithm 1.

1) UPPER BOUND CALCULATION

Algorithm 2 is used to calculate the upper CHP performance bound conditioned on the placement of a given subset of popular video files from $\mathcal{M}$ according to the allocation vector $\hat{x}$. In the sequel, we use the notation $\{L_n\}$ to denote the array of values taken by L_n and the notation $\{D_n\}$ to denote the arry of values taken by the stacks D_n, where $n \in \{1, \ldots, N\}$. The allocation vector $\hat{x}$ is a two-dimensional array of size N by $|\mathcal{M}|$ that indicates whether file m is allocated to the cache of the MH n, or not. At this point, it is important to note that Algorithm 2 is called in intermediate steps of the proposed bound-and-bound content placement strategy to evaluate the CHP potential of solutions branches conditioned on the placement of a subset of files according to $\hat{x}$. However, since the proposed strategy is enabled to call Algorithm 2 when the cache of the currently evaluated MH i is not yet fully filled, the identifier i is also passed as an input to calculate the cache size remaining for the respective MH. Recall that by this step, the proposed strategy will have filled the caches of MHs $1, \ldots, i-1$ and part of the i-th MHs cache.

Algorithm 2: Upper CHP Bound Calculation

```
Input: $\mathcal{M}, \mathcal{P}, \mathcal{S}, \{L_n\}, \{\mathcal{D}_n\}, \hat{x}, i$
Output: $\Psi_U$
Function: UB-MKP ($\mathcal{M}, \mathcal{P}, \mathcal{S}, \{L_n\}, \{\mathcal{D}_n\}, i, \hat{x}$)
    $\bar{L} = \sum_{n=i}^{N} L_n - \sum_{m \in \mathcal{D}_i} s_m \cdot \hat{x}_{im}$;
    $\bar{\mathcal{M}} = \{f_m \in \mathcal{M} : \hat{x}_{nm} = 0 \text{ for } n = 1, \ldots, i\}$;
    $\bar{\mathcal{P}} = \{\rho_m \in \mathcal{P} : \hat{x}_{nm} = 0 \text{ for } n = 1, \ldots, i\}$;
    $\bar{\mathcal{S}} = \{s_m \in \mathcal{S} : \hat{x}_{nm} = 0 \text{ for } n = 1, \ldots, i\}$;
    $[\bar{\Psi}_U, \sim] = DP - ZOSKP\left(\bar{\mathcal{M}}, \bar{\mathcal{P}}, \bar{\mathcal{S}}, \bar{L}\right)$;
    $\Psi_U = \bar{\Psi}_U + \sum_{n=1}^{i} \sum_{m \in \mathcal{D}_n} \hat{x}_{nm} \rho_m$;
    return $\Psi_U$;
end
```

In step 4, Algorithm 2 calculates the available cache size of MHs that have not yet been examined (i.e. helpers $i + 1, \ldots, N$) adding the residual cache size of MH i based on the placement of files depicted in the two-dimensional vector $\hat{x}$. In step 5, the algorithm identifies the set of popular videos in $\mathcal{M}$ that have not been cached to MHs at previous steps (including the current MH i). Steps 6 and 7 filter the popularity and size of video files in $\bar{\mathcal{M}}$ that have not yet been placed in any MH.

Accordingly, Algorithm 2 uses the set of non-cached video files $\bar{\mathcal{M}}$, together with the respective sets of popularity and file sizes $\bar{\mathcal{P}}$ and $\bar{\mathcal{S}}$, respectively, to deploy the ZOSKP algorithm 1 on the respective sub-problem (i.e. 0/1 allocation of files in $\bar{\mathcal{M}}$ into a single knapsack of size $\bar{L}$). In step 9, Algorithm 2 calculates the current achievable ultimate upper performance bound of the ZOMKP formulation taking into consideration the CHP following from the placement described by the allocation vector $\hat{x}$. To this end, step 9 adds i) the CHP obtained for the ZOSKP sub-problem with library $\bar{\mathcal{M}}$ and single knapsack of size $\bar{L}$ to ii) the

CHP following from the placement of files performed in previous steps according to $\hat{x}$.

2) LOWER BOUND CALCULATION

The calculation of the lower bound (Ψ_L) is based on solving $N - i + 1$ independent ZOSKP formulations sequentially given the allocation vector $\hat{x}$ and the identifier of the i-the MH to which the cached is not yet fully filled. In each round, the Algorithm 3 removes from the library files that have been placed to previous MHs and a new ZOSKP sub-problem is defined. Algorithm 3 implements the aforementioned procedure to calculate a tight lower CHP performance bound conditioned on the content placement described by the allocation vector $\hat{x}$. The solutions obtained by the aforementioned logic are stored into a two-dimensional array $\tilde{x}$ and are provided as an output of the algorithm.

Algorithm 3: Lower CHP Bound Calculation

```
1  Input: $\mathcal{M}, \mathcal{P}, \mathcal{S}, \{L_n\}, \{\mathcal{D}_n\}, \hat{x}, i$
2  Output: $\Psi_L, \tilde{x}$
3  Function: LB-MKP $(\mathcal{M}, \mathcal{P}, \mathcal{S}, \{L_n\}, \{\mathcal{D}_n\}, \hat{x}, i)$
4      $\bar{\Psi}_L = \sum_{n=1}^{i} \sum_{f_m \in \mathcal{D}_n} \rho_m \cdot \hat{x}_{nm}$;
5      $\hat{\mathcal{M}} = \{f_m \in \mathcal{M} : \hat{x}_{nm} = 0, n = 1, \ldots i\}$;
6      $\bar{\mathcal{M}} = \hat{\mathcal{M}} \setminus \mathcal{D}_i$;
7      $\bar{\mathcal{P}} = \{\rho_m : \rho_m \in \mathcal{P} \text{ and } f_m \in \bar{\mathcal{M}}\}$;
8      $\bar{\mathcal{S}} = \{s_m : s_m \in \mathcal{S} \text{ and } f_m \in \bar{\mathcal{M}}\}$;
9      $\bar{L} = L_i - \sum_{f_m \in \mathcal{D}_i} s_m \cdot \hat{x}_{im}$;
10     for $n{=}i$ to $N$ do
11         $\left[\tilde{\Psi}_L, y\right] = DP - ZOSKP(\bar{\mathcal{M}}, \bar{\mathcal{P}}, \bar{\mathcal{S}}, \bar{L})$;
12         $\Psi_L = \bar{\Psi}_L + \tilde{\Psi}_L$;
13         for $1 \le m \le |\mathcal{M}|$ do
14             $\tilde{x}_{nm} = y_m$;
15         end
16         $\bar{\mathcal{M}} = \hat{\mathcal{M}} \setminus \{f_m : \tilde{x}_{nm} = 1\}$;
17         $\bar{\mathcal{P}} = \{\rho_m : \rho_m \in \mathcal{P} \text{ and } f_m \in \bar{\mathcal{M}}\}$;
18         $\bar{\mathcal{S}} = \{s_m : s_m \in \mathcal{S} \text{ and } f_m \in \bar{\mathcal{M}}\}$;
19         $\bar{L} = L_{n+1}$;
20     end
21     return $\Psi_L, \tilde{x}$;
22 end
```

In step 4, Algorithm 3 evaluates the baseline CHP according to the content placement described by $\hat{x}$, which refers on the allocation of files into the caches of MHs $1, \ldots, i$. In step 5 the algorithm identifies the set of files that have not yet been placed to previous MHs (i.e. $1, \ldots, i - 1$). In steps 6-9, the algorithm makes necessary initializations to run the ZOSKP algorithm for the i-th MH, by excluding files that have already been considered for MH i and are included in D_i(step 6), filtering the popularity (step 7) and file size (step 8) vectors, and taking into consideration only the residual cache available at the i-the MH (step 9).

In steps 10-19, Algorithm 3 performs a MH-per-MH assignment of files to the caches of MHs $i, \ldots, N$. In more detail, steps 11-12 are used to solve the ZOSKP sub-problem for the n-th MH and to add the respective part of the MEC-wide CHP to the lower bound Ψ_L, steps 13-15 enable storing of the LB solution to the $\tilde{x}$ vector, whereas steps 14-19 update the input parameters $\bar{\mathcal{M}}$, $\bar{\mathcal{P}}$, $\bar{\mathcal{S}}$ and $\bar{L}$ for the next MH taking into consideration the content placement that took place during step n of the for loop.

3) PROPOSED BOUND-AND-BOUND CONTENT PLACEMENT STRATEGY

The proposed content placement strategy implements a recursive enumeration method where the lower and upper performance bounds are used to omit content placement branches that lead to sub-optimal performance. In the sequel, we term the proposed strategy as the Bound and Bound 0/1 Multiple Knapsack Problem (BB-ZOMKP) strategy and present it in the pseudocode form of Algorithm IV-B. The proposed BB-ZOMKP strategy consist of four main parts: *Initialize* (steps 4-8), *BaselinePlacement* (steps 9-24), *ContentPlacement* (steps 25-40) and *Backtrack* (steps 41-56).

The *Initialize* part sets necessary strategy parameters (steps 5-6) and evaluates the ultimate upper CHP performance bound (steps 7-8), which is derived using the optimal ZOSKP allocation (Algorithm 1) assuming the original file library $\mathcal{M}$ and a single knapsack of a total $L = \sum_{n=1}^{N} L_n$ (i.e. files can be segmented within the MEC cluster but as a total, the MEC cluster contains only full files). The *BaselinePlacement* part is responsible for calculating the lower performance bound Ψ_{LB} and the corresponding file placement in $\tilde{x}$ (step 10), given a problem instance where the MHs' caches are partially filled (ZOKMP sub-problem). The respective fixing is described by the D_n stacks and the allocation vector $\hat{x}$, which are both adapted during the *ContentPlacement* part of the strategy (see below).

If the content placement solution found by the LB-MKP algorithm results in improved CHP performance ψ (step 12), then the solution is included in the final allocation vector x (steps 12 -17). Accordingly, if the CHP of the respective solution matches the ultimate upper performance bound Ψ_{UB}, the proposed strategy concludes to avoid unnecessary computations (steps 18-20). If not, the strategy continues and verifies whether the current lower and upper performance bounds match and if so, the Backtrack part of the strategy is called (steps 21-23). This step enables the strategy to improve the upper performance bound Ψ_U and explore solutions with higher CHP performance gains (i.e. closer to the ultimate upper bound Ψ_{UB}.

The *ContentPlacement* part of the strategy is the place where the solution derived from the LB calculation during the *BaselinePlacement* process is implemented by stacking the respective files into the cache of each MH sequentially. MHs are examined in ascending caching size (steps 26, 38 and 39), while the files fixed per MH are selected based on the highest available 'value per unit size' order (steps 27 and 29). Recall that by construction MHs are ordered in ascending cache size and files are ordered in descending 'value per

Algorithm 4: Content Placement Strategy

```
1  Input: $\mathcal{M}, \mathcal{P}, \mathcal{S}, \{L_n\}$
2  Output: $\Psi_C^*, x_{nm}$
3  Function: BB-ZOMKP ($\mathcal{M}, \mathcal{P}, \mathcal{S}, \{L_n\}$)
4      [Initialize]
5      for $n = 1$ to $N$ do $\mathcal{D}_n = \emptyset$ end;
6      $\hat{x} = Zeros(N, |\mathcal{M}|); \psi = 0; i = 1;$
7      $\Psi_U = UB - MKP(\mathcal{M}, \mathcal{P}, \mathcal{S}, \{L_n\}, \{\mathcal{D}_n\}, \hat{x}, i);$
8      $\Psi_{UB} = \Psi_U;$
9      [BaselinePlacement]
10     $[\Psi_L, \tilde{x}_{nm}] = LB - MKP(\mathcal{M}, \mathcal{P}, \mathcal{S}, \{L_n\}, \{\mathcal{D}_n\}, \hat{x}, i);$
11     if $\Psi_L > \psi$ then
12         $\psi = \Psi_L; x = \hat{x};$
13         for $n = i$ to $N$ do
14             for $m$=1 to $|\mathcal{M}|$ do
15                 if $\tilde{x}_{nm} = 1$ then $x_{nm} = 1$; end
16             end
17         end
18         if $\psi = \Psi_{UB}$ then
19             $\Psi_C^* = \Psi_{UB}$; return $\Psi_C^*, x_{nm}$;
20         end
21         if $\psi = \Psi_U$ then
22             Go to Backtrack
23         end
24     end
25     [ContentPlacement]
26     repeat
27         $\mathcal{F} = \{\delta : \tilde{x}_{i\delta} = 1\};$
28         while $\mathcal{F} \neq \emptyset$ do
29             $m = \min\{\delta : \delta \in \mathcal{F}\};$
30             $\mathcal{F} = \mathcal{F} \setminus \{m\};$
31             $\mathcal{D}_i = Push(\mathcal{D}_i, f_m);$
32             $\hat{x}_{im} = 1;$
33             $\Psi_U = UB - MKP(\mathcal{M}, \mathcal{P}, \mathcal{S}, \{L_n\}, \{\mathcal{D}_n\}, \hat{x}, i);$
34             if $\Psi_U \leq \psi$ then
35                 Go to Backtrack;
36             end
37         end
38         $i = i + 1;$
39     until $i$=$N$;
40     $i = i - 1;$
41     [Backtrack]
42     repeat
43         while $\mathcal{D}_i \neq \emptyset$ do
44             Let $m$ be content on top of $\mathcal{D}_i$;
45             if $\hat{x}_{im} = 0$ then
46                 $\mathcal{D}_i = Pop(\mathcal{D}_i, f_m)$
47             else
48                 $\hat{x}_{im} = 0;$
49                 $\Psi_U =$
                     $UB - MKP(\mathcal{M}, \mathcal{P}, \mathcal{S}, \{L_n\}, \{\mathcal{D}_n\}, \hat{x}, i);$
50                 if $\Psi_U > \psi$ then
51                     Go to BaselinePlacement;
52                 end
53             end
54         end
55         $i = i - 1;$
56     until $i$=0;
57 end
```

unit size'. Every time a file m is placed (fixed) into the cache of MH n (steps 30-32), the upper CHP performance bound is re-calculated (step 33) and if it is lower than the current CHP performance ψ (step 34), part of the solution will be backtracked using the Backtrack part of the algorithm (steps 34-36). The procedure resumes after corrective measures are taken using the Backtrack logic, potentially running recursively part of the strategy again to identify a better content placement solution.

The *Backtrack* part of the strategy is called under different occasions, including the ones that we have discussed above. The main role of Backtrack is to remove allocations from the fixed MHs' stacks D_n that are shown to deteriorate the CHP performance of the conditional (partial) solution to the problem as depicted by the allocations D_n (steps 44-48). To this end, Backtrack starts removing files from the MHs' cache that triggered the Backtrack call (e.g. through the Content-Placement trigger in line 36) and then evaluates if the upper bound CHP performance is improved through this action (step 49). If not, the Backtrack logic continues to remove files from the D_n stacks of the MHs (steps 41, 55 and 56) until the *BaselinePlacement* call is triggered to take corrective measures (steps 50-52). The proposed bound-and-bound content placement strategy terminates either when a solution is shown to attain the ultimate CHP probability Ψ_{UB} (step 20), or when no further improvements can take place on the CHP ψ attained by the current solution x (step 57).

V. NUMERICAL RESULTS

In this section, we assess the performance of the proposed content placement strategy and compare it to that of two widely used strategies: the random and the greedy content placement strategies. Different variants of the two strategies are found in current state-of-the-art, e.g. for random in [21], [36]–[38] and for greedy in [19], [20], [22], [34], [35]. For our performance comparisons, we have considered the following variants in view of the ZOMKP formulation. The random strategy arbitrarily selects content from the library $\mathcal{M}$ of popular contents and places it without repetition to the caches of all MHs belonging to the cluster. The greedy strategy orders video files of $\mathcal{M}$ in descending order based on their popularity and places them in the cache of MHs sequentially without repetition, i.e. the cache of the first MH is filled by skipping files that cannot fit, then the cache of the second MH is filled with the remaining files, etc., until no other file can fit into the cache of any MH. Similar strategies have been considered in [21] and [33], respectively. In both the random and greedy strategies, the MHs are ordered based on their available cache size in ascending order, filling the caches of MHs with smaller cache size first.

We investigate the performance of the proposed, random and greedy content placement strategies using extensive system-level simulations in MATLAB. To this end, we consider a multi-tier HCN that includes two types of cache-enabled MHs: small base stations (SBSs) and femto base stations (FBSs). Accordingly, we consider a MEC cluster of a fixed number of N_1 SBSs and N_2 FBSs. The cache size of each SBS is set according to a truncated normal distribution of mean cache size ξ_1 GBs with standard deviation σ_1 GBs. On the other hand, the cache size of each SBS is set assuming a truncated normal distribution of mean cache size ξ_2 GBs with standard deviation σ_2 GBs. Thus, for a tagged SBS MH i the cache size is given as $L_i = \mathcal{N}(\xi_1, \sigma_1^2)$, whereas for a tagged FBS MH j the cache size is given as $L_j = \mathcal{N}(\xi_2, \sigma_2^2)$ with $i, j \in \{1, \ldots, N\}$. Accordingly, the total cache size available in the cluster is distributed as follows $L \sim \mathcal{N}\left(N_1 \cdot \xi_1 + N_2 \cdot \xi_2, N_1 \cdot \sigma_1^2 + N_2 \cdot \sigma_2^2\right)$.

Regarding the library of popular files $\mathcal{M}$ we consider that it includes a total of $|\mathcal{M}|$ popular videos. The size of each video is assumed to follow an exponential distribution of mean size μ GBs (i.e. $s_m \sim \exp(\mu)$ for $m \in \mathcal{M}$). This gives us on the average a total size (of all popular videos) of $E\left[\sum_{m=1}^{|\mathcal{M}|} s_m\right] = |\mathcal{M}| \cdot \mu$. The popularity distribution of video files in $\mathcal{M}$ is assumed to follow the widely-accepted Zipf distribution according to which, the popularity value ρ_m of the m^{th} most popular content in library $\mathcal{M}$ is given by Eq. (3), where $\gamma \geq 0$ is the Zipf parameter (a.k.a. popularity skewness) and $m \in \{1, \ldots, |\mathcal{M}|\}$.

$$\rho_m = \frac{m^{-\gamma}}{\sum_{j=1}^{|\mathcal{M}|} j^{-\gamma}} \tag{3}$$

Note that as the value of γ increases, a smaller amount of videos in $\mathcal{M}$ exhibit high popularity and thus, the popularity is concentrated to a smaller number of videos. On the other hand, as the value of γ decreases, the popularity is distributed in more videos and thus, a larger number of videos of lower popularity are encountered. Unless differently stated, the values for the main simulation parameters discussed above are fixed as in Table 2.

TABLE 2. Simulation parameter values.

Parameters	Values	Distribution
Number of SBS (N_1)	20	Fixed
Number of FBS (N_2)	100	Fixed
Mean SBS cache size (ξ_1)	$20 \cdot \xi_2$ GBs	Normal
SBS cache size variance σ_1	10 GBs	Normal
Mean FBS cache size (ξ_2)	10 GBs	Normal
FBS cache size variance σ_2	2 GBs	Normal
Library size $\lvert\mathcal{M}\rvert$	5000 video files	Fixed
Mean content size (μ)	4GB	Exponential
Popularity parameter γ	1.0	Zipf

According to the average bit rate values recommended by Google for video uploads in YouTube [39] and the NTT-DOCOMO guideline for video delivery in mobile data networks in [40], the mean content size depends on the video codec type, the video bitrate in Mbps and the type of the application (e.g. gaming, music videoclips). In Table 3, we summarize some common YouTube video types and their respective size, whereas in Table 4 we overview the limitations adopted by major social media providers on the upload of videos to their platforms. According to the values presented in this table, we fix the mean content size value μ to 4GB, which corresponds to an HDR (4K) YouTube video of display resolution 3840×2160 (2160p) at High Frame Rate and length of approximately 8 minutes at bit rate 66Mbps. Such types of videos currently dominate the Internet traffic. We further fix the Zipf skewness parameter to $\gamma = 1.0$, which corresponds to the scenario where 10% of video contents to account for 75% of the popularity (i.e. their popularity probabilities sum up to 0.75).

TABLE 3. Sample content statistics calculated from YouTube video upload recommendation.

Codec type	Frame Rate	Video Bitrate (Mbps)	Type of application	
			Gaming, films, (18.9 min) average	VIP people, music video clips (8.2 min) average
SDR(4K) (2160p)	Standard Frame Rate	35-45	5.67 GB	2.46 GB
	High Frame Rate	53-68	8.6 GB	3.7 GB
HDR(4K) (2160p)	Standard Frame Rate	44-56	7.1 GB	3GB
	High Frame Rate	66-85	10.7GB	4.6GB
Audio bitrate =192Kbps is considered for all video bitrates				

TABLE 4. Major social media video limits.

Social Media Platform	Max. Video Size (GB)	Maximum Duration	Maximum Resolution
Facebook	10 GB	240 min.	4096x2048
Instagram	4 GB	1 min.	1080x1080
Pinterest	2 GB	30 min	640x640
Snapchat	1 GB	no limit	1080x1290
YouTube	128 GB	12 hrs.	3840x2160
LinkedIn	5 GB	10 min.	1920x1080

A. IMPACT OF THE MH MEAN CACHE SIZE

We begin the performance evaluation of the three content placement strategies for an increasing mean SBS cache size ξ_1. Note that in our simulations, we have set the FBS mean cache size parameter ξ_2 to be proportional to the respective SBS mean cache size value ξ_1 as follows $\xi_1 = 20 \cdot \xi_2$ (Table 2); thus, an increase to the mean SBS cache size also increases proportionally both the FBS cache size and the total cache size available in the MEC cluster under scope (i.e. $L \sim \mathcal{N}\left(\xi_1(N_1 + 20 \cdot N_2), N_1 \cdot \sigma_1^2 + N_2 \cdot \sigma_2^2\right)$).

In Fig. 2 we plot the cluster-wide CHP for increasing SBS cache size ξ_1 under three different mean file sizes μ. Recall that the cache size of the SBS MH i is distributed according to a normal distribution $L_i \sim \mathcal{N}(x_1, \sigma_1^2)$ and that the file size of popular videos follows an exponential distribution $s_m \sim \exp(\mu)$. As expected, when the library size of popular videos is fixed ($\mathcal{M} = 5000$ - Table 2), an increasing cache size at the SBS MHs improves the average CHP of the cluster for all content placement strategies. The improvement of the

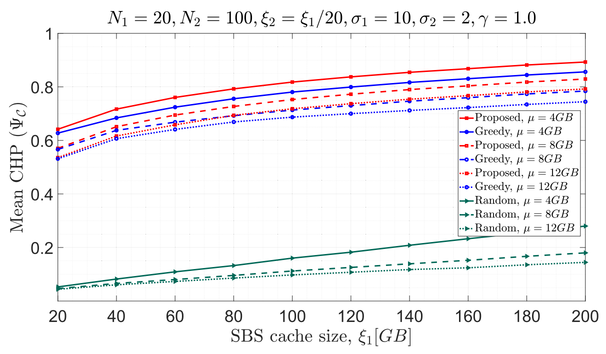


FIGURE 2. CHP performance for increasing SBS cache size and different file sizes.

CHP for the random strategy is linear due to the random utilization of the available cache size, whereas a fast increase is observed for the proposed and greedy strategies in low ξ_1 values, indicating that both strategies can better exploit the additional buffer available to the cluster by the SBS MHs.

For the given set of fixed system parameters and a fixed mean file size μ, the performance of the proposed and greedy strategies is similar when the mean SBS cache size ξ_1 is small (e.g. $\xi_1 \leq$ 50GBs). Nonetheless, the respective performance gap grows rapidly as the SBS cache size increases to higher values for all μ values under scope. The performance improvement attained by the proposed strategy can be explained as follows. The full state space of the ZOMKP can be viewed as a content placement tree, where each e2e branch is a solution instance fixing specific files to the cache of specific MHs. The number of solutions (size of the state space) can be derived by taking into consideration all potential file placements in the available cache of MHs (depth varies depending on the file and cache sizes). Solutions that involve the same sequence of (file-MH) placements share a common route in the content placement tree, i.e. they are ancestors to the same solution branch. Each content placement strategy finds its way to the solution tree according to its logic. The random strategy moves randomly from a parent tree node to an ancestor, whereas the greedy strategy always places the file with the current highest popularity until the caches of MHs are unable to fit more files.

Accordingly, the greedy placement strategy moves through the solution space by appending into the solution a file placement with the current highest popularity value, attaining a sequence of local optimums in a greedy fashion. In contrast, the proposed strategy explores the solution tree by moving in depth to assess the CHP performance of every solution branch, using upper / lower performance bounds to eliminate sub-branches (including their ancestor solutions) with sub-optimal CHP performance and backtracking the current optimal solution by moving back to the solution space tree whenever necessary. In this fashion, the proposed strategy is capable of exploring the full state space in a smart fashion, identifying the optimal content placement and avoiding unnecessary calculations.

The superior performance of the proposed strategy is also highlighted by Fig. 3 as well where, instead of the CHP, we plot the utilization of the available cache in a cluster-wide scale under different mean file size values. Observer in Fig. 3 that the proposed strategy attains a utilization close to 100% under all scenarios under scope, whereas the greedy and random strategies are shown to leave a small amount of cache resources underutilized (i.e. cluster cache utilization lower than 1). For the given set of fixed system parameters (Table 2), Fig. 3 illustrates that the impact of the mean file size μ is negligible to the utilization of the available cluster-wide cache size, i.e., the different μ values considered in the legend result in similar performance for each strategy.

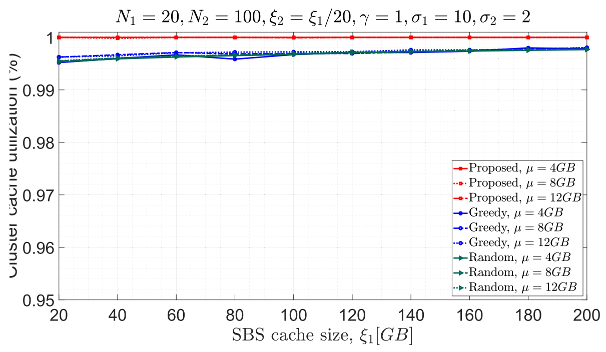


FIGURE 3. Percentage of total used cache size for increasing SBS cache size and different file sizes.

Regarding the combined impact of the mean file size μ parameter and the mean SBS cache size ξ_1, Fig. 2 illustrates that a lower mean file size μ (e.g., $\mu = 4GB$ vs. $\mu = 12GB$) enables all strategies to attain a higher CHP, especially when the mean SBS cache size ξ_1 increases. This directly follows from the fact that for a fixed number (and size) of $\mathcal{M}$, a larger cache size shall increase the probability of all strategies to make a better allocation to the caches of cluster MHs provided that a larger volume of popular video files can fit in the existing MHs' caches. Besides, this is the reason why for a lower mean file size $\mu = 4$, the random strategy exhibits a fast increase to its performance as compared to higher mean file sizes (e.g. $\mu = 8, 12$). Interestingly, the proposed strategy for $\mu = 12GB$ is shown to attain a similar CHP with the greedy strategy for $\mu = 8GB$, highlighting that the proposed strategy enables an enhanced utilization of the cache capabilities available by the MHs constituting the MEC cluster even when larger files are considered.

Let us now investigate the impact of a different mixture of cache sizes across the SBS and the FBS MHs on the cluster CHP performance, assuming that the mean cache size of an SBS MH is proportional to the mean cache size of FBS MHs with ratio β, i.e. $\xi_1 = \beta\xi_2$. Fig. 4 plots the impact of the FBS/SBS cache size ratio β on the cluster-wide CHP under different mean cache size values for the FBS tier $\xi_2 = 4, 8, 20$ GBs. For $\beta = 0$ the CHP performance is the result of utilizing only the N_2 FBS MHs (i.e. no SBS MHs since $\xi_1 = 0$GBs). As expected, the performance of all

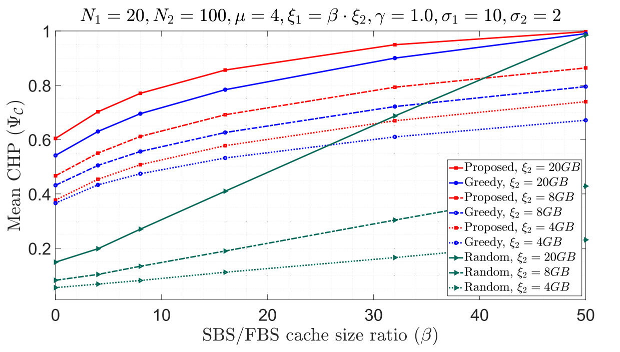


FIGURE 4. CHP performance for a different mixture of SBS and FBS MHs' cache size in the MEC cluster.

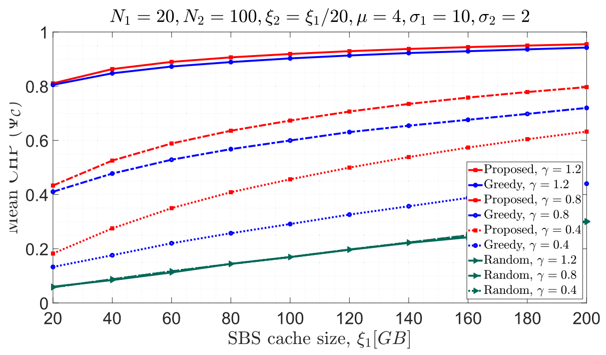


FIGURE 5. CHP performance for increasing SBS cache size and Zipf popularity skewness γ.

content placement strategies increases with ξ_2 and β. In fact, for $\xi_2 = 50$GBs and $\beta = 50$, the available cache size in the MEC cluster is sufficient to store the entire set of video files in $\mathcal{M}$, enabling all strategies to attain a CHP close to 1.

Similarly to Fig. 2, the CHP of the random strategy increases linearly with the available cache size (modeled by the ratio β in Fig. 4). However, the ratio of the CHP improvement strongly depends on the mean cache size ξ_2 assumed for the FBS MHs. For $\beta = 0$ (i.e. no SBS MHs), we observe that the random strategy performs poorly independent of the mean file size μ, whereas the proposed and greedy strategies exhibit similar performance as they are able to exploit the available cache size fitting popular files with higher popularity (e.g. for $\xi_2 = 4$GBs, they fit a single popular file on the average given that $\mu = 4$). As the β parameter increases and the mean FBS cache size is higher, e.g. $\beta > 40$ and $\xi_2 = 20$GBs, the CHP performance of the random strategy is shown to fast close the CHP gap with the proposed and greedy strategies, indicating that random content placement can provide comparable benefits with more sophisticated strategies if the MHs have a large volume of storage resources available for content caching. This observation can be useful to MEC clusters with high storage capacity but low processing capabilities, enabling the cluster head to deploy random content placement with good CHP performance.

Nonetheless, random content placement when the MEC cluster consists of SBS MHs with low storage capacity (e.g. $\beta < 20$) performs poorly. On the contrary, both the proposed and the greedy strategies exhibit high CHP performance gains even when the available cache size at the FBS MHs is low (e.g. $\xi_2 = 4$GBs) and when the cache size of SBS MHs is high (e.g. $\beta > 20$). In both cases, the proposed content placement strategy outperforms the greedy strategy enabling the MEC cluster to attain 3-5% better CHP performance, especially when the total cluster size cannot support caching of a large volume of popular video files in $\mathcal{M}$ (i.e. CHP is lower than 1).

In Fig. 5 we investigate the interplay between increasing the mean SBS cache size under different Zipf popularity values γ. Recall that a lower γ parameter corresponds to 'spreading' the popularity to a larger number of video files in $\mathcal{M}$, whereas a higher γ parameter corresponds to 'concentrating' the popularity to a smaller set of video files. Interestingly, the performance of the random strategy is shown to remain unaffected by the Zipf popularity parameter γ. This follows from the fact that, on the average, random content placement will result in caching video files with popularity summing up to the same value. Fig. 5 also illustrates that a lower Zipf popularity parameter γ (e.g. by comparing $\gamma = 1.2$ to $\gamma = 0.4$) the performance of all strategies deteriorates.

Another interesting observation is that the performance gap between the proposed and the greedy content placement strategies fast increases as the value of γ is decreased (e.g. compare $\gamma = 0.8$ and $\gamma = 0.4$). This readily follows from the fact that a lower popularity skewness γ spreads the popularity to more files leading to a large number of files with comparable popularity but different size. Provided that the greedy strategy is size-agnostic and assigns popular video files to the cache of MHs taking into consideration only the popularity of the files in $\mathcal{M}$, it readily follows that popularity-based greedy content placement is unable to infer on an appropriate combination of video files to be allocated to the available cache of MHs. This is a result of the different file sizes and MH caches considered in our simulation (i.e. the respective parameters are not assumed to be fixed and equal for all files and MHs, respectively).

On the contrary, the proposed strategy is shown to better adapt to the heterogeneity of both the file sizes met in $\mathcal{M}$ and the cache size available to the MHs, enabling an enhanced CHP performance as compared to the greedy strategy. In particular, when the popularity is spread to a larger volume of video files (e.g. for a low popularity value $\gamma = 0.4$) and the cache size available in the different types of MHs forming the MEC cluster is more diverse (e.g. for $\xi_1 = 100$GBs, SBSs have 95GBs more cache available as compared to FBSs), the performance gap between the proposed and the greedy strategies largely increases. For $\xi_1 = 100$GBs. we observe that the respective performance gap reaches up to a 20% CHP improvement in an absolute scale and 40% improvement in a relative scale.

As a takeaway result, we conclude that the proposed bound-and-bound MKP 0/1 strategy can better exploit the available cache size of MEC clusters with higher heterogeneity, in terms of available cache size per MH, enabling the MEC cluster to attain a higher CHP performance. This performance improvement is even more evident when the popularity of video files in $\mathcal{M}$ is spread to a larger volume of video files (e.g. when the Zipf parameter γ is lower). Taking into consideration that the performance of the proposed and greedy strategies is comparable for a higher values of the popularity parameter γ, e.g. $\gamma > 1.2$, another interesting observation is that in such scenarios, the MEC cluster head may choose to run the greedy strategy instead of the proposed one to save processing resources.

B. IMPACT OF THE FILE POPULARITY

In Fig. 6 we assess the impact of the Zipf popularity parameter γ on the CHP performance under different mean file sizes μ. Note that for $\gamma = 0.2$ the popularity of contents is close to uniform (1% of files account for 2.8% popularity), whereas for $\gamma = 2$ a very small amount of video files concentrates a very high popularity (1% content account for 98.8% of popularity). Similar to Fig. 5 we observe that the popularity parameter γ has a small impact on the performance of the random content placement strategy. In more detail, we observe a small CHP deterioration of the random strategy for a small mean file size $\mu = 3$GBs due to the fact that when a very small number of video files in $\mathcal{M}$ concentrates a high popularity (i.e. high γ values) the random strategy select with a higher probability a combination of files with lower popularity (i.e. more allocation options result in low CHP performance).

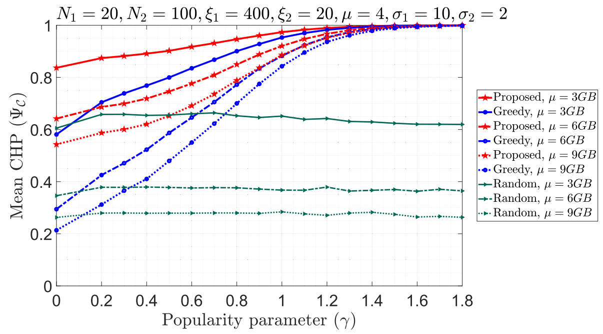


FIGURE 6. CHP performance for increasing popularity skewness γ and different mean file sizes μ.

On the contrary, the performance of both the proposed and the greedy strategies is shown to fast increase with the popularity parameter γ and even reach to $\Psi_C = 1$ for $\gamma > 1.4$ under the fixed system parameters under scope. This performance trend readily follows from the fact that a higher γ parameter translates to the concentration of the popularity in $\mathcal{M}$ to lower volume of video files, enabling popularity-aware strategies to fill the cache sizes available in the MEC cluster with highly popular files while leaving less popular files out of the content placement process if necessary. Another interesting observation is that for the proposed and greedy strategies, the impact of the popularity parameter γ on the CHP performance strongly depends on the mean cache size μ for low popularity values γ (e.g. $\gamma < 0.8$).

The proposed strategy is shown to outperform the greedy strategy in terms of CHP probability, especially when the file size of popular videos is high. For example, for $\mu =$ 9GBs, the proposed strategy is shown to attain a double-fold increase on the CHP when the Zipf parameter is very low (e.g. $\gamma = 0.2$). Once again, the proposed strategy is shown to better handle the cache available in the MEC cluster even for larger video files, if we consider that the CHP of the proposed strategy for $\mu = 9$GBs is higher compared to the CHP performance of the greedy strategy for $\mu = 6$GBs, for all popularity parameter values under scope.

As expected, a lower mean file size μ enables all strategies to better handle the storage capacity available in the MEC cluster and increase the CHP performance. This readily follows by comparing the performance of any strategy for $\mu = 3$GBs to the respective performance of the same strategy for $\mu = 6$GBs. Nonetheless, a constant increase of the mean file size μ is shown to have different effects on the CHP performance of all strategies, independent of the γ value. For example, the CHP performance of the random strategy for $\mu = 3$GBs is shown to increase almost three-fold as compared to the one for $\mu = 9$GBs, whereas the CHP of the same strategy for $\mu = 3$GBs does not increase two-fold as compared to the one for $\mu = 6$GBs. Another interesting observation is that for low popularity values of γ, where the popularity distribution tends to be more uniform, the proposed strategy attains the highest CHP gains as compared to the greedy strategy. On the other hand, for high popularity parameters γ, the CHP performance of the greedy and the proposed strategies is comparable. This performance trend can be useful to scenarios where the MEC cluster head identifies that the popularity of files concentrates to small number of video files, enabling it to save processing resources by employing the greedy strategy instead of the proposed (exact) one.

In Fig. 7 we investigate the impact of the popularity skewness γ under a different number of SBS and FBS MHs. Note that as the number of SBS and FBS MHs increases, the available cluster cache size is considered to increase proportionally, i.e. the total pool of available cache resources in the cluster increases proportionally. Similar to Fig. 6, the CHP performance of the random strategy is shown to deteriorate with γ, independent of the number of MHs in the MEC cluster. Another interesting observation is that for all strategies, a two-fold increase on the number of high-end MHs (i.e. MHs with large cache capacity like SBSs in our simulations) is shown to provide significantly larger CHP gains as compared to the ones attained due to a similar two-fold increase on the number of low-end MHs (i.e. MHs with low cache capacity like FBSs in our simulations). This readily follows from the fact that SBSs are considered to host a 20x higher cache capacity as compared to FBSs, enabling the content placement strategies to better handle popular files of larger file sizes.

The CHP performance gap between the proposed and the greedy strategies is also shown to fast decrease as the popularity skewness parameter γ increases. Fig. 7 that increasing the number of high-end MHs can play a key role when the popularity of video files in $\mathcal{M}$ is spread across more files, i.e. the impact of N_1 dominates that of γ for low γ values, but the addition of more MHs will have a lower impact on the CHP performance when the popularity of files in $\mathcal{M}$ is concentrated in a lower number of files (i.e. for higher γ values). This performance trend can be used to adjust the MEC cluster size dynamically in view of the popularity distribution characterizing the library of popular video files, enabling the formation of additional MEC clusters with a lower number of MHs in order to better adapt the number (and structure) of MEC clusters to the spatial diversity of content popularity observed in a HCN.

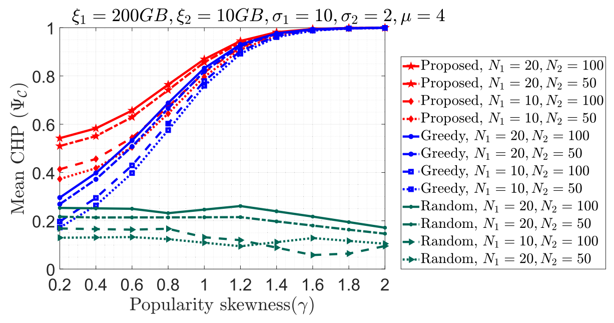


FIGURE 7. **CHP performance for increasing popularity skewness γ and different number of SBSs N_1 and FBSs N_2.**

Accordingly, we derive a valuable cluster formation design guideline. If for a given geographical area a lower amount of files concentrates a higher popularity (i.e. higher γ values) but the content popularity distribution changes fast across neighbor geographical areas, the MNO can form a larger number of MEC clusters constituting by a lower volume of MHs to better adapt to the diverse popularity met across neighbor areas. On the other hand, if the content popularity is known to be similar across neighbor geographical areas, the MNO may choose to form a smaller number of MEC clusters with an increased volume of MHs to attain an enhanced network-wide CHP.

Another interesting observation is that the proposed content placement strategy can better adapt to the existence of a limited number of MHs in the MEC cluster while attaining a similar performance with that of the greedy strategy. In particular, we observe that the proposed strategy for $N1 = 10$ attains a comparable performance with that of the greedy strategy for $N_2 = 20$ (double-fold increase of high-end MHs) even when the popularity parameter γ is medium-to-high (>0.6).

C. IMPACT OF THE MEAN CONTENT SIZE

In Fig. 8 we plot the cluster-wide CHP for increasing mean content size μ in $\mathcal{M}$ under different popularity parameters γ.

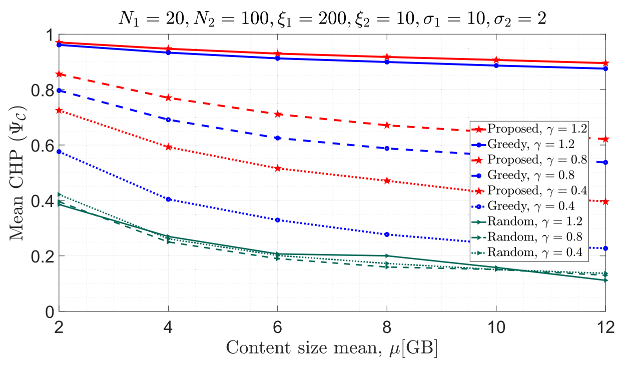


FIGURE 8. **CHP performance for increasing content size μ and different popularity skewness γ.**

As expected, when a larger content size μ is considered for popular video files in $\mathcal{M}$, the CHP performance of all strategies deteriorates. This mainly follows from the fact that the available cache size on a per MH (and a cluster-wide) basis remains fixed while the size of the cached content increases. We observe a roughly linear performance deterioration for all strategies above the mean content size $\mu = 4$GBs value. The performance deterioration is shown to strongly depend on the popularity distribution parameter γ.

For the different γ parameters plotted in Fig. 8, we observe that the performance of the random strategy remains roughly unaffected by γ, while a constant increase of the γ parameter (e.g. from 0.4 to 0.8, or 0.8 to 1.2) is shown to increase the CHP performance of both the proposed and the greedy strategies with a constant rate (e.g. for $\mu = 4$ the CHP of the proposed increases by roughly 20% when γ increases by 0.4). Another interesting observation is that the performance gap between the proposed and greedy strategies increases with the mean content size of popular video files μ, highlighting that the proposed strategy can better cope with larger contents in $\mathcal{M}$ as compared to the greedy strategy.

Fig. 9 investigates the CHP performance for increasing mean content size μ under a different number of SBS and FBS MHs on the CHP. Recall that an increase to the number

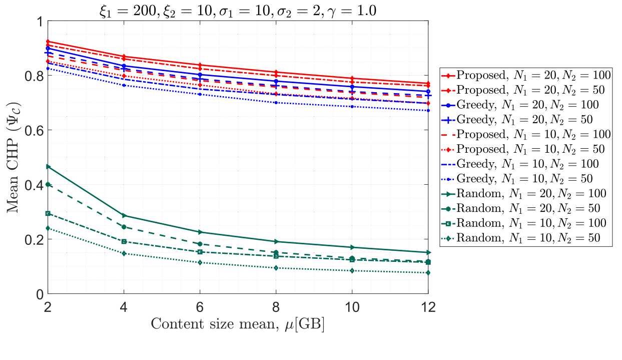


FIGURE 9. **CHP performance for increasing content size μ and different number of SBSs N_1 and FBSs N_2.**

of SBS, or FBS, MHs results in a proportional increase of the total cache size available in the cluster. For the given selection of the Zipf law parameter $\gamma = 1.0$, we observe that the impact of the number N_2 of FBS MHs is negligible on the CHP performance of both the proposed and the greedy strategy when a large number of high-end SBS MHs is considered (i.e. $N_1 = 20$). However, when the number of high-end MHs is lower ($N_1 = 10$), the respective CHP improvement observed by increasing two-fold the number of low-end MHs is larger for both the proposed and the greedy strategies. On the other hand, for the random strategy, we observe that the CHP performance shows notable improvement with a two-fold increase of either the SBS, or the FBS helpers, an improvement that is shown to be similar between the two scenarios (i.e. two-fold increase of N_1 results in similar improvements with a two-fold increase of N_2).

Interestingly, for all content placement strategies, the impact of increasing N_1 and N_2 is roughly the same on the CHP independent on the values taken by the μ parameter (e.g. the same improvement is observed while changing $N_1 = 10$ to $N_1 = 20$ independent of the values taken by μ and N_2). This performance trend indicates that increasing the number of MHs in the MEC cluster will result in the same CHP improvement, independent of the mean content size μ characterizing the files in library $\mathcal{M}$.

D. IMPACT OF THE NUMBER AND TYPE OF CACHE-ENABLED MHs

In Fig. 10 we assess the impact of an increasing number of high-end SBS MHs under different values on the number of FBS MHs. As expected, the performance of the random strategy is shown to scale linearly with the number of SBS MHs N_1 in the MEC cluster. On the contrary, a fast improvement of the CHP performance is observed for both the proposed and the greedy strategies in lower N_1 values as the number of SBS MHs increases, especially when the number of FBS MHs is lower (e.g. compare $N_2 = 10$ and $N_2 = 100$ for an increase of N_1 from 1 to 4). This performance trend highlights that the addition of only a few high-end MHs (like the SBS in our simulations) has a high added value on the CHP performance, especially when the size of the MEC cluster is low.

On the other hand, a similar performance trend is observed for increasing the number of FBS MHs N_2, when a high number of SBS MHs already exists in the MEC cluster. In more detail, as the N_1 value increases, the proposed and greedy strategies are shown to improve the CHP at a lower rate for the same increase of N_2 (e.g. for $N_1 = 2$ SBS MHs we observe a higher CHP improvement when N_2 changes from 10 to 40 as compared to that obtained for $N_1 = 16$ for the same N_2 increase). This performance trend can be valuable to the design of the cluster formation algorithm run by the MNO, as it highlights that the CHP improvement following from the addition of more MHs to a tagged MEC cluster should be examined in view of the current mixture of MHs in the cluster in order to maximize the added value following from the formation of a larger cluster.

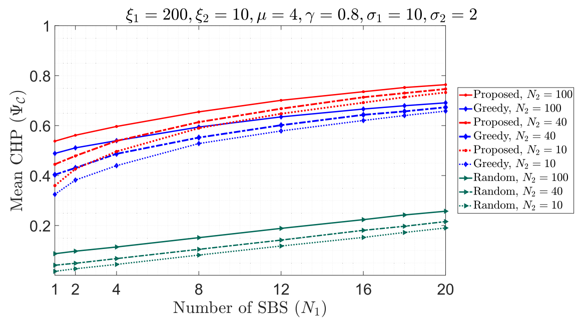


FIGURE 10. CHP performance for increasing number of SBS MHs N_1 and different number of FBS MHs N_2 without fixing the total cache sizes L_{SBS} and L_{FBS}.

The underlying performance trade-off is that, on the one hand, the addition of more MHs can improve the CHP of a given MEC cluster but at the same time, the same number of MHs can be used to set up a new MEC cluster having a different library $\mathcal{M}$ that matches better the user requests on popular videos (which can be different in size and popularity for neighbor geographical areas). To this end, the MEC clustering algorithm should evaluate the addition of a new MH in the MEC cluster not only in light of the content library assumed per geographical area (and how good this adapts to the actual MEC service coverage) but also in view of the selected mixture of MHs allocated per MEC cluster.

E. IMPACT OF THE NUMBER OF POPULAR VIDEO CONTENTS IN $\mathcal{M}$

In Fig. 11, we assess the impact of an increasing number of popular video contents in the library $\mathcal{M}$ under different popularity values γ. The CHP performance of the random strategy is shown to drop rapidly as the number of popular video contents increases, in a roughly independent fashion from the values taken by the Zipf parameter γ. Another interesting observation is that when the popularity of files in $\mathcal{N}$ is concentrated to a lower amount of video files in $\mathcal{M}$, the performance of the greedy strategy is very close to that of the proposed (optimal) one. This performance trend can be exploited to save computations at the MEC cluster head upon

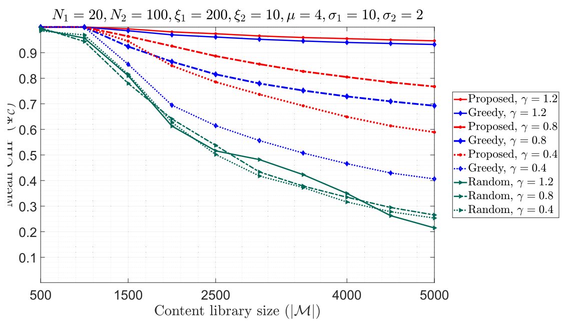


FIGURE 11. CHP performance for increasing number of contents in the library $\mathcal{M}$ and different Zipf parameters γ.

content placement of popular video files that are characterized by large γ values (e.g. $\gamma > 1.2$).

Nonetheless, the CHP performance gap between the proposed and the greedy strategies is shown to rapidly increase with the library size $|\mathcal{M}|$ when the popularity is spread across more video files in $\mathcal{M}$ (i.e. $\gamma < 0.8$). This performance trend highlights the importance of employing more sophisticated planning of the cluster-based content placement when the popularity of video files in $\mathcal{M}$ is characterized by higher γ values. As compared to the greedy and random strategies, the proposed content placement strategy is shown to be highly-robust against an increase of the number of files in $\mathcal{M}$ as well as to a lower Zipf parameter γ.

F. PROCESSING OVERHEADS AND COMPUTATION TIME

In this section, we evaluate the performance of the three content placement strategies in terms of computation time necessary to complete the content allocation to the caches of MHs constituting the MEC cluster. Recall that this operation shall be deployed on an epoch-by-epoch basis and the allocated contents shall be replaced by new ones by the beginning of the next cache epoch. Thus, each epoch is expected to last for at least a few hours (or days). For our evaluations we have used a standard desktop PC equipped with i) Intel(R) Core(TM) i7-8550U CPU @1.8GHz, ii) DDR3 RAM of 8GBs, and iii) a 64-bit Windows 10 operating system. All strategies were implemented using a 64-bit version of Matlab R2018b and the respective computation time measurements were averaged over multiple samples (>100 per tick). Most of the computation time results have derived in line with the system parameter values used to derive the CHP measurement results of the previous subsections.

The main reason that led us measure that computation time necessary for completing the content placement allocation by each strategy is the fact that, in a general system parameter setup, the proposed strategy is exact and solves a NP-hard problem by exploiting the bound-and-bound search algorithm, which eliminates non-optimal solution branches a-priori (i.e. without exploring the entire state space of branches leading to sub-optimal allocations). As a general outcome of the simulation campaigns, we have observed that for the given set of system parameter values, which are in full accordance with practical content placement scenarios in medium-sized MEC clusters of HCN nodes, the proposed strategy requires only a few seconds to complete in a standard desktop PC. This result suggests that the proposed bound-and-bound exact content placement strategy is both optimal and feasible in terms of processing overhead under realistic MEC deployment scenarios (i.e. when the number of MEC nodes is in the order of hundreds of cache-enabled HCN nodes).

Interestingly, the proposed exact content placement strategy typically required more computation time in parameter setups where the performance of the proposed strategy is very close to that of the greedy strategy, highlighting that such marginal scenarios can be identified a-priori by the MEC cluster head, which can switch from the proposed to the greedy content placement strategy. Nonetheless, we note that in all scenarios under scope, the computation time necessary for deploying the proposed optimal content placement strategy is in the order of a few seconds, enabling its practical deployment in MEC clusters consisting of a few hundreds of MEC-enabled nodes and a library size of thousands of popular video files.

1) IMPACT OF LIBRARY SIZE AND NUMBER OF MEC MHs

In Fig. 12 we plot the computation time necessary for completing the content placement allocation for an increasing number of video files in library $\mathcal{M}$ and different γ values. Note that the parameter values considered in this scenario are fully aligned with the ones used to derive the CHP performance of all strategies in Fig. 11. First of all we observe that the computation time necessary to complete for all content placement strategies is in the range of sub-seconds under all system parameter values under scope. We further observe that the computation time increases proportionally to the content library size $|\mathcal{M}|$ for all content placement strategies under scope.

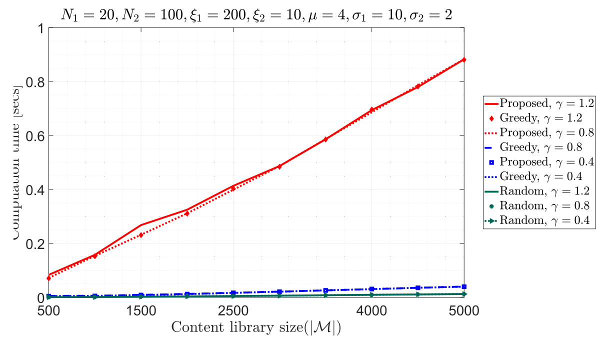


FIGURE 12. **Computation time for increasing number of contents in library $\mathcal{M}$ and different γ parameters.**

According to Fig. 12, the proposed content placement strategy is computationally feasible (sub-second computation time) in MEC scenarios where the number of MEC-enabled MHs is in the order of hundreds nodes, their cache size is in the order of a few GBs, the mean size of popular videos is in the order of a few GBs and the total number of popular video contents selected for the library $\mathcal{M}$ is in the order of a few thousands of files. As expected, the computation time necessary for the random strategy increases linearly with the size of the content library $\mathcal{M}$, whereas the computation time for the greedy strategy is shown to scale with $\mathcal{M}$ in a roughly sub-squared fashion, due to the sorting algorithm employed to order video files based on their popularity on a per MH basis.

In Fig. 13 we plot the computation time required for the completion of all strategies assuming an increasing number of SBS and FBS MH values. Once again, the time necessary to complete the content placement logic for all strategies is in the range of sub-seconds. The performance of the greedy and the random strategies is shown to remain roughly unaffected

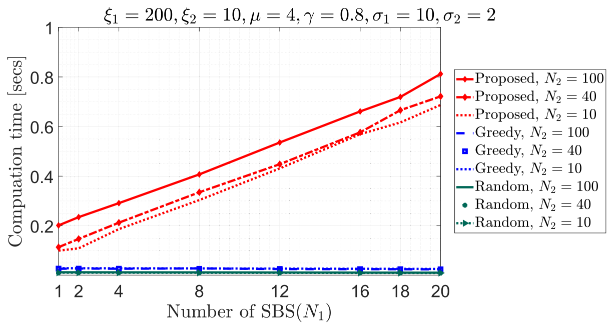


FIGURE 13. **Computation time for increasing number of SBS MHs N_1 and different number of FBS MHs N_2.**

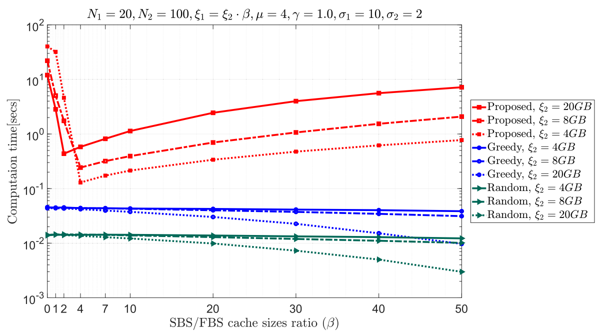


FIGURE 14. **Computation time for increasing number of the β SBS/FBS cache size ratio and different FBS cache size ξ_2 values.**

by the number of MHs in the MEC cluster, including both SBS (N_1) and FBS (N_2) MHs. On the one hand, this is expected for the greedy strategy where the main processing overhead comes from the sorting popular video files $f_m \in \mathcal{M}$ based on their popularity ρ_m and scanning the order list until the cache size of each MH is filled. On the other hand, this is also expected for the random strategy where the files are randomly allocated in a MH-by-MH basis and the entire list of non-cached video files is to be scanned in order to concluded that no other file can fit into the cache of a given MH.

The computation time necessary to complete the proposed strategy is shown to increase linearly with the number of high-end SBS MHs, under all N_2 parameters under scope. The inclusion of additional low-end FBS MHs is also shown to add a roughly fixed computation overhead to the processing requirements of the proposed strategy, independent of the current value of the number of high-end SBS MHs. For example, this can be observed by comparing the computation time of the proposed strategy for $N_2 = 100$ and $N_2 = 10$ under different N_1 values. Notably, the computation time needed to complete the proposed strategy is shown to remain in the area of sub-seconds when the number of high-end SBS MHs reaches $N_1 = 20$ and the number of low-end FBS MHs is set to $N_2 = 100$, indicating that the proposed bound-and-bound content placement strategy can be readily deployed in medium-to-large sized MEC clusters (i.e. >120 MHs).

2) IMPACT OF SBS MHs' CACHE SIZE, ZIPF POPULARITY PARAMETER AND MEAN FILE SIZE

In Fig. 14 we plot the impact of the SBS/FBS cache size ratio β on the computation time of all content placement strategies, under different SBS cache size ξ_2 values. As in Fig. 4, we use parameter $\xi_1 = \beta \cdot \xi_2$ to increase the SBS MH mean cache size ξ_1 in proportion to the FBS MH mean cache size ξ_2. The value of ξ_2 is assumed to remain fixed. For $\beta = 0$ the MEC cluster consists of only FBS MHs, i.e. $\xi_1 = 0 \cdot \xi_2$. We start our discussion with the performance of the greedy and random content placement strategies. As shown in Fig. 14, the computation time necessary for completing the content placement under both the greedy and the random strategies is in the sub-second range. As expected, the random strategy requires less computation time compared to the greedy one, which is required to additionally order the (remaining) files based on their popularity.

When the available cache size in FBS MHs is $\xi_2 = 20$GBs, the computation time required for both the greedy and the random strategies is shown to drop with an increase of β. The aforementioned trend follows from the fact that both strategies run the content placement on a per MH basis as follows: i) pick a MH to fill its cache, ii) scan the list of popular video contents that have not been cached by previous MHs, and iii) sequentially select (based on popularity order, or randomly) a list of files to cache until the available cache of the MH is full. Files which cannot fit into the cache of a MH are skipped and the evaluation continues with the next ones in the list of non-cached files, until no other file can fit into the cache of the MH. Thus, in each iteration, the full list of **remaining** files will be scanned per MH. As the mean cache size of MHs increases, the number of non-cached files remaining for content placement decreases fast, reducing the overall computation time (Fig. 14). As a conclusion, the computation time required for the greedy and random strategies drops rapidly when the available cache of MHs in the MEC cluster is high.

Let us now discuss the performance of the proposed content placement strategy. As expected, the proposed strategy is shown to require more computation time compared to the random and greedy content placement strategies; however, its requirements are still in the scale of a few seconds. Hence, the proposed content placement strategy is not only exact but is also computationally feasible in scenarios where hundreds of MEC-enabled MHs perform coordinated content placement. In Fig. 14 we observe that under all ξ_2 values under scope, the computation time of the proposed strategy at its highest point for $b = 0$, drops to a minimum as b increases from 0 to low b values and then it increases again slowly with b. For example, when $xi_2 = 20$ GBs, the proposed strategy requires the highest computation time (11 seconds) for $b = 0$, reaches to the minimum requirement of 0.2 seconds for $b = 1$ and increases again slowly above $b > 1$. A similar performance trend is observed for other values of xi_2 (e.g. 4 and 8 GBs).

Recall that the proposed content placement strategy is based on a dynamic exploration of the full state space and uses upper/lower performance bounds to eliminate sub-optimal solution branches, reducing unnecessary computations whenever possible. Accordingly, the proposed strategy requires more computation time when the bound-and-bound exploration methodology should go deeper into solution branches in order to assess their CHP performance. When the state space consists of solutions with diverse CHP performance values, e.g. due to the diverse cache sizes of MHs, bound-and-bound exploration enables fast elimination of sub-optimal solution branches avoiding in-depth examination. However, when the state space consists of solutions with similar CHP performance, e.g. due to similar file and cache size, bound-and-bound exploration should go deeper into the examination of solution branches before eliminating them, increasing the computation time.

When b is at the low end, cache diversity in the MEC cluster is weak and the cache sizes of MHs are roughly the same. Under this scenario, most of the solution branches involve the placement of files into similarly-sized MHs, enforcing the bound-and-bound exploration strategy to evaluate an increased volume of placement combinations with similar CHP performance. For that reason, in Fig. 14 we observe a different behavior of the proposed strategy when b is low (close to 1), as compared to the one for b taking medium-to-high values (e.g. $b > 4$). For example, when $b = 0$ we record the highest computation time for all ξ_2 values under scope (i.e. the MEC cluster includes only FBS MHs of similar size). However, as b increases, the cache diversity is more evident and the bound-and-bound strategy quickly reduces its computation time. Above a certain b value, which depends on the value of ξ_2, cache diversity is well established and the performance of the proposed strategy scales with an increase of b, or ξ_2. Once again, this performance trend follows from the fact that a larger value of b, or ξ_2, enables the MEC cluster to cache more files, enforcing the bound-and-bound exploration strategy to investigate feasible solution branches in more depth and to increase its computation time.

Interestingly, when b is at the low end (i.e. SBS and FBS cache sizes are similar), a larger mean cache size ξ_2 reduces the computation requirements of the proposed strategy. This performance trend can be explained as follows. When a low cache diversity characterizes the MEC cluster (low b values) and the cache size available per MH is close to the mean file size $\mu = 4$GBs, the state space is dominated by a vast volume of solutions with similar CHP performance that involve the placement of only a few files into the caches of similarly-sized MHs. As a result, bound-and-bound exploration requires more computation time to eliminate sub-optimal solutions. This effect is alleviated when the available cache size ξ_2 is larger, enabling an enhanced file placement diversity per MH to be achieved.

In Fig. 15 we examine the impact of the popularity distribution parameter γ and the mean file size μ on the computation time of all strategies, as the mean SBS cache size

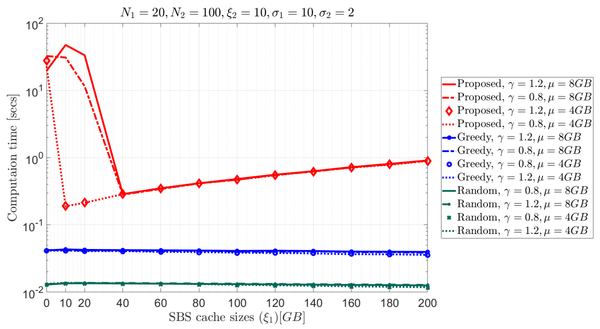


FIGURE 15. Computation time for increasing SBS mean cache size ξ_1 under different Zipf popularity parameter γ and mean (popular) file size μ values.

ξ_1 increases. The performance of the greedy and random content placement strategies remains roughly unaffected by the γ and μ values under scope, with a slight decrease of the required computation time observed when the mean SBS cache size is on the high-end of Fig. 15 and the mean file size is low (i.e. $\mu = 4$GBs). The main reason of the reduced computation time necessary for both the greedy and random strategies is that a lower mean file size μ results in a higher content placement rate in the caches of MHs in the early steps of the strategies. To this end, as the caches of MHs are filled sequentially by both strategies, a smaller volume of non-cached files will be available for subsequent MHs thus, reducing the scanning time necessary for filling the cache of MHs examined at the final steps of the greedy and random strategies. A similar behavior has been discussed for both strategies in Fig. 14.

Let us now assess the performance of the proposed content placement strategy. As expected, the proposed strategy requires more computation time as compared to the baseline strategies. Nonetheless, the computation time of the proposed strategy still remains within feasible limits, which are of the order of a few seconds for all γ and μ parameters under scope. Notably, above a certain mean SBS cache size value ξ_1, which is close to ξ_2, we observe that the computation time remains roughly unaffected by the Zipf popularity parameter γ and the mean file size μ, e.g., for $\xi_1 > 60$ GBs. Similar to what we have observed in Fig. 14, this result follows from the fact that above a certain ξ_1 value, the content placement state space includes solution branches exhibiting a diverse CHP performance. This enables the proposed strategy to eliminate sub-optimal branches without going deeper into their assessment. Accordingly, as shown in Fig. 15, the impact of μ and γ is shown to be negligible, enabling the proposed strategy to scale its computation time requirements only with the mean cache size ξ_2 available at high-end SBS MHs. This follows from the fact that when the SBS cache size is very large, high-end MHs can fit an adequate number of popular video files, i.e. the large cache size dominates the impact of the file size and popularity on the computation time requirements.

On the other hand, similar to our findings from Fig. 14, when the mean SBS cache size value ξ_1 is comparable with the FBS mean cache size value of ξ_2 (which is set to 10 GBs in this setup), an additional computation overhead is observed for the proposed strategy (Fig. 15). Under this scenario, we observe that a larger mean file size μ increases the computation time of the proposed strategy, a performance that can be validated by comparing the plots of the proposed strategy for $\mu = 4$ and $\mu = 8$ GBs. This can be explained as follows: a larger mean file size μ lowers the 'effective' mean cache size of all MHs, reducing the average number of popular video files that are cached into the buffer of a given MH type. Besides, on the average, an SBS MH will cache ξ_1/μ popular video files and a FBS MH will cache ξ_2/μ files. This effect diminishes when the mean SBS cache size ξ_1 is high and the solution branches are highly diverse, enabling the proposed strategy to lower its computation time requirements (for $\xi_1 > 60$ GBs in Fig. 15).

When ξ_1 is at the low end, we further observe that the impact of the Zipf popularity parameter γ varies depending on the mean file size μ. As shown in Fig. 15, when μ is low, the Zipf parameter value has a negligible impact on the computation time performance, e.g. compare the plots of the proposed strategy for $\mu = 4$ GBs $\gamma = 1.2$ and $\mu = 4$ GBs $\gamma = 0.8$. However, when the mean file size μ is larger ($mu = 8$ GBs) and ξ_1 is at the low end, a larger Zipf parameter γ (less files concentrate the highest popularity) is shown to increase the computation time requirements of the proposed strategy. This performance trend follows from the fact that a higher mean file size μ reduces the average number of files cached per MH (as explained above) and that for $\gamma = 1.2$ less files concentrate the highest popularity, enforcing the proposed strategy to explore solution branches with similar CHP performance in more depth, i.e. branches that involve the placement of video files with low popularity.

Taking into consideration the results in Figs. 2 and 5, we conclude that a higher computation time is necessary for the proposed bound-and-bound exact content placement strategy mainly under marginal system parameter setups where the CHP performance of the proposed strategy is very close to that of the greedy strategy. The processing complexity of such marginal scenarios can be mitigated by the content placement decision making entity by switching to the greedy strategy, or employing relaxed upper performance bounds under the proposed strategy. It should also be noted that a very low computation time is required in scenarios where the proposed strategy exhibits superior performance as compared to the greedy and random strategies.

VI. CONCLUSION

In this paper, we have proposed a novel content placement architecture where cache-enabled mobile helpers (MHs) can be grouped into MEC-enabled clusters to perform cooperative content placement in view of a library of popular video files that remains fixed in size and popularity distribution. Using a ZOMKP formulation with respect to constraints on caching entire video files and not allowing an overlap of cached files within the same MEC cluster, we have presented an exact (optimal) content placement strategy that employs bound-and-bound enumerate search of the content placement state space, discarding solution branches that are identified to lead in sub-optimal performance.

Using extensive system-level simulations we have shown that the employment of the proposed bound-and-bound exact strategy is both efficient and computationally feasible in the context of cluster-based content placement in MEC-enabled heterogeneous cellular networks. Valuable design guidelines and best practices have been derived through the system-level simulation analysis, highlighting key performance trade-offs for MEC cluster formation and optimal content placement in MEC-enabled HCNs. Future work includes the design of MEC-enabled content delivery and cluster formation strategies in view of the results derived in this paper.

REFERENCES

[1] J. Yao, T. Han, and N. Ansari, "On mobile edge caching," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 3, pp. 2525–2553, 3rd Quart., 2019.
[2] (2020). *Ericsson Mobility Q4-2019 Report, Ericsson, Stockholm, Sweden*. Accessed: Sep. 29, 2020. [Online]. Available: https://www.ericsson.com/en/mobility-report/reports/june-2020
[3] W. Jiang, G. Feng, and S. Qin, "Optimal cooperative content caching and delivery policy for heterogeneous cellular networks," *IEEE Trans. Mobile Comput.*, vol. 16, no. 5, pp. 1382–1393, May 2017.
[4] International Telecommunication Union (ITU). (2018). *Setting the Scene for 5G: Opportunities & Challenges*. Geneva, Switzerland. Accessed: Jun. 12, 2020. [Online]. Available https://www.itu.int/en/ITU-D/Documents/ITU_5G_REPORT-2018.pdf
[5] M. A. Salahuddin, J. Sahoo, R. Glitho, H. Elbiaze, and W. Ajib, "A survey on content placement algorithms for cloud-based content delivery networks," *IEEE Access*, vol. 6, pp. 91–114, 2018.
[6] H. S. Goian, O. Y. Al-Jarrah, S. Muhaidat, Y. Al-Hammadi, P. Yoo, and M. Dianati, "Popularity-based video caching techniques for cache-enabled networks: A survey," *IEEE Access*, vol. 7, pp. 27699–27719, 2019.
[7] M. A. Maddah-Ali and U. Niesen, "Fundamental limits of caching," *IEEE Trans. Inf. Theory*, vol. 60, no. 5, pp. 2856–2867, May 2014.
[8] T. X. Tran, A. Hajisami, and D. Pompili, "Cooperative hierarchical caching in 5G cloud radio access networks," *IEEE Netw.*, vol. 31, no. 4, pp. 35–41, Jul. 2017.
[9] L. P. Qian, Y. Wu, H. Zhou, and X. S. Shen, "Ultra-dense heterogeneous small cell deployment in 5G and beyond non-orthogonal multiple access vehicular small cell networks: Architecture and solution," *IEEE Netw.*, vol. 31, no. 4, pp. 15–21, Jul./Aug. 2017.
[10] D. Purkayastha, F. Jiangping, D. Frydman, G. Verin, K.-W. Wen, K. Kim, R. Arora, A. Odgers, L. M. Contreras, and S. Scarpina, "MEC in 5G networks," Sophia Antipolis CEDEX, Nice, France, ETSI White Paper 28, 2018.
[11] S. Safavat, N. N. Sapavath, and D. B. Rawat, "Recent advances in mobile edge computing and content caching," *Digit. Commun. Netw.*, vol. 6, no. 2, pp. 189–194, May 2020.
[12] Y. M. Saputra, D. T. Hoang, D. N. Nguyen, and E. Dutkiewicz, "A novel mobile edge network architecture with joint caching-delivering and horizontal cooperation," *IEEE Trans. Mobile Comput.*, vol. 20, no. 1, pp. 19–31, Jan. 2021.
[13] S. Zhang, W. Sun, and J. Liu, "Spatially cooperative caching and optimization for heterogeneous network," *IEEE Trans. Veh. Technol.*, vol. 68, no. 11, pp. 11260–11270, Nov. 2019.
[14] T. M. Ayenew, D. Xenakis, N. Passas, and L. Merakos, "A novel content placement strategy for heterogeneous cellular networks with small cells," *IEEE Netw. Lett.*, vol. 2, no. 1, pp. 10–13, Mar. 2020.
[15] 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 Commun. Surveys Tuts.*, vol. 20, no. 3, pp. 1710–1732, 3rd Quart., 2018.
[16] K. Poularakis, G. Iosifidis, A. Argyriou, I. Koutsopoulos, and L. Tassiulas, "Caching and operator cooperation policies for layered video content delivery," in *Proc. IEEE INFOCOM 35th Annu. IEEE Int. Conf. Comput. Commun.*, Apr. 2016, pp. 1–9.

[17] A. Sengupta, S. Amuru, R. Tandon, R. M. Buehrer, and T. C. Clancy, ''Learning distributed caching strategies in small cell networks,'' in *Proc. 11th Int. Symp. Wireless Commun. Syst. (ISWCS)*, Aug. 2014, pp. 917–921.
[18] R. Liu, H. Yin, X. Cai, G. Zhu, L. Yu, Q. Fan, and J. Xu, ''Cooperative caching scheme for content oriented networking,'' *IEEE Commun. Lett.*, vol. 17, no. 4, pp. 781–784, Apr. 2013.
[19] X. Li, X. Wang, K. Li, Z. Han, and V. C. M. Leung, ''Collaborative multi-tier caching in heterogeneous networks: Modeling, analysis, and design,'' *IEEE Trans. Wireless Commun.*, vol. 16, no. 10, pp. 6926–6939, Oct. 2017.
[20] C. Li, L. Toni, J. Zou, H. Xiong, and P. Frossard, ''QoE-driven mobile edge caching placement for adaptive video streaming,'' *IEEE Trans. Multimedia*, vol. 20, no. 4, pp. 965–984, Apr. 2018.
[21] J. Wen, K. Huang, S. Yang, and V. O. K. Li, ''Cache-enabled heterogeneous cellular networks: Optimal tier-level content placement,'' *IEEE Trans. Wireless Commun.*, vol. 16, no. 9, pp. 5939–5952, Sep. 2017.
[22] S. Zhang, P. He, K. Suto, P. Yang, L. Zhao, and X. Shen, ''Cooperative edge caching in user-centric clustered mobile networks,'' *IEEE Trans. Mobile Comput.*, vol. 17, no. 8, pp. 1791–1805, Aug. 2018.
[23] K. S. Reddy and N. Karamchandani, ''On the exact rate-memory trade-off for multi-access coded caching with uncoded placement,'' in *Proc. Int. Conf. Signal Process. Commun. (SPCOM)*, Jul. 2018, pp. 1–5.
[24] J. Kwak, Y. Kim, L. B. Le, and S. Chong, ''Hybrid content caching in 5G wireless networks: Cloud versus edge caching,'' *IEEE Trans. Wireless Commun.*, vol. 17, no. 5, pp. 3030–3045, May 2018.
[25] A. Khreishah and J. Chakareski, ''Collaborative caching for multicell-coordinated systems,'' in *Proc. IEEE Conf. Comput. Commun. Workshops (INFOCOM WKSHPS)*, Apr. 2015, pp. 257–262.
[26] Y. Nam, S. Song, and J.-M. Chung, ''Clustered NFV service chaining optimization in mobile edge clouds,'' *IEEE Commun. Lett.*, vol. 21, no. 2, pp. 350–353, Feb. 2017, doi: 10.1109/LCOMM.2016.2618788.
[27] C. S. M. Babou, D. Fall, S. Kashihara, Y. Taenaka, M. H. Bhuyan, I. Niang, and Y. Kadobayashi, ''Hierarchical load balancing and clustering technique for home edge computing,'' *IEEE Access*, vol. 8, pp. 127593–127607, 2020, doi: 10.1109/ACCESS.2020.3007944.
[28] W. You, C. Dong, X. Cheng, X. Zhu, Q. Wu, and G. Chen, ''Joint optimization of area coverage and mobile-edge computing with clustering for FANETs,'' *IEEE Internet Things J.*, vol. 8, no. 2, pp. 695–707, Jan. 2021, doi: 10.1109/JIOT.2020.3006891.
[29] M. Bouet and V. Conan, ''Mobile edge computing resources optimization: A geo-clustering approach,'' *IEEE Trans. Netw. Service Manage.*, vol. 15, no. 2, pp. 787–796, Jun. 2018, doi: 10.1109/TNSM.2018.2816263.
[30] N. Carlsson and D. Eager, ''Ephemeral content popularity at the edge and implications for on-demand caching,'' *IEEE Trans. Parallel Distrib. Syst.*, vol. 28, no. 6, pp. 1621–1634, Jun. 2017.
[31] S. Martello and P. Toth, ''A bound and bound algorithm for the zero-one multiple knapsack problem,'' *Discrete Appl. Math.*, vol. 3, no. 4, pp. 275–288, Nov. 1981.
[32] S. Martello and P. Toth, *Knapsack Problems, Algorithms and Computer Implementations*. Hoboken, NJ, USA: Wiley, 1990.
[33] D. T. Hoang, D. Niyato, D. N. Nguyen, E. Dutkiewicz, P. Wang, and Z. Han, ''A dynamic edge caching framework for mobile 5G networks,'' *IEEE Wireless Commun.*, vol. 25, no. 5, pp. 95–103, Oct. 2018.
[34] X. Zhong, X. Wang, L. Li, Y. Yang, Y. Qin, T. Yang, B. Zhang, and W. Zhang, ''CL-ADMM: A cooperative-learning-based optimization framework for resource management in MEC,'' *IEEE Internet Things J.*, vol. 8, no. 10, pp. 8191–8209, May 2021.
[35] A. Mehrabi, M. Siekkinen, and A. Yla-Jaaski, ''Energy-aware QoE and backhaul traffic optimization in green edge adaptive mobile video streaming,'' *IEEE Trans. Green Commun. Netw.*, vol. 3, no. 3, pp. 828–839, Sep. 2019.
[36] A.-T. Tran, N.-N. Dao, and S. Cho, ''Bitrate adaptation for video streaming services in edge caching systems,'' *IEEE Access*, vol. 8, pp. 135844–135852, 2020.
[37] X. Zhang, T. Lv, W. Ni, J. M. Cioffi, N. C. Beaulieu, and Y. J. Guo, ''Energy-efficient caching for scalable videos in heterogeneous networks,'' *IEEE J. Sel. Areas Commun.*, vol. 36, no. 8, pp. 1802–1815, Aug. 2018.
[38] B. N. Bharath, K. G. Nagananda, and H. V. Poor, ''A learning-based approach to caching in heterogenous small cell networks,'' *IEEE Trans. Commun.*, vol. 64, no. 4, pp. 1674–1686, Apr. 2016.
[39] YouTube. *Recommended Upload Encoding Settings*. Accessed: Sep. 20, 2020. [Online]. Available: https://support.google.com/youtube/answer/1722171?hl=en
[40] *Guidelines for Video Delivery Over a Mobile Network-Version 1.0*, NTT Docomo, Inc., Tokyo, Japan, Jul. 2014.

TADEGE MIHRETU AYENEW received the B.Sc. degree in electrical engineering from Arba Minch University, Ethiopia, in 2009, the M.Tech. degree in microelectronics engineering from Addis Ababa University, Ethiopia, and IIT-Delhi, in 2012, and the M.Sc. degree in telecommunication and multimedia engineering from the University of Brescia, Italy, in 2016. He is currently pursuing the Ph.D. degree. From 2009 to 2013, he worked as an Electrical Engineer and a Network Specialist at Ethio Telecom. He was an Intern Researcher at TNO, The Netherlands, in 2016. Since 2017, he has been an Early Stage Researcher (ESR) with the GAIN Group, Athens University, in the SPOTLIGHT Project, under the EU Marie. Skłodowska Curie Initiative. His research interests include the design and optimization of wireless networks, mainly the 5G and beyond, transmission protocols, network traffic modeling, multi-access edge computing (MEC), and signal processing.

DIONYSIS XENAKIS received the Ph.D. degree from the Department of Informatics and Telecommunications, National and Kapodistrian University of Athens, Greece, in 2014. He has participated in numerous EU-funded projects and coauthored more than 30 peer-reviewed journal and conference papers. His current research interests include the design and analysis of 5G and beyond mobile data network technologies with the emphasis given in multi-access edge computing and distributed ledger technologies. He has served as a TPC member and the chair for top-tier IEEE conferences. He was a recipient of the Special Grant and Support Program by the Onassis Foundation. He has been a reviewer to high-ranking IEEE journals in computer science and data networks.

NIKOS PASSAS received the Diploma degree (Hons.) from the Department of Computer Engineering, University of Patras, Greece, in 1992, and the Ph.D. degree from the Department of Informatics and Telecommunications, University of Athens, Greece, in 1997. He is currently a member of the Teaching Staff with the Department of Informatics and Telecommunications, University of Athens. He is a Group Leader of the Green, Adaptive and Intelligent Networking (GAIN) Research Group. Over the years, he has participated and coordinated a large number of national and European research projects. He has served as a Guest Editor and a Technical Program Committee Member in prestigious magazines and conferences, such as *IEEE Wireless Communications Magazine*, *Wireless Communications and Mobile Computing* journal, IEEE Vehicular Technology Conference, IEEE PIMRC, and IEEE GLOBECOM.

LAZAROS MERAKOS received the Diploma degree in electrical and mechanical engineering from the National Technical University of Athens, Greece, in 1978, and the M.S. and Ph.D. degrees in electrical engineering from the State University of New York, Buffalo, in 1981 and 1984, respectively. From 1983 to 1986, he was with the Faculty of the Electrical Engineering and Computer Science Department, University of Connecticut. From 1986 to 1994, he was with the Faculty of the Electrical and Computer Engineering Department, Northeastern University, Boston. From 1993 to 1994, he was the Director of the Communications and Digital Signal Processing Research Center, Northeastern University. Since 1994, he has been a Professor with the Department of Informatics and Telecommunications and the Director of the Communication Networks Laboratory, where he has led several EU funded projects that have shaped European Research and Development in the area of mobile networking. His research interests include network technologies, services, and applications. He has authored more than 280 articles in the above areas. He is currently the Chairman of the board of the Greek Universities Network (GUNet).