Dynamic Programming Based Content Placement Strategy for 5G and Beyond Cellular Networks

Tadege M. Ayenew, Dionysis Xenakis, Nikos Passas, Lazaros Merakos
Department of Informatics and Telecommunication
National and Kapodistrian University of Athens, Greece
{tadegem, nio, passas, merakos}@di.uoa.gr

***Abstract*—Optimal caching strategies of popular contents in heterogeneous cellular networks are studied. The increasing demand for data traffic by users of the wireless network can be handled by rapaciously caching most frequently accessed contents by users. Hence, we propose an efficient popular content placement strategy, the first step in the content caching process, typically for popular video files. To do so, we introduce a novel approach for caching popular contents. This caching strategy follows a dynamic programming approach to tackle the optimization complexity of selecting most popular files among a wide range of files, under certain constraints. The proposed strategy gives the combination of popular files to be cached that maximizes the optimal cache hit probability with a pseudo-polynomial time complexity. To that end, we used the well-known resourcing algorithm, called the 0/1-Knapsack problem, assuming that files are cached without partitioning.**

I. INTRODUCTION

In the heterogeneous cellular network (HCN), a small percentage of the services acquire higher popularity, at some time, and produces the largest portion of the traffic in the network. These traffic contents are vogue files which receive frequent requests from many users. Local caching of these contents, both at the radio access edge and the user equipments (UE), is very important to improve the quality of experience (QoE) in mobile networks. Content caching brings files closer to UE; thus it reduces the redundant requests of files to the backhaul network. Once the files are stored at the network edge closer to the user, i.e, the mobile helper (MH), the requests can be served by these MH. As a result, the backhaul congestion may be significantly reduced. In addition, the network response time may drastically reduce, even zero when the requested file is already cached at the UE itself. Fortunately, the advance in memory technology and the caching capability of current end-user devices renders the proposed caching strategy to be practical. The performance attained by the caching of popular contents is further enhanced by the deployment of advanced techniques such as device-to-device (D2D) communication, clustering, cooperation and coordination, transmission media control, and coded information transmission.

A. Background and Related Work

Presently, few contents are very popular and generate large percentage of the mobile traffic. Hence, in mobile network architectures, these popular contents are locally cached in the order of their popularity. They are cached at different parts of dedicated network edges such as, macrocell base stations (BS), microcell base stations (SBS), or cache enabled UE [1]. For the past decade, the research has focused on the strategies and policies of caching popular contents to increase the cache hit and content delivery probability, the number of served requests, and decrease the response delay.

In [2], the authors analyse the gains of popular video content caching strategy in D2D communications. They evaluate the downlink traffic load reduction on the cellular backhaul attained by distributed caching scheme that uses family of admissible protocol, that handles the random file and helper association. The content exchange in the D2D link is further facilitated by a D2D-aware caching policy in work of [3]. In this policy, files are partitioned into two non-overlapping groups, which is far from the most popular content (MPC) policy. In [4], authors proposed a hybrid caching strategy where identical files are cached at each BSs while different SBS cache different files. This increases the time and spatial reuse. In fact, unplanned network densification affects increases the cost of caching. In [5] an appropriate network densification, with advanced D2D-discovery mechanism, is proposed to make the D2D communication more effective and less costly.

In [6], the authors developed a contention-based multimedia content delivery protocol, which avoids possible collision among concurrent transmissions by different active transmitters. An optimal caching mechanism is proposed based on caching the most popular files to each MH. This maximizes the successful content delivery probability to the user equipment and the coverage probability. In addition, in [7], the authors focused on cooperative caching strategy, and show that, by caching files at the off-peak time, it is possible to further enhance the performance through a broadcasting technique.

In the literature, we find three general caching approaches. The first one is cooperative caching where the helpers cooperate to cache required files and pass to the user [2], [7]–[10]. The second approach uses clustering of the network elements based on parameters such as the requested file or the cooperation distance [4], [9], [10]. The third approach is distributed caching

where the required files are stored at different tier of the network to increase the areal spectral efficiency [6], [8], [9].

The aforementioned caching strategies mainly deal with the physical parameters of the network. Beyond that, in [8], the authors proposed a systematic strategy of caching contents based on the predicted location of the UE. Indeed, as the network densification increases, this caching scheme becomes very complex and demands a robust handover decision algorithm. Hence, the predicted-location based caching can benefit the synergy of using effective handover techniques, such as the one proposed in [11]. In addition, the work in [12] presented a caching strategy based on the prediction of file popularity and future requests. This study considers the ephemerality of contents popularity while most other works stick to the static Zipf property. There are still emerging opportunistic techniques that improve content caching, based on an information theory perspective. For example, in [8], [13], [14], the authors proposed the use of mature information coding schemes in content caching strategies that can improve network performance and meet the traffic demand.

In most caching strategies, content placement is controlled by a central controller that decides which content has to be cached to which MH or UE. This is more functional in the clustered and coordinated caching techniques. In some other cases, caching is done independently: the association of a file with a UE or MH is maybe done randomly.

In the popular content caching process, we have three steps: content placement, content transmission, and content delivery. In this work, we focus only on the first step and propose an intuitive approach to improve content placement performance.

B. Motivation and Contribution

The content caching strategies developed so far focus on improving the physical parameters of the network and basically are probabilistic approaches. Most of the researches stick to constant parameters such as equal sized file fragments, static popularity distribution, and non-overlapping association. These assumptions are far from the real-time network traffic behaviour. In fact, the real time traffic dynamically changes and has extremely varying parameters. Hence, characterizing real-time mobile network is complex. Because of this, we do not have optimal content caching policy. Both the physical and content related parameters hinder the solution of optimal caching.

Here, we envision that the cellular traffic content has to be considered as a resource, when referring to the request redundancy measure. Frequently requested files are important resources. Hence, it has to be properly allocated, similarly to the radio resource allocation done in [15] and [16]. Considering as a file popularity as a resource, we introduce a new resource allocation technique to the content caching policy. To that end, we use a dynamic programming method that can tackle the complexity of obtaining the optimal caching solution, subject to a physical parameter constraint. The approach is enumerating of files and optimizing files popularity gain, without dealing with the physical parameters of the medium. As a result, the proposed algorithm showed a high performance of caching strategy over the baseline strategies.

The contents are placed to the caching device of the network in such a way that we can maximize the gain, in our case the cumulative popularity, subject to a given constraint of the cache memory size. By using the Knapsack algorithm, we optimize the gain of the cache hit probability.

The rest part of this paper is organized as follows. Section II presents the system model, Section III formulates the content placement optimization problem and gives the proposed solution, and Section IV, evaluates the performance of proposed caching strategy. Lastly, Section V, contains the conclusions.

II. System model

We consider the downlink direction of a cache enabled, three tier heterogeneous cellular network. As shown in Fig. 1, the network composes of BS, SBS, and UEs which are assumed to follow a general point process (PPP). The BS and SBS are the first and

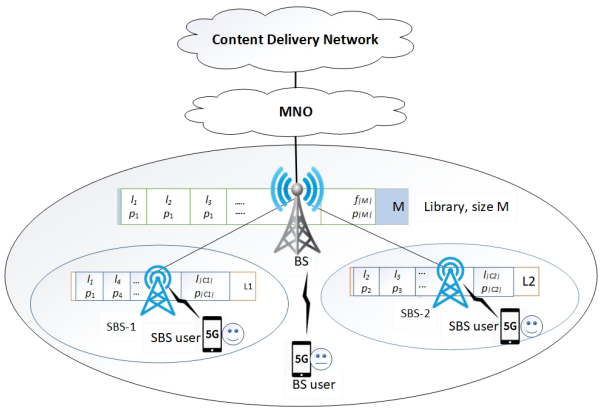

Fig. 1: *Representative mobile network. The BS caches files, mainly at off-peak time, to non-overlapping SBS.*

second tiers, respectively. The BS contains set of popular video files in its finite-sized library, $M = \{\{\rho_1, l_1\}, \{\rho_2, l_2\}, ..., \{\rho_{|M|} l_{|M|}\}\}$ where ρ_i and l_i [MB] are the popularity and length of i^{th} file, respectively. The size of these files is not necessarily equal. We consider that the SBS can cache the selected popular files in its cache memory, size L, which makes a subset of files, C, such that $C \subseteq M$. The buffer size of different SBS varies but each SBS shall be capable of caching all requested files from the UEs. In this scheme, we use the SBS and the MH interchangeably. The UEs request files independently from the cache enabled MHs, where files are cached based on their popularity. Files' request profile is collected by the MH from UEs. The frequency of request to a file accounts for the file's popularity, such that $\sum_{i=1}^{N} \rho_i \leq 1$ [17]. The popularity measure of the files is not necessarily static but may dynamically change in time.

In this work, we give attention to a hierarchical placement of traffic content on the cache-enabled devices; in the 5G and beyond mobile networks. A simple representation of the popular file placement is shown in Fig. 2, where the content is cached from the BS to the SBS. The files are sent to targeted SBS by selecting the best combination of candidate files, from a large set of combinatorial search space. Since the placement of the popular contents takes place at the off-peak time period, we assume that the backhaul link is not a performance limitation.

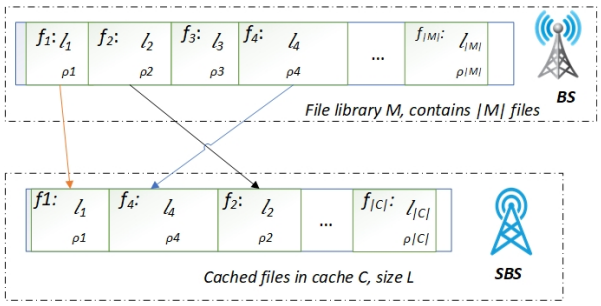

Fig. 2: *Content placement scheme. The BS can select the best popular files that can give maximum popularity at the target caching SBS.*

Symbol	Definition
ρ_i	Popularity of i^{th} file
l_i	Length of i^{th} file
γ	Zipf parameter
L	Buffer size of the MH
M	Set of popular files at BS
	M=$\{\{\rho_1, l_1\},\{\rho_2, l_2\},...,\{\rho_{\|M\|}, l_{\|M\|}\}\}$
C	Set of cached files at the helper
	$C \subseteq M$
$\|M\|$, $\|C\|$	Number of files in M and C, respectively

III. Proposed Solution

In this section, we present the enumerative approach of file selection to get the best combination of files, which are to be cached at the SBS, that maximizes the cached files popularity. For this work, we consider that each file in M will be cached without partitioning to sub parts, means we take them as indivisible objects, that either the whole file is cached or not cached at all. Hence, given independent popularity and lengths for each file, we make an exhaustive search for the whole combinations of files that can fit the cache size, called the constraint. This leads to the brute-force method which gives an optimal selection of contents from a wide range of combinatorial search space, $S = \{S_1, S_2, \ldots, S_{|M|}\}$, where S_i is a subspace that contains exactly one file, two files, three files and so on. Hence, the total search space will have $2^{|M|} - 1$ subspaces. The brute-force method gives an optimal selection but at a cost of high complexity $O(2^{|M|})$, which becomes infeasible as the $|M|$ increases. To solve this, we apply dynamic programming (DP) approach by which the problem is partitioned to smaller sub-problems and solved sequentially. The solution of the sub-problems is memoized for a repeated use in solving the next sub-problems. The optimal solution is found by combining these optimal sub-solutions. This process has computational complexity of $O(|M|L)$ so that it saves time at cost of space. To this extent, we use the well known 0/1-Knapsack problem, subject to a single constraint. This algorithm gives an optimal solution with a pseudo-polynomial time, mainly as the L gets larger [18]. In this algorithm, the file selection problem is solved by using a memoization table. Its pseudo-code is shown in Table 1, where the optimal sum of popularity of cached files is the last entry of the table. Finally, selected files can be tracked back from the memoization space.

Algorithm 1 Knapsack memoization problem

procedure Bottom-up computation
Input: $M = \{\{\rho_1, l_1\}, \{\rho_2, l_2\}, ..., \{\rho_{|M|}, l_{|M|}\}\}$
SBS cache size L
for $0 \leq j \leq L$ **do**
V[0,j]=0;
end for
for $1 \leq i \leq |M|$ **do**
for $0 \leq j \leq L$ **do**
if $j \geq l_i$ **then**
$V[i,j] = max(V[i-1,j], \rho[i] + V[i-1, j-l_i])$
else
$V[i,j] = V[i-1,j]$
end if
end for
end for
optimal value= $V[|M|, L]$

****Identify the files****
i=$|M|$+1, j=L+1;
Input: temp= V[i,j]
While V[i,j]>0
if $temp \neq V[i-1,j]$ **then**
select i^{th} file
$j = j - l_i$
$i = i - 1$
temp=V[i,j]
else
$i = i - 1$, (do not select file)
end if
end while

A. Performance Metrics

The content placement strategy is basically to maximize the file popularity gain at the SBS, hence, we evaluate the performance of the network by using the cache hit probability.

Cache hit probability: Is the probability that a requested file by a UE is found stored at the MH. In [19], it also indicates as the degree of successfully transmitting the requested file to the user. In [10], the cache hit probability is used as a measure of the ratio of the cache space allocated to the most popular files compared with non-popular files. Cache hit probability is used as objective function in many literatures such

as in [10], [6] and [17]. It is suitable for instantaneous serving of requests where there is no need of a queueing files. Hence, we used the cache hit probability (Ψ) as the measure of performance in our model. It is denoted as follows.

$$\Psi[\rho, l] = \sum_{c \in C} \rho_c \tag{1}$$

where ρ_c is the popularity of cached file c in C. It is clear that $|C| \leq |M|$.

Our target is to cache as many popular files as possible to get a maximized sum of file popularity at the MH. In other words, for all files whose IDs are assumed stored at C, we want to have the optimum cache hit probability, as follows:

$$\Psi^*[\rho, l] = \max_{C} \sum_{c \in C} \rho_c \tag{2}$$

with respect to one dimensional constraint function of:

$$\sum_{c \in C} l_c \leq L \tag{3}$$

IV. Performance Evaluation

We evaluated the performance of the network under our strategy by doing simulations. We compared our algorithm with two baseline strategies. The first baseline strategy is a greedy algorithm that caches a portion of most popular files after it arranges the files in the descending order of their popularity. The second strategy is a random algorithm that randomly chooses files, and caches to the SBS, from the search space. In this case, we iterate the selection hundred times to get the mean cache hit probability.

In all cases, unless otherwise explained, we worked the simulation with 100 video files, whose length varies. The files are standard frame rate videos files of: a 10 minutes length SDR 720p, a 2 minutes length SDR 720p, a 2 minutes HDR 2160p(4K), and a 10 minutes HDR 2160p(4K), as recommended by Google. They have an approximate file size of 3 Gb, 600 Mb, 6 Gb, and 30 Gb, respectively. The cache size of the SBS is considered to range from 10 Gb to 100 Gb, which has to be less than the sum of all file sizes. We chose the Zipf parameter $\gamma = 1$, means that the popularity is highly concentrated on a fewer number of files, and since the $\gamma \geq 1$, the network will have higher successful transmission probability [4], [8]. In the performance evaluation, we did many simulations for different cases and, here, only three main scenarios are reported.

In the first scenario, we choose two sets of HDR 2160(4K) files with equal file sizes of 30 Gb and 6 Gb, independently. In both cases, the files follow a Zipf distribution. In each set, files are considered as an equal sized fragments, like in most researchers such as [7], [9], [20]. The result in Fig. 3 shows that the greedy algorithm gives an optimal solution, same also the proposed strategy. In both results, as the file size increases, we are able to cache less number of popular files. Therefore, it gives lower cache hit probability. All algorithms have a monotonically

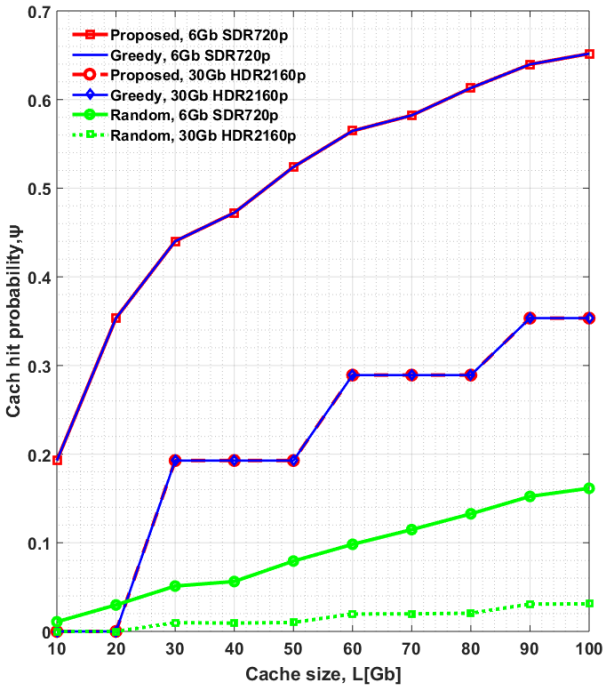

Fig. 3: *Scenario 1: In case 1 (upper plot), all files have a 6 Gb size, and in case 2 (lower plot), all files have a 30 Gb size. All files follow the Zipf popularity distribution*

increasing performance. As the file size is smaller, the hit probability resembles to strictly increasing monotonic because it can include more files in the cache. Here, the random algorithm shows almost a linear proportionality over the cache size because it has no optimization role but only tries to fit the sum of sizes of the cached files to the cache size. Hence, cache hit probability does on improve by random content caching. The proposed and greedy algorithms deal with the popularity and size trade-off. Even though we increase the cache size, we may not fill all the space. This shows that only increasing the cache size does not always improve the cache hit probability. We can see easily see this by comparing the cases of using the 6 Gb and the 30 Gb files. In the first case, the files are smaller in size so that higher numbers of files can be compacted into the cache space. But in the second case, only a few most popular files fit the cache size. As a result, the first case has higher performance and behaves more strictly monotonic than the second case, which behaves like a step function. This is the idea of bin packing dynamic programming. Clearly, the horizontal step size, seen in the plot, is equivalent to the file size.

In the second scenario, we take the same set of files as first scenario but files considered to have equal popularity. The result in Fig. 4 tells us that any random selection of files equally works with other two algorithms, in both file sizes. It is expected that since all files have the same popularity and size, the selection

process is same for all algorithms and does not matter on the popularity. But this is a worst-case assumption because, in real mobile networks, contents do not have the same length nor popularity. Hence. caching these contents will not bring any performance improvement.

The third scenario is by taking four group of mixed

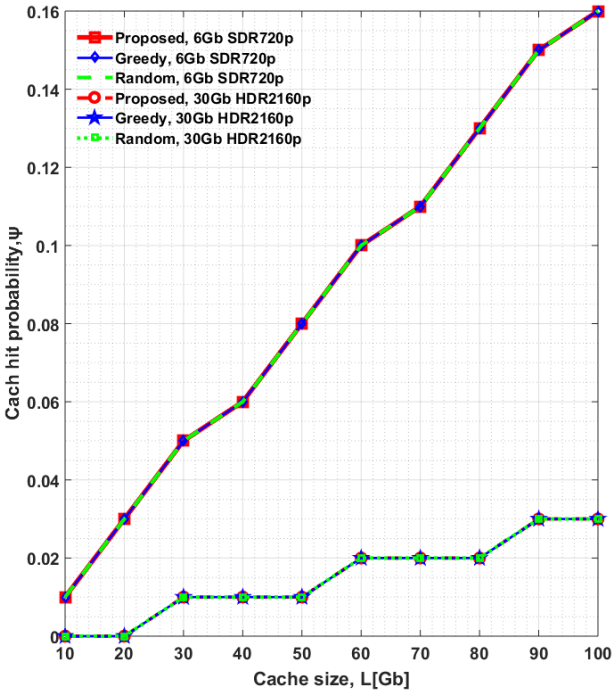

Fig. 4: *Scenario 2: All files have same size (6 Gb upper plot and 30Gb lower plot) and equal popularity* $\rho = 0.01$

files sizes. Every group, of 25% files in BS, has a 600 Mb, a 3Gb, a 6 Gb, and a 30 Gb size. We used two file popularity distributions: the Zipf and general distributions. In the general popularity distribution, we considered that approximately 30% of files in the set are most popular [12], which are assumed to generate the 90% of total popularity. In fact, this popularity distribution is more concentrated than the Zipf distribution with $\gamma = 1$. We further assume that both the file popularity and file sizes are independently and identically distributed (iid). To guarantee this, we reshuffled the association ten thousand times and took the mean of cache hit probability of ten iterations. This avoids the biased assignment of, which affects the baseline algorithms, higher popularity to bigger files. This scenario can better represent the real-time traffic contents in the HCN. In this real-time scenario, content caching becomes a constrained problem and, hence, we preferred to use a dynamic programming approach than the greedy ones.

The output in Fig. 5 clearly indicates that the proposed algorithm outperforms, by far, the baseline algorithms, in both file popularity distributions. This is because the search to all files in the BS is not blocked by sizes of the files, under the cache size constraint. The algorithm can simply check all combinations and

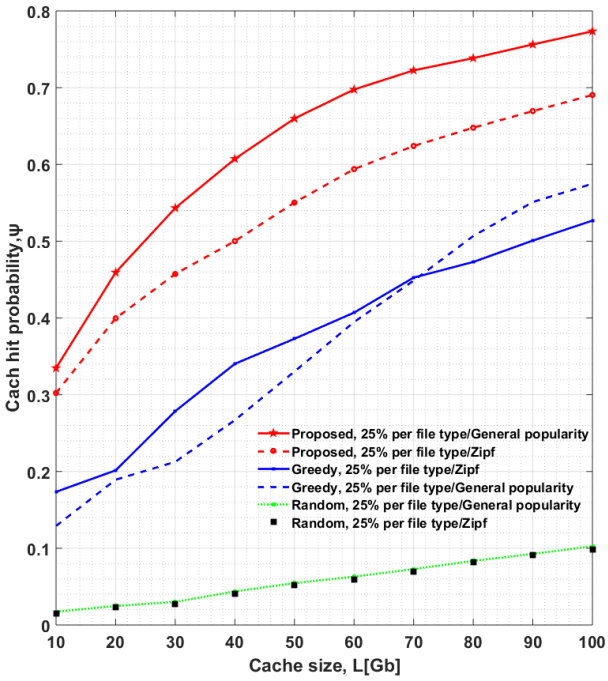

Fig. 5: *Scenario 3: Varying file size and different file popularity types. Every group of 25% files has size of 600 Mb, 3 Gb, 6 Gb, and 30 Gb. Files follow a Zipf* ($\gamma = 1$) *and general popularity distribution*

take the maximum possible file popularity.

The proposed algorithm performs better in the general popularity distribution because, in this setup, the file popularity concentrates on few numbers of files, mean the file popularity is highly skewed. Hence, the sum of popularity function vanishes faster in the general popularity than in Zipf distribution. After caching these files, as far the cache size allows, the remaining cache space is filled with many small sized files. It is also clear that the more cache size we have, a better freedom to cache files; means higher cache hit probability.

In this setup, the greedy algorithm has better performance with the Zipf distribution at lower cache size. It is because in the Zipf case, the popularity has order of a formed distribution so that the selection is only taking the front groups of most popular files. But in larger cache size, we have to include more files with varied file popularity. Thus, the selection becomes less effective with greedy strategy. We also have seen that as the skewness of the file popularity increases, the intersection point of the two plots goes to the left. This is because, the sum of popularity vanishes faster.

Also comparing with scenario 1, we notice that the proposed strategy increases the network performance as the file diversity is increased. The small steps in the greedy function indicate the size of selected file sizes, after some free spaces in the cache. This is not seen in the proposed solution because it can effectively do the exhaustive search which does not leave cache space gaps till the next popular file is fitting. Instead,

it further caches other files that are matching size but maybe less popular. This further selection increases the cache hit popularity. We also prove that, the proposed solution reduces the cache memory wastage at the MHs.

In summary, the cache hit probability can be highly improved by the proposed placement strategy. Many literatures unrealistic assume files as equal sized fragments. This is issue is solved in the proposed strategy because it can handle extremely diversified file sizes. Parallel to increasing the cache memory size, lifting up the constraint-which is also limited, we can make a nice mathematical optimization to increase the cache hit probability.

The content placement is more likely done at the off-peak time, according to a request history. In some cases, the content will be cached while it is instantaneously requested by UEs. Hence, we recommend making a prioritization strategy for the two options. It can increase the cache hit probability while reducing backhaul congestion and response time.

V. conclusions

In this work, we have considered the downlink direction of the heterogeneous cellular mobile network. We have given new insights how to deal with non-uniform parameters of the mobile data traffic such as library length, file size, and the diverse file popularity. Developing an optimal caching strategy subject to these parameters is complex. We used a dynamic programming approach to tackle this constrained optimization problem. To that end, we have used the 0/1-Knapsack problem formulation to maximize the cache hit probability under a cache size constraint. The proposed strategy outperforms the baseline algorithms. Also, the memory wastage at the MHs is reduced when using this strategy. As far the cache size allows, we can cache many popular files to the SBS by selecting from all possible file combinations. The proposed strategy can effectively handle any type of file popularity and increases content diversity. Finally, this strategy can be deployed at any network hierarchy.

Acknowledgement

This work has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska Curie grant agreement No. 722788.

References

[1] F. Boccardi, R. W. H. Jr., A. E. Lozano, T. L. Marzetta, and P. Popovski, "Five disruptive technology directions for 5g," *IEEE Communications Magazine*, vol. 52, no. 2, pp. 74–80, 2014.

[2] A. Altieri, P. Piantanida, L. R. Vega, and C. G. Galarza, "On fundamental trade-offs of device-to-device communications in large wireless networks," *IEEE Trans. Wireless Communications*, vol. 14, no. 9, pp. 4958–4971, 2015.

[3] N. Giatsoglou, K. Ntontin, E. Kartsakli, A. Antonopoulos, and C. V. Verikoukis, "D2d-aware device caching in mmwave-cellular networks," *IEEE Journal on Selected Areas in Communications*, vol. 35, no. 9, pp. 2025–2037, 2017.

[4] Y. Cui and D. Jiang, "Analysis and optimization of caching and multicasting in large-scale cache-enabled heterogeneous wireless networks," *IEEE Trans. Wireless Communications*, vol. 16, no. 1, pp. 250–264, 2017.

[5] D. Xenakis, M. Kountouris, L. F. Merakos, N. I. Passas, and C. V. Verikoukis, "Performance analysis of network-assisted D2D discovery in random spatial networks," *IEEE Trans. Wireless Communications*, vol. 15, no. 8, pp. 5695–5707, 2016.

[6] X. M. J. L. W. L. Xiaoshi Song, Yuting Geng, "Cache-enabled device to device networks with contention-based multimedia delivery," in *IEEE Access Volume 5*, 2017.

[7] C. Yang, Y. Yao, Z. Chen, and B. Xia, "Analysis on cache-enabled wireless heterogeneous networks," *IEEE Trans. Wireless Communications*, vol. 15, no. 1, pp. 131–145, 2016.

[8] N. Golrezaei, A. F. Molisch, A. G. Dimakis, and G. Caire, "Femtocaching and device-to-device collaboration: A new architecture for wireless video distribution," *IEEE Communications Magazine*, vol. 51, no. 4, pp. 142–149, 2013.

[9] L. Fan, Z. Dong, and P. Yuan, "The capacity of device-to-device communication underlaying cellular networks with relay links," *IEEE Access*, vol. 5, pp. 16 840–16 846, 2017.

[10] Z. Chen, J. Lee, T. Q. S. Quek, and M. Kountouris, "Cooperative caching and transmission design in cluster-centric small cell networks," *IEEE Trans. Wireless Communications*, vol. 16, no. 5, pp. 3401–3415, 2017.

[11] D. Xenakis, N. I. Passas, L. F. Merakos, and C. V. Verikoukis, "Handover decision for small cells: Algorithms, lessons learned and simulation study," *Computer Networks*, vol. 100, pp. 64–74, 2016.

[12] N. Carlsson and D. L. 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, 2017.

[13] Q. Jia, R. Xie, T. Huang, J. Liu, and Y. Liu, "The collaboration for content delivery and network infrastructures: A survey," *IEEE Access*, vol. 5, pp. 18 088–18 106, 2017.

[14] G. Parisis, V. Sourlas, K. V. Katsaros, W. K. Chai, G. Pavlou, and I. Wakeman, "Efficient content delivery through fountain coding in opportunistic information-centric networks," *Computer Communications*, vol. 100, pp. 118–128, 2017.

[15] D. Xenakis, D. Tsolkas, N. I. Passas, N. Alonistioti, and L. F. Merakos, "Dynamic resource allocation in adaptive wireless OFDMA systems," *Wireless Communications and Mobile Computing*, vol. 12, no. 11, pp. 985–998, 2012.

[16] M. Jiang, D. Xenakis, S. Costanzo, N. I. Passas, and T. Mahmoodi, "Radio resource sharing as a service in 5g: A software-defined networking approach," *Computer Communications*, vol. 107, pp. 13–29, 2017.

[17] J. Wen, K. Huang, S. Yang, and V. O. K. Li, "Cache-enabled heterogeneous cellular networks: Optimal tier-level content placement," *IEEE Trans. Wireless Communications*, vol. 16, no. 9, pp. 5939–5952, 2017.

[18] G. Plateau and A. Nagih, "0–1 knapsack problems," *Paradigms of Combinatorial Optimization, 2nd Edition*, pp. 215–242, 2010.

[19] B. Blaszczyszyn and A. Giovanidis, "Optimal geographic caching in cellular networks," in *2015 IEEE International Conference on Communications, ICC 2015, London, United Kingdom, June 8-12, 2015*, 2015, pp. 3358–3363.

[20] M. Afshang, H. S. Dhillon, and P. H. J. Chong, "Fundamentals of cluster-centric content placement in cache-enabled device-to-device networks," *IEEE Trans. Communications*, vol. 64, no. 6, pp. 2511–2526, 2016.