This paper is a postprint of a paper submitted to and accepted for publication in IET Communications and is subject to Institution of Engineering and Technology Copyright. The copy of record is available at the IET Digital Library.

Cooperative Caching in Fog Radio Access Networks : A Graph-Based Approach

Yanxiang Jiang[1,2,3], Xiaoting Cui[1], Mehdi Bennis[4], Fu-Chun Zheng[1,5], Baotian Fan[1], Xiaohu You[1]

[1]National Mobile Communications Research Laboratory, Southeast University, Nanjing 210096, People's Republic of China

[2]State Key Laboratory of Integrated Services Networks, Xidian University, Xi'an 710071, People's Republic of China

[3]Key Laboratory of Wireless Sensor Network and Communication, Shanghai Institute of Microsystem and Information Technology, Chinese Academy of Sciences, 865 Changning Road, Shanghai 200050, People's Republic of China

[4]Centre for Wireless Communications, University of Oulu, Oulu 90014, Finland

[5]School of Electronic and Information Engineering, Harbin Institute of Technology, Shenzhen 518055, People's Republic of China

Abstract: In this study, cooperative caching is investigated in fog radio access networks. To maximise the offloaded traffic, a cooperative caching optimisation problem is formulated. By analysing the relationship between clustering and cooperation and utilising the solutions of the knapsack problems, the above challenging optimisation problem is transformed into a clustering subproblem and a content placement subproblem. To further reduce complexity, the authors propose an effective graph-based approach to solve the two subproblems. In the graph-based clustering approach, a node graph and a weighted graph are constructed. By setting the weights of the vertices of the weighted graph to be the incremental offloaded traffics of their corresponding complete subgraphs, the objective cluster sets can be readily obtained by using an effective greedy algorithm to search for the max-weight independent subset. In the graph-based content placement approach, a redundancy graph is constructed by removing the edges in the complete subgraphs of the node graph corresponding to the obtained cluster sets. Furthermore, they enhance the caching decisions to ensure each duplicate file is cached only once. Compared with traditional approximate solutions, their proposed graph-based approach has lower complexity. Simulation results show remarkable improvements in terms of offloaded traffic by using the proposed approach.

1 Introduction

With the continuous and rapid proliferation of various intelligent devices and advanced mobile application services, wireless networks have been suffering an unprecedented data traffic pressure in recent years. Ever-increasing mobile data traffic brings tremendous load on capacity-limited fronthaul links, especially at peak traffic moments. As a promising architecture, fog radio access networks (F-RANs) can effectively offload the traffic in fronthaul links by placing popular content at fog access points (F-APs), which are equipped with limited caching resources [1]. [As has been pointed out by Peng *et al*.[1], the fibre is generally used as the backhaul link, and F-APs are generally interfaced to the baseband unit pool in the cloud computing layer through the fronthaul links, which may well be wireless links due to costs. Furthermore, even if the fronthaul bandwidth is sufficiently large and the requested file is fetched from the cloud server, the file security and the user privacy cannot be fundamentally guaranteed. The reason is that the requested file stored in the cloud server is more vulnerable to network attacks and thefts than in the F-APs.] Due to storage constraints and fluctuant spatio-temporal traffic demands, cooperative caching is an effective way to increase the offloaded traffic.

Recently, there have been a lot of works on cooperative caching. In [2], a cooperative caching and delivery policy was proposed to minimise the latency, where each base station (BS) and user equipment (UE) cached files according to the request probability independently. However, the caching decision of one BS was influenced by that of the neighbouring cooperative BSs, and different BSs should cache diverse files in a cooperative manner [3, 4]. In [5–7], the cooperative content placement strategy for the given cache nodes cluster was studied. In [5], a cooperative content placement strategy was proposed to maximise the service probability, where the storage space of each BS in the given cluster was divided into a proportion for caching the same files and a rest proportion for caching different files. In [6], a cooperative caching algorithm for multiple operators was proposed to maximise the delay savings, where all the cache nodes in the given cluster firstly cached the globally popular files together and then cached the locally popular files independently. In [7], a cooperative content placement method was proposed to minimise the latency for multi-cell cooperative networks, where a heuristic greedy algorithm with limited performance guarantee was developed. In [8–10], the cooperative content placement strategy for unknown cache nodes cluster was studied. In [8], the uncoded and coded cooperative content assignment strategies were proposed to minimise the expected downloading time, where the connectivity graph between UE and BSs was used to reflect the cooperation relationship among neighbouring BSs. By optimising relay clustering and content placement in a joint manner, a cooperative caching strategy was developed to minimise the outage probability in [9], where identical files were cached among the relays in each cluster for simplicity. Based on the similarities among users requesting similar contents, a user clustering and cooperative caching algorithm to improve the cache hit rate was proposed in [10]. In [11], a cluster-based cooperative caching approach with mobility prediction was proposed to construct reliable and stable communications among vehicles, which can group vehicles via the similar characteristics of vehicle mobility. Based on the content diversity and piecewise interest similarity, a clustering scheme of sectionalised cooperative caching is proposed to minimise the total transmission delay in [12]. In [13], a user-centric open-loop cooperative transmission and interference-aware dual-mode caching scheme was proposed to maximise the approximate average successful transmission probability, and a low complexity algorithm which combines enumeration and submodular maximisation was designed. Recently, the emerging mobile edge caching (MEC) strategy based on reinforcement learning has also been studied extensively. In [14], a multi-agent reinforcement learning-based cooperative content caching policy for the MEC architecture was proposed to

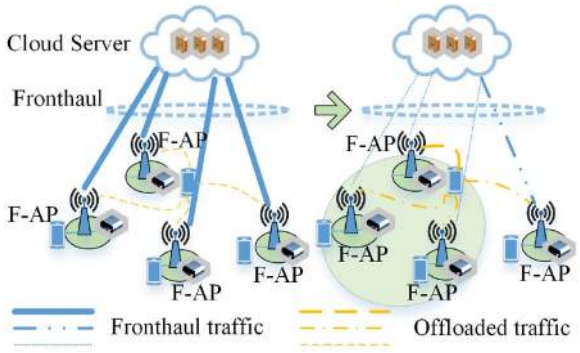

Fig. 1 *Illustration of the cooperative caching scenario in F-RANs*

maximise the total expected caching reward when the preference of users is unknown and only the historical demand information can be obtained. In [15], a Q-learning algorithm-based wireless cooperative caching framework was designed to maximise the sum mean opinion score of users in the network, which can be both used online with extension performances and trained offline with historical information.

However, the prior works on cooperative content placement tend to exploit global content popularity rather than the local content popularity, which might not even replicate the global content popularity. The local content popularity indeed reflects user interest at the coverage of each cache node and might be different from each other [16, 17]. It was investigated in [18, 19] that the cooperative content placement algorithms based on the local content popularity could obtain lower delay or higher cache hit rate than that based on the global content popularity.

Motivated by the aforementioned discussions, the main contributions of this study are summarised below.

- We propose a new idea for solving the challenging cooperative caching optimisation problem based on the local content popularity. Analysing the relationship between clustering and cooperation and utilising the solutions of the knapsack problems, we transform the cooperative caching optimisation problem into a clustering subproblem and a content placement subproblem.
- We propose a graph-based clustering approach. Constructing a node graph and a weighted graph, we transform the clustering subproblem into an equivalent 0–1 integer programming problem. Furthermore, we propose an effective greedy algorithm to search for the objective cluster sets.
- We propose a graph-based content placement approach. Constructing a redundancy graph based on the obtained cluster sets, we determine the duplicate files that will indeed cause cache redundancy at each edge and further enhance the caching decisions for each file. Correspondingly, all the possible cache redundancy can be eliminated by caching each duplicate popular file only once.

The rest of this paper is organised as follows. In Section 2, the system model and problem formulation are briefly described. In Section 3, the problem transformation is presented. The proposed graph-based cooperative caching scheme including clustering and content placement is presented in Section 4. Simulation results are shown in Section 5. Final conclusions are drawn in Section 6.

2 System model and problem formulation

Consider a cooperative caching scenario in F-RANs as illustrated in Fig. 1, which consists of a cloud server, M F-APs, and a certain number of users. The cloud server can be accessed by the F-APs via fronthaul links. Let $\mathscr{M} = \{1, 2, \ldots, m, \ldots, M\}$ denote the F-AP set. Assume that neighbouring F-APs can share files and cooperate with each other [20]. Whether two F-APs can cooperate or not depend on how well they satisfy some certain rules. According to [21], F-APs with appropriate distance and load difference from each other are more likely to cooperate. Let $\mathscr{S}_m$ denote the set of all the co-operators of F-AP m. Without loss of generality, assume that all the files have the same size of L bits, each F-AP has the same storage space and can store up to K files from the content library $\mathscr{F} = \{1, 2, \ldots, f, \ldots, F\}$ located in the cloud server. Let p_{mf} denote the request probability of file f at F-AP m (referred to as the local content popularity). Assume that the request probability at each F-AP is stationary during the given time period. Let λ_m denote the aggregate request arrival rate at F-AP m, and $w_m = \lambda_m / \sum_{m' \in \mathscr{M}} \lambda_{m'}$ denote the ratio of the traffic load at F-AP m to the sum load of the M F-APs.

Let $x_{mf} \in \{1, 0\}$ denote the caching decision of file f at F-AP m, where $x_{mf} = 1$ if file f is cached at F-AP m and $x_{mf} = 0$ otherwise. Let $x^{\mathrm{l}}_{mf} \in \{1, 0\}$ denote the local state of file f at F-AP m and its co-operators, where $x^{\mathrm{l}}_{mf} = 1$ if file f is successfully cached locally; $x^{\mathrm{l}}_{mf} = 0$ if file f is not cached locally and must be fetched from the cloud server. Then, x^{l}_{mf} can be expressed as follows:

$$x^{\mathrm{l}}_{mf} = x_{mf} + (1 - x_{mf}) \left[1 - \prod_{m' \in \mathscr{S}_m} (1 - x_{m'f}) \right]. \tag{1}$$

Once the requested file is cached locally, the traffic in the fronthaul links can be offloaded. Let T denote the offloaded traffic for all the considered M F-APs. Then, it can be expressed as follows:

$$T = \sum_{m \in \mathscr{M}} \sum_{f \in \mathscr{F}} \lambda_m p_{mf} x^{\mathrm{l}}_{mf} L. \tag{2}$$

Note that the offloaded traffic increases with the number of locally cached files, and decreases with duplicate cached files at the requested F-APs and their co-operators. The caching decisions should be determined cooperatively by the neighbouring F-APs for a larger number of unduplicated cached files.

To maximise the offloaded traffic, the co-operators should be neighbouring F-APs with closer distance and greater load difference. [As for the communication interface and protocol among F-APs during the traffic offload, the corresponding analysis in [17] can be adopted where the partition-based content placement method was employed. In [17], two learning-based edge caching architectures have been proposed where both the signalling overhead among clustered F-APs and the computational burden undertaken by the F-APs can be greatly reduced. We also remark here that the signalling overhead can be further reduced by setting a cluster head for the clustered F-APs.] The selected F-APs can efficiently offload traffic among each other and are more likely to cooperative with each other [21]. Let d_m denote the geographical coordinate of F-AP m in the Euclidean space, $D_{mm'} = \| d_m - d_{m'} \|_2$ denote the distance between F-AP m and F-AP m', and $L_{mm'} = \| \lambda_m - \lambda_{m'} \|_2$ denote the load difference between F-AP m and F-AP m'. Then, the cooperative caching optimisation problem can be formulated as follows:

$$\max_{x_{mf}} \quad T \tag{3}$$

$$\text{s.t.} \quad D_{mm'} \le \gamma^{\mathrm{d}}, \quad \forall m \in \mathscr{M}, \forall m' \in \mathscr{S}_m, \tag{3a}$$

$$L_{mm'} \ge \gamma^{\mathrm{l}}, \quad \forall m \in \mathscr{M}, \forall m' \in \mathscr{S}_m, \tag{3b}$$

$$x_{mf} \in \{1, 0\}, \quad \forall m \in \mathscr{M}, \forall f \in \mathscr{F}, \tag{3c}$$

$$\sum_{f \in \mathscr{F}} x_{mf} \le K, \quad \forall m \in \mathscr{M}, \tag{3d}$$

where γ^{d} and γ^{l} denote the distance threshold and the load threshold, respectively.

The objective of this study is to find the optimal caching decisions $\{x_{mf} \mid m \in \mathscr{M}, f \in \mathscr{F}\}$ by maximising the offloaded traffic using cooperative caching in F-RANs.

3 Problem transformation

The optimisation problem in (3) is a 0–1 integer programming problem, which is non-deterministic polynomial-hard [2, 6]. A dynamic programming approach is generally required for obtaining a globally optimal solution [22]. However, such an approach has an exponential complexity with respect to (w.r.t.) the number of F-APs and the size of the content library, and it is computationally impracticable even for a small size network. In the previous works [8, 22], by reformulating the original problem into a matroid constrained monotone submodular optimisation problem, the approximate solutions with limited performance can be obtained. However, by using the above approach, it incurs a long running time to evaluate the marginal value of the objective function.

In fact, by utilising the relationship between clustering and cooperation, the co-operators of an F-AP can be divided into intra-cluster co-operators, inter-cluster co-operators, and non-clustered co-operators. Certainly there exist differences among these three types of co-operators. If the requested file has already been cached at the serving F-AP or the intra-cluster co-operators, it will be transmitted directly to the requesting user. Else if the requested file has been cached at the inter-cluster co-operators, it will be forwarded from the inter-cluster co-operators to the serving F-AP. Otherwise, if the requested file has been cached at the non-cluster co-operators, it cannot be forwarded to the serving F-AP and must be fetched from the cloud server.

According to the above discussions, the objective function of the cooperative caching optimisation problem in (3) can be decomposed into three items. All three items indicate that the offloaded traffic is affected by the clustering strategy. In addition, the first item indicates that the offloaded traffic is also affected by the cached files at the requested F-APs and their intra-cluster co-operators. The second item indicates that the offloaded traffic is also affected by the cached files at the non-clustered co-operators and the inter-cluster co-operators of the requested F-APs. The third item indicates that the offloaded traffic is also affected by the duplicate cached files between the requested F-APs (or their intra-cluster co-operators) and their inter-cluster co-operators. In summary, all three items indicate that the solution of the original optimisation problem requires to determine clusters and content placement. Therefore, in this study, we propose to transform the challenging cooperative caching optimisation problem into a clustering subproblem and a content placement subproblem.

3.1 Clustering and cooperation

Cooperative F-APs can form a cluster to make the storage space in a cluster be seen as an entirety [23]. Correspondingly, clustering can increase content diversity. Any two F-APs in a cluster can cooperate with each other. However, two F-APs that can cooperate may not necessarily be members of the same cluster.

Assume that the considered M F-APs can constitute N [note that the number of disjoint clusters N can be determined by our proposed graph-based clustering approach presented in the following section] disjoint clustered sets denoted by $\mathscr{M}_n^{\mathrm{c}}$ for $n \in \mathscr{N} = \{1, 2, \ldots, N\}$ and one non-clustered set denoted by $\mathscr{M}^{\mathrm{n}}$, and the set size of $\mathscr{M}_n^{\mathrm{c}}$ is denoted by S_n. Disjoint clustering makes one F-AP only be a member of one cluster, which ensures exclusive and sufficient usage of its storage space to all the users in the cluster. Correspondingly, the following relationship can be readily established:

$$\mathscr{M} = \left(\bigcup_{n \in \mathscr{N}} \mathscr{M}_n^{\mathrm{c}} \right) \cup \mathscr{M}^{\mathrm{n}}, \tag{4}$$

$$\mathscr{M}_n^{\mathrm{c}} \cap \mathscr{M}_{n'}^{\mathrm{c}} = \varnothing, \quad \forall n, n' \in \mathscr{N}, n \neq n'. \tag{5}$$

Without loss of generality, let $\mathscr{S}_m^1$, $\mathscr{S}_m^2$, and $\mathscr{S}_m^3$ denote the set of intra-cluster co-operators, inter-cluster co-operators, and non-clustered co-operators of F-AP m, respectively. Define

$$\mathscr{S}_m = \mathscr{S}_m^1 \cup \mathscr{S}_m^2 \cup \mathscr{S}_m^3. \tag{6}$$

Then, the following relationship can be readily established:

$$\mathscr{S}_m^i \cap \mathscr{S}_m^j = \varnothing, \quad \forall i, j \in \{1, 2, 3\}, i \neq j, \tag{7}$$

$$m \cup \mathscr{S}_m^1 = \mathscr{M}_n^{\mathrm{c}}, \quad m \in \mathscr{M}_n^{\mathrm{c}}, \tag{8}$$

$$\mathscr{S}_m^1 = \mathscr{S}_m^3 = \varnothing, \quad m \in \mathscr{M}^{\mathrm{n}}. \tag{9}$$

Let p_{nf} denote the request probability of file f in cluster n. Then, according to [24], we have

$$p_{nf} = \sum_{m \in \mathscr{M}_n^{\mathrm{c}}} p_{mf} \frac{w_m}{\sum_{m' \in \mathscr{M}_n^{\mathrm{c}}} w_{m'}}. \tag{10}$$

Introducing p_{nf}, we can conveniently calculate the offloaded traffic through fetching files that are cached at the requested F-APs and their intra-cluster co-operators.

Assume that cluster n can cache $K_n = S_n K$ different files. Generally, $S_n K \ll F$. Let $x_{nf} \in \{1, 0\}$ denote the caching decision of file f in cluster n, where $x_{nf} = 1$ if file f is cached at any F-AP in cluster n and $x_{nf} = 0$ otherwise. Then, we have

$$x_{nf} = 1 - \prod_{m \in \mathscr{M}_n^{\mathrm{c}}} (1 - x_{mf}), \tag{11}$$

$$= 1 - (1 - x_{mf}) \prod_{m' \in \mathscr{S}_m^1} (1 - x_{m'f}), \quad m \in \mathscr{M}_n^{\mathrm{c}}, \tag{12}$$

$$\sum_{f \in \mathscr{F}} x_{nf} \leq K_n, \quad n \in \mathscr{N}. \tag{13}$$

3.2 Objective function decomposition

Substituting (6) into (1), the local state x_{mf}^{l} of the requested file f at F-AP $m \in \mathscr{M}$ and its co-operators can be expressed in an equivalent form in (14)

$$\begin{aligned} x_{mf}^{\mathrm{l}} = x_{mf} + (1 - x_{mf}) \left[1 - \prod_{m' \in \mathscr{S}_m^1} (1 - x_{m'f}) \right] \\ + (1 - x_{mf}) \prod_{m' \in \mathscr{S}_m^1} (1 - x_{m'f}) \left[1 - \prod_{m' \in \mathscr{S}_m^2 \cup \mathscr{S}_m^3} (1 - x_{m'f}) \right]. \end{aligned} \tag{14}$$

When $m \in \mathscr{M}^{\mathrm{n}}$, according to (9) and (14), x_{mf}^{l} can be further expressed as follows:

$$x_{mf}^{\mathrm{l}} = x_{mf} + (1 - x_{mf}) \left[1 - \prod_{m' \in \mathscr{S}_m^2} (1 - x_{m'f}) \right], \quad m \in \mathscr{M}^{\mathrm{n}}. \tag{15}$$

When $m \in \mathscr{M}_n^{\mathrm{c}}$, according to (12) and (14), x_{mf}^{l} can be further expressed as follows:

$$x_{mf}^{\mathrm{l}} = x_{nf} + (1 - x_{nf}) \left[1 - \prod_{m' \in \mathscr{S}_m^2 \cup \mathscr{S}_m^3} (1 - x_{m'f}) \right], \quad m \in \mathscr{M}_n^{\mathrm{c}}, n \in \mathscr{N}. \tag{16}$$

In the following, for description convenience, define

$$x_{mf}^{2} = 1 - \prod_{m' \in \mathscr{S}_m^2} (1 - x_{m'f}), \tag{17}$$

$$x_{mf}^{23} = 1 - \prod_{m' \in \mathscr{S}_m^2 \cup \mathscr{S}_m^3} (1 - x_{m'f}). \tag{18}$$

Substitute (4), (15), and (16) into (2). Then, the objective function of the original optimisation problem in (3) can be expressed in an equivalent form in (19)

$$T = \sum_{n \in \mathcal{N}} \sum_{m \in \mathcal{M}_n^{\mathrm{c}}} \sum_{f \in \mathcal{F}} \lambda_m p_{mf} \left[x_{nf} + (1 - x_{nf}) x_{mf}^{23} \right] L + \sum_{m \in \mathcal{M}^{\mathrm{n}}} \sum_{f \in \mathcal{F}} \lambda_m p_{mf} \left[x_{mf} + (1 - x_{mf}) x_{mf}^{2} \right] L . \quad (19)$$

For all the considered M F-APs, let T^{c} denote the offloaded traffic through fetching files that are cached at the requested F-APs and their intra-cluster co-operators, T^{n} denote the offloaded traffic through fetching files that are cached at the non-clustered co-operators and the inter-cluster co-operators of the requested F-APs, and T^{d} denote the offloaded traffic through fetching duplicate files that are cached between the requested F-APs (or their intra-cluster co-operators) and their inter-cluster co-operators, respectively. Then, (19) can be decomposed into three items as follows:

$$T = T^{\mathrm{c}} + T^{\mathrm{n}} - T^{\mathrm{d}}, \quad (20)$$

where

$$T^{\mathrm{c}} = \sum_{n \in \mathcal{N}} \sum_{m \in \mathcal{M}_n^{\mathrm{c}}} \sum_{f \in \mathcal{F}} \lambda_m p_{mf} x_{nf} L + \sum_{m \in \mathcal{M}^{\mathrm{n}}} \sum_{f \in \mathcal{F}} \lambda_m p_{mf} x_{mf} L, \quad (21)$$

and T^{n} and T^{d} are expressed in (22) and (23), respectively.

$$T^{\mathrm{n}} = \sum_{n \in \mathcal{N}} \sum_{m \in \mathcal{M}_n^{\mathrm{c}}} \sum_{f \in \mathcal{F}} \lambda_m p_{mf} x_{mf}^{23} L + \sum_{m \in \mathcal{M}^{\mathrm{n}}} \sum_{f \in \mathcal{F}} \lambda_m p_{mf} x_{mf}^{2} L . \quad (22)$$

$$T^{\mathrm{d}} = \sum_{n \in \mathcal{N}} \sum_{m \in \mathcal{M}_n^{\mathrm{c}}} \sum_{f \in \mathcal{F}} \lambda_m p_{mf} x_{nf} x_{mf}^{23} L + \sum_{m \in \mathcal{M}^{\mathrm{n}}} \sum_{f \in \mathcal{F}} \lambda_m p_{mf} x_{mf} x_{mf}^{2} L . \quad (23)$$

It can be readily seen from (21) that T^{c} can be maximised if the cluster sets are determined, the most popular K_n files in each cluster and the most popular K files at each non-clustered F-AP are cached, respectively. It can be readily seen from (22) that T^{n} can be maximised if the cluster sets are determined, the most popular K files at the inter-cluster co-operators and the non-clustered co-operators of a clustered F-AP are cached at the clustered F-AP, and the most popular K files at the inter-cluster co-operators of a non-clustered F-AP are cached at the non-clustered F-AP. It can be readily seen from (23) that T^{d} can be minimised if the cluster sets are determined, different files are cached between a clustered F-AP and its inter-cluster co-operators (or its non-clustered co-operators), different files are cached between a non-clustered F-AP and its inter-cluster co-operators.

According to the above presentation, firstly, if a clustered F-AP does not have inter-cluster co-operators and non-clustered co-operators, or a non-clustered F-AP does not have inter-cluster co-operators, the cache files at this F-AP cannot be determined through maximising T^{n} whereas they must be determined through maximising T^{c}. Secondly, for the problem of maximising T^{n}, the number of most popular files at the inter-cluster co-operators (or the non-clustered co-operators) of a clustered F-AP that should be cached at the clustered F-AP, and the number of popular files at the inter-cluster co-operators of a non-clustered F-AP that should be cached at the non-clustered F-AP cannot be determined. It is hardly possible to solve the problem of maximising T^{n}. Thirdly, for the clustered F-APs which have inter-cluster co-operators or non-clustered co-operators, and the non-clustered F-APs which have inter-cluster co-operators, both maximising T^{c} and maximising T^{n} require them to cache the most popular files. The difference between maximising T^{c} and maximising T^{n} lies in the caching locations of these files between each pair of a clustered F-AP and its inter-cluster co-operator (or non-clustered co-operator), and between each pair of a non-clustered F-AP and its inter-cluster co-operator. There exists an exchange relationship between the caching locations of the above F-AP pairs. Finally, once the popular files at the clustered and non-clustered F-APs are determined, the duplicate cache files between a clustered F-AP and its inter-cluster co-operators (or non-cluster co-operators), and the duplicate cache files between a non-clustered F-AP and its inter-cluster co-operators can be determined. By reducing the number of duplicate cached files, caching a duplicate popular file at one F-AP and replacing it by a new popular file at the other F-AP, T^{d} can then be minimised. Based on the above analysis, we propose to solve the cooperation caching optimisation problem by firstly maximising T^{c} and further minimising T^{d}.

3.3 Optimisation problem reformulation

From (21), we can see that T^{c} is affected by the clustering strategy and the caching decisions $\{x_{nf}, x_{mf} \mid f \in \mathcal{F}, n \in \mathcal{N}, m \in \mathcal{M}^{\mathrm{n}}\}$. If the cluster sets are determined, T^{c} can be maximised through solving the $N + |\mathcal{M}^{\mathrm{n}}|$ independent knapsack problems for each $n \in \mathcal{N}$ and each $m \in \mathcal{M}^{\mathrm{n}}$ [2]. Sort p_{mf} and p_{nf} in descending order. Let p_{mf}^{o} and p_{nf}^{o} denote the request probability of the f th most popular file at F-AP m and in cluster n, respectively. According to the solutions of the knapsack problems [2], and the caching storage constraints in (3c), (3d), and (13), we have

$$T^{\mathrm{c}} = \sum_{n \in \mathcal{N}} \sum_{m \in \mathcal{M}_n^{\mathrm{c}}} \sum_{f=1}^{K_n} \lambda_m p_{nf}^{\mathrm{o}} L + \sum_{m \in \mathcal{M}} \sum_{f=1}^{K} \lambda_m p_{mf}^{\mathrm{o}} L . \quad (24)$$

Define

$$T_m = \sum_{f=1}^{K} \lambda_m p_{mf}^{\mathrm{o}} L, \quad (25)$$

$$T_n^{\mathrm{i}} = \sum_{m \in \mathcal{M}_n^{\mathrm{c}}} \lambda_m \left(\sum_{f=1}^{K_n} p_{nf}^{\mathrm{o}} - \sum_{f=1}^{K} p_{mf}^{\mathrm{o}} \right) L, \quad (26)$$

$$T^{\mathrm{i}} = \sum_{n \in \mathcal{N}} T_n^{\mathrm{i}}, 8 \quad (27)$$

where T_m denotes the offloaded traffic at F-AP m through fetching files that are cached in its own storage space, T_n^{i} denotes the incremental offloaded traffic of cluster n, and T^{i} denotes the incremental offloaded traffic of all the N clusters. Then, (24) can be further expressed in an equivalent form as follows [25]:

$$T^{\mathrm{c}} = T^{\mathrm{i}} + \sum_{m \in \mathcal{M}} T_m . \quad (28)$$

It can be readily seen that the second item in the right-hand side of (28) is unaffected by the clustering strategy, and that maximising T^{c} under the constraints in (3a)–(3d) is equivalent to maximising T^{i} under the constraints in (3a)–(3b). Therefore, we can reformulate the clustering subproblem to maximise T^{c} as follows:

$$\max_{\{\mathcal{M}_n^{\mathrm{c}}\}_{n \in \mathcal{N}}, \mathcal{M}^{\mathrm{n}}} \quad T^{\mathrm{i}} \quad (29)$$

$$\text{s.t.} \quad (3a), (3b) . \quad (29a)$$

Solving the above optimisation problem, we can obtain the clustered and non-clustered F-AP sets. Let $\mathcal{F}_n^{\mathrm{c}}$ and $\mathcal{F}_m^{\mathrm{n}}$ denote the set of K_n most popular files in cluster n and the set of K most popular files at F-AP $m \in \mathcal{M}^{\mathrm{n}}$, respectively. Then, they can be expressed as follows:

$$\mathscr{F}_n^{\mathrm{c}} = \left\{ f \middle| p_{n1}^{\mathrm{o}} \geq p_{n2}^{\mathrm{o}} \geq \cdots \geq p_{nf}^{\mathrm{o}} \geq \cdots \geq p_{nK_n}^{\mathrm{o}} \right\}, \quad n \in \mathscr{N}, \quad (30)$$

$$\mathscr{F}_m^{\mathrm{n}} = \left\{ f \middle| p_{m1}^{\mathrm{o}} \geq p_{m2}^{\mathrm{o}} \geq \cdots \geq p_{mf}^{\mathrm{o}} \geq \cdots \geq p_{mK}^{\mathrm{o}} \right\}, \quad m \in \mathscr{M}^{\mathrm{n}}. \quad (31)$$

Correspondingly, the caching decisions $\left\{ x_{nf}, x_{mf} \middle| f \in \mathscr{F}, n \in \mathscr{N}, m \in \mathscr{M}^{\mathrm{n}} \right\}$ through maximising T^{c} can be expressed as follows:

$$x_{nf} = \begin{cases} 1, & f \in \mathscr{F}_n^{\mathrm{c}}, \\ 0, & f \in \mathscr{F} \setminus \mathscr{F}_n^{\mathrm{c}}, \end{cases} \quad (32)$$

$$x_{mf} = \begin{cases} 1, & f \in \mathscr{F}_m^{\mathrm{n}}, \\ 0, & f \in \mathscr{F} \setminus \mathscr{F}_m^{\mathrm{n}}. \end{cases} \quad (33)$$

Substitute (32) and (33) into (23). Then, T^{d} can be expressed in an equivalent form as

$$T^{\mathrm{d}} = \sum_{n \in \mathscr{N}} \sum_{m \in \mathscr{M}_n^{\mathrm{c}}} \sum_{f \in \mathscr{F}_n^{\mathrm{c}}} \lambda_m p_{mf} x_{mf}^{23} L + \sum_{m \in \mathscr{M}^{\mathrm{n}}} \sum_{f \in \mathscr{F}_m^{\mathrm{n}}} \lambda_m p_{mf} x_{mf}^{2} L \quad (34)$$

Therefore, we can reformulate the content placement subproblem to minimise T^{d} as follows:

$$\min_{x_{mf}} \quad T^{\mathrm{d}} \quad (35)$$

$$\text{s.t.} \quad (3a), (3b), (3c), (3d). \quad (35a)$$

For convenience, a summary of major notations is presented in Table 1.

4 Proposed graph-based cooperative caching scheme

In the previous section, we have transformed the challenging cooperative caching optimisation problem into a clustering subproblem and a content placement subproblem. The clustering subproblem in (29) and the content placement subproblem in (35) fall into the scope of combinatorial programming [21, 25]. A brute force approach is generally required to obtain the globally optimal solution of each subproblem. However, such an approach has an exponential complexity w.r.t. the number of F-APs and the number of disjoint cluster sets or the sizes of popular file sets $\mathscr{F}_n^{\mathrm{c}}$ and $\mathscr{F}_m^{\mathrm{n}}$. Although its computational complexity is indeed reduced compared to the original dynamic programming approach, it is still computationally impracticable even for a small size network. By mapping each F-AP as one vertex in a graph, the candidate cluster can be represented by its subgraph [26]. By mapping each obtained subgraph as the vertex in a new graph, the disjoint cluster sets can be represented by an independent subset of the vertex set of this new graph. According to the graph theory [27], all the subgraphs of a graph and the independent subset of the vertex set of a graph can be obtained in polynomial time complexity. Correspondingly, the clustering subproblem can be solved in polynomial time complexity. Furthermore, by mapping each pair of cooperative F-APs which are not in the same cluster as two vertices that are connected by one edge in a graph, all the edges can be traversed to control the cached files at the corresponding paired F-APs and the duplicate cached files can then be eliminated. According to the graph theory [27], all the edges in a graph can be found in polynomial time complexity. Correspondingly, the content placement subproblem can also be solved in polynomial time complexity. Therefore, we commit to an effective graph-based approach to solve the clustering subproblem and the content placement subproblem, respectively.

Table 1 Summary of major notations

M, $\mathscr{M}$, m,	number of the considered F-APs, set of the M F-APs,
$\mathscr{S}_m$, $\mathscr{S}_m^1$, $\mathscr{S}_m^2$, $\mathscr{S}_m^3$	index of F-AP, set of all the co-operators of F-AP m, set of intra-cluster co-operators of F-AP m, set of inter-cluster co-operators of F-AP m, set of non-clustered co-operators of F-AP m
n, $\mathscr{M}_n^{\mathrm{c}}$, $\mathscr{M}^{\mathrm{n}}$, S_n	index of clusters, set of F-APs in cluster n, set of non-clustered F-APs, set size of $\mathscr{M}_n^{\mathrm{c}}$
f, $\mathscr{F}$, F	index of files, content library, library size
K, K_n	storage size of each F-AP, storage size of cluster n
λ_m, w_m	aggregate request arrival rate at F-AP m, ratio of the traffic load at F-AP m to the sum load of the M F-APs
p_{mf}, p_{nf}, p_{mf}^{o}, p_{nf}^{o}	request probability of file f at F-AP m, request probability of file f in cluster n, request probability of the f th most popular file at F-AP m, request probability of the f th most popular file in cluster n
x_{mf}, x_{nf}, x_{mf}^{l}	caching decision of file f at F-AP m, caching decision of file f in cluster n, local state of file f at F-AP m and its co-operators
T, T^{c}, T^{n}, T^{d}	whole offloaded traffic for all the M F-APs, offloaded traffic for all the M F-APs through fetching files that are cached at the requested F-APs and their intra-cluster co-operators, offloaded traffic for all the M F-APs through fetching files that are cached at the non-clustered co-operators and the inter-cluster co-operators of the requested F-APs, offloaded traffic for all the M F-APs through fetching duplicate files that are cached between the requested F-APs (or their intra-cluster co-operators) and their inter-cluster co-operators
T_m, T^{i}	offloaded traffic at F-AP m through fetching files that are cached in its own storage space, incremental offloaded traffic of the N clusters
d_m, $D_{mm'}$, $L_{mm'}$, γ^{d}, γ^{l}	geographical coordinate of F-AP m in the Euclidean space, distance between F-AP m and F-AP m', load difference between F-AP m and F-AP m', distance threshold, load threshold

4.1 Proposed graph-based clustering approach

4.1.1 Description of the proposed approach: In our proposed graph-based clustering approach, firstly, all the considered M F-APs are checked to determine which pair satisfies the constraints in (29a). It is already known that F-APs with appropriate distance and load differences from each other are more likely to cooperate together [21]. Then, according to the checking results, the node graph denoted by $\mathscr{G}^{\mathrm{n}} = (\mathscr{M}, \mathscr{E})$ is constructed, whose vertex set denoted by $\mathscr{M}$ is the F-AP set and whose edge set denoted by $\mathscr{E}$ reflects the distance and load difference among the F-APs. In $\mathscr{G}^{\mathrm{n}}$, two vertices are connected through an edge if their representing F-APs can cooperate with each other. Note that one subgraph of $\mathscr{G}^{\mathrm{n}}$, any vertex of which can connect through an edge with a certain vertex in the same subgraph, represents one cluster which consists of a certain number of cooperative F-APs, and one complete subgraph of $\mathscr{G}^{\mathrm{n}}$, any two vertices of which can connect through an edge, essentially represents one candidate cluster of the optimisation problem in (29) whose cluster members can cooperative with each other. We point out here that there may exist a certain vertex not belonging to any subgraph of $\mathscr{G}^{\mathrm{n}}$, which means that its representing F-AP is non-clustered. For illustration, a node graph with 13 vertices as shown in Fig. 2 is taken for example. According to the above descriptions, seeking candidate clusters is equivalent to searching for complete subgraphs in $\mathscr{G}^{\mathrm{n}}$. The algorithm of searching for complete subgraphs will be presented in detail in Section 4.1.2.

Let $\mathscr{H} = \{h_1, h_2, \ldots, h_n, \ldots, h_{N'}\}$ denote the complete subgraph set that has been obtained through the above searching algorithm, where N' denotes the number of complete subgraphs so obtained. It is clear that $\{\mathscr{M}_n^{\mathrm{c}}\}_{n=1}^{N} \subseteq \mathscr{H}$. Then, a weighted graph denoted by

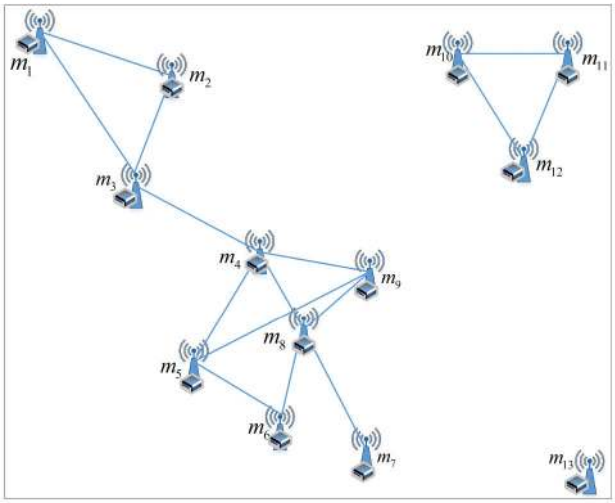

Fig. 2 *Illustration of a node graph including 13 vertices*

Input: $\mathcal{G}^{\mathrm{n}}$
Output: $\mathcal{G}^{\mathrm{m}}$
1: **for** each $\mathcal{T}_m^{\mathrm{o}} \in \mathcal{T}$ **do**
2: Initialize $i = 0, \mathcal{T}_i^{\mathrm{t}} = \varnothing, \mathcal{T}' = \mathcal{T}_m^{\mathrm{o}}$;
3: **for** each $j \in \mathcal{T}'$ **do**
4: **if** $\mathcal{T}_m^{\mathrm{o}}$ or $\mathcal{T}_{i'}^{\mathrm{t}} \in \{\mathcal{T}_{i'}^{\mathrm{t}}\}_{i'=0}^{i-1}$ contains both vertex
5: j and nonadjacent vertices of j **then**
6: $\mathcal{T}_i^{\mathrm{t}} = \mathcal{T}_m^{\mathrm{o}} - \mathcal{T}_j$ or $\mathcal{T}_i^{\mathrm{t}} = \mathcal{T}_{i'}^{\mathrm{t}} - \mathcal{T}_j$;
7: Remove all the nonadjacent vertices of j
8: from the corresponding $\mathcal{T}_m^{\mathrm{o}}$ or $\mathcal{T}_{i'}^{\mathrm{t}}$, set
9: $i = i + 1$;
10: **end if**
11: **end for**
12: $\mathcal{G}^{\mathrm{m}} = \mathcal{G}^{\mathrm{m}} \cup \mathcal{T}_m^{\mathrm{o}} \cup \{\mathcal{T}_{i'}^{\mathrm{t}}\}_{i'=0}^{i-1}$
13: **end for**

Fig. 3 *Algorithm 1: searching for maximal complete subgraphs*

$\mathcal{G}^{\mathrm{w}} = (\mathcal{H}, \mathcal{B}, \boldsymbol{w})$ can be constructed, where $\mathcal{H}$ denotes the vertex set, $\mathcal{B}$ denotes the edge set, and $\boldsymbol{w}$ denotes the weight vector corresponding to the vertices of $\mathcal{G}^{\mathrm{w}}$ whose elements are set to be the incremental offloaded traffic of their corresponding complete subgraphs, i.e. $[\boldsymbol{w}]_n = T_n^{\mathrm{i}}$. In $\mathcal{G}^{\mathrm{w}}$, two vertices are connected through an edge if their representing complete subgraphs have a certain identical vertex. It is known from the graph theory that an independent or stable set is a set of vertices in a graph, no two of which are adjacent [27]. Then, the independent subset of $\mathcal{H}$ certainly satisfies the constraint in (5). Correspondingly, the objective cluster sets of the optimisation problem in (29) can be readily obtained by searching for the equivalent max-weight independent subset of $\mathcal{H}$ of the corresponding weighted graph $\mathcal{G}^{\mathrm{w}}$. The max-weight independent subset of $\mathcal{H}$ can be obtained by solving a 0–1 integer programming problem, which will be presented in detail in Section 4.1.3.

Remark here that we map one cluster to one complete subgraph, which guarantees proper-sized clusters and avoids unnecessary intra-cluster signalling overhead, instead of one connected subgraph as in [26], which tends not to constrain the cluster size.

4.1.2 Searching for complete subgraphs: We propose to search for maximal complete subgraphs to find all the possible complete subgraphs. It is known from [27] that any complete subgraph must belong to a maximal complete subgraph and it is more difficult to find complete subgraphs through direct searching than through indirect searching for maximal complete subgraphs. We propose to exploit the adjacency table of each vertex in the node graph $\mathcal{G}^{\mathrm{n}}$ to search for maximal complete subgraphs. For $m \in \mathcal{M}$, let $\mathcal{T}_m = \{m\} \cup \{m' \mid m' \in \mathcal{M}, m' > m, (m', m) \in \mathcal{E}\}$ denote the adjacency table of vertex m of $\mathcal{G}^{\mathrm{n}}$, and L_m denotes the table size of $\mathcal{T}_m$. If $L_m = 1$ or $\mathcal{T}_m \subseteq \mathcal{T}_{m'}$ for $m, m' \in \mathcal{M}$ and $m' < m$, it is unnecessary to search for a maximal complete subgraph in $\mathcal{T}_m$. Remove all the unnecessary or redundant adjacency tables and sort the remaining in descending order denoted by $\mathcal{T}_m^{\mathrm{o}}$ according to their table sizes. Let $\mathcal{T}$ denote the set of the reordered adjacency tables of $\mathcal{G}^{\mathrm{n}}$. Remove any vertex that does not connect with all the other vertices in $\mathcal{T}_m^{\mathrm{o}}$. Then, the remaining vertices in $\mathcal{T}_m^{\mathrm{o}}$ form a maximal complete subgraph. Let $\mathcal{G}^{\mathrm{m}}$ denote the set of maximal complete subgraphs. The detailed description of our proposed algorithm of searching for maximal complete subgraphs is presented in Algorithm 1 (see Fig. 3). After maximal complete subgraphs are found, all the possible complete subgraphs can be readily obtained.

4.1.3 Searching for max-weight independent subset: According to the construction of the weighted graph $\mathcal{G}^{\mathrm{w}}$, two vertices in $\mathcal{H}$ are adjacent and there exists an edge between them if their representing candidate cluster sets have some identical elements. Let $\boldsymbol{x}$ denote the binary indicating vector for the vertices in $\mathcal{H}$ with $[\boldsymbol{x}]_n = 1$ if the candidate cluster set represented by the vertex h_n belongs to the objective disjoint cluster sets of the original optimisation problem in (29) and $[\boldsymbol{x}]_n = 0$ otherwise. If the vertices h_n and $h_{n'}$ can be connected through an edge $(h_n, h_{n'}) \in \mathcal{B}$, the relationship $[\boldsymbol{x}]_n[\boldsymbol{x}]_{n'} = 0$ should be satisfied.

According to the above description, the original optimisation problem in (29) can be transformed into the following 0–1 integer programming problem:

$$\max_{\boldsymbol{x}} \quad \boldsymbol{w}^T \boldsymbol{x} \tag{36}$$

$$\text{s.t.} \quad [\boldsymbol{x}]_n \in \{0, 1\}, \quad \forall h_n \in \mathcal{H}, \tag{36a}$$

$$[\boldsymbol{x}]_n[\boldsymbol{x}]_{n'} = 0, \quad \forall (h_n, h_{n'}) \in \mathcal{B}. \tag{36b}$$

The above optimisation problem can be solved by linear programming only if its linear relaxation is tight and has a unique integral solution. However, the above two conditions are hard to be satisfied [28]. Actually, the optimisation problem in (36) is a classical problem that maximises a submodular set function and can often be solved by greedy algorithms [29]. Considering that traditional greedy algorithms cannot take full advantage of the specific constraints in (36a) and (36b), we then propose a more effective greedy algorithm. Let $\mathcal{G}_n$ denote the independent subset of $\mathcal{H}$ and w_n denote the sum weight of all the vertices in $\mathcal{G}_n$. Each time move one vertex with the largest weight from $\mathcal{H}$ to $\mathcal{G}_n$ and remove its adjacent vertices from $\mathcal{H}$. Repeat the above step until $\mathcal{H}$ is empty. The independent subset $\mathcal{G}_n$ so obtained with the maximum sum weight w^{n} is just the max-weight independent subset denoted by $\mathcal{G}^{\mathrm{o}}$ that we are searching for. The detailed description of our proposed greedy algorithm of searching for the max-weight independent subset is presented in Algorithm 2 (see Fig. 4).

In traditional greedy algorithms [30, 31], the vertex with the largest weight is generally chosen as the initial vertex to search for the max-weight independent subset. In contrast, we set N' outer loops in Algorithm 2 (Fig. 4). Correspondingly, each vertex in $\mathcal{H}$ has a chance to be the initial vertex to constitute an independent subset. Therefore, the N' outer loops in Algorithm 2 (Fig. 4) guarantee to find the max-weight independent subset of the vertex set of the weighted graph.

To further illustrate the above issue, take a weighted graph with nine vertices as shown in Fig. 5 for example. In the weighted graph, the vertices are divided into three groups according to their weights, the weight of each vertex in the first group is larger than that in the second and third groups, and the weight of each vertex in the second group is larger than that in the third group. Assume vertex 1 has the largest weight among the nine vertices. In traditional greedy algorithms, vertex 1 will be chosen as the initial vertex. Then, the output max-weight independent subset will be

Input: $\mathcal{G}^{\mathrm{w}}$
Output: $\mathcal{G}^{\mathrm{o}}$
Initialize $w^{\mathrm{o}} = 0, \mathcal{G}^{\mathrm{o}} = \varnothing$;
for each $h_n \in \mathcal{H}$ **do**
  Initialize $\mathcal{H}' = \mathcal{H}, \mathcal{G}_n = \{h_n\}, w_n = [\boldsymbol{w}]_n$;
  Remove all the adjacent vertices of h_n from $\mathcal{H}'$;
  while $\mathcal{H}' \neq \varnothing$ **do**
    Find the vertex with the largest weight from $\mathcal{H}'$ denoted by $h_{n^{max}}$;
    $\mathcal{G}_n = \mathcal{G}_n \cup \{h_{n^{max}}\}, w_n = w_n + [\boldsymbol{w}]_{n^{max}}$;
    Remove all the adjacent vertices of $h_{n^{max}}$ from $\mathcal{H}'$;
  end while
  if $w^{\mathrm{o}} < w^{\mathrm{n}}$ **then**
    $w^{\mathrm{o}} = w_n, \mathcal{G}^{\mathrm{o}} = \mathcal{G}_n$.
  end if
end for

Fig. 4 *Algorithm 2: searching for max-weight independent subset*

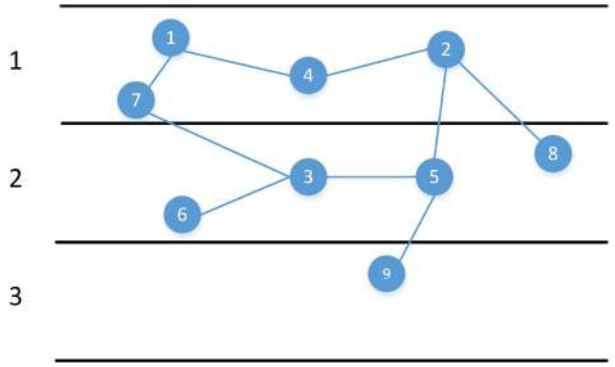

Fig. 5 *Illustration of a weighted graph with nine vertices*

$\{1, 2, 3\}$. However, in Algorithm 2 (Fig. 4), vertex 4 is also allowed to be the initial vertex. Then, the output max-weight independent subset will be $\{4, 5, 6, 7, 8\}$ if its sum weight is larger than that of $\{1, 2, 3\}$. It can be readily seen that the sum weight of all the vertices in the obtained independent subset will not be the maximum if the initial vertex is not selected properly. Therefore, the N' outer loops in Algorithm 2 (Fig. 4) can indeed guarantee to find the max-weight independent subset.

After the max-weight independent subset $\mathcal{G}^{\mathrm{o}}$ is found, the clustered sets and non-clustered set can be determined. Then, the set of popular files $\mathcal{F}_n^{\mathrm{c}}$ for $n \in \mathcal{N}$ in cluster n and the set of popular files $\mathcal{F}_m^{\mathrm{n}}$ for $m \in \mathcal{M}^{\mathrm{n}}$ at the non-clustered F-AP m can be determined according to (30) and (31), respectively.

4.2 Proposed graph-based content placement approach

4.2.1 Description of the proposed approach: In our proposed graph-based content placement approach, firstly, we find the complete subgraphs corresponding to the elements in the obtained max-weight independent subset $\mathcal{G}^{\mathrm{o}}$, and remove the edges in these complete subgraphs. Utilising the vertices and the remaining edges in the node graph $\mathcal{G}^{\mathrm{n}}$, we propose to construct a redundancy graph denoted by $\mathcal{G}^{\mathrm{r}} = (\mathcal{M}, \mathcal{E}^{\mathrm{r}})$, where $\mathcal{M}$ denotes its vertex set and $\mathcal{E}^{\mathrm{r}}$ denotes its edge set reflecting the cache redundancy among cooperative F-APs. Let $e = \{m, m'\} \in \mathcal{E}^{\mathrm{r}}$ denote the edge that connects vertex m and vertex m', and $\mathcal{F}_e^{\mathrm{d}}$ denote the set of duplicate popular files in the obtained popular file set of the cooperative F-APs corresponding to the vertices connected by edge e. If edge e connects $m \in \mathcal{M}_n^{\mathrm{c}}$ and $m' \in \mathcal{M}^{\mathrm{n}}$, then we have: $\mathcal{F}_e^{\mathrm{d}} = \mathcal{F}_n^{\mathrm{c}} \cap \mathcal{F}_{m'}^{\mathrm{n}}$. If edge e connects $m \in \mathcal{M}_n^{\mathrm{c}}$ and $m' \in \mathcal{M}_{n'}^{\mathrm{c}}$, then we have $\mathcal{F}_e^{\mathrm{d}} = \mathcal{F}_n^{\mathrm{c}} \cap \mathcal{F}_{n'}^{\mathrm{c}}$, whose size may exceed K, i.e. the storage size of each F-AP. Correspondingly, only a portion of the duplicate popular files in $\mathcal{F}_e^{\mathrm{d}}$ will indeed cause cache redundancy. Furthermore, when edge e connects $m \in \mathcal{M}_n^{\mathrm{c}}$ and edge e' connects $m' \in \mathcal{M}_n^{\mathrm{c}}$ with $e' \neq e$ and $m' \neq m$, $\mathcal{F}_e^{\mathrm{d}}$ and $\mathcal{F}_{e'}^{\mathrm{d}}$ may contain duplicate files. Correspondingly, only a portion of the duplicate popular files in $\mathcal{F}_e^{\mathrm{d}}$ and $\mathcal{F}_{e'}^{\mathrm{d}}$ will indeed cause cache redundancy. Therefore, we propose to separate $\mathcal{F}_e^{\mathrm{d}}$ to determine the duplicate files that will indeed cause cache redundancy at edge e. The process of separating $\mathcal{F}_e^{\mathrm{d}}$ will be presented in detail in Section 4.2.2.

Then, we propose to enhance the caching decisions to control the caching locations of the duplicate popular files and ensure that each duplicate popular file is cached only once between each pair of cooperative F-APs. After determining the caching locations for all the duplicate popular files, the remaining storage space of each F-AP is filled by the rest files according to their request probability. The process of caching-decision enhancement will be presented in detail in Section 4.2.3.

4.2.2 Separate the set of duplicate popular files: Let $\mathcal{T}_m$ denote the adjacency table of vertex m of $\mathcal{G}^{\mathrm{r}}$. Sort all the adjacency tables in descending order according to their table sizes. Let $\mathcal{T}$ denote the set of the reordered adjacency tables of $\mathcal{G}^{\mathrm{r}}$, and $\mathcal{T}_m^{\mathrm{o}}$ denote the mth adjacency table in $\mathcal{T}$. Let K_m denote the size of the remaining storage space of the F-AP corresponding to vertex m. Initialise $K_m = K$. Let $\mathcal{F}_m^{\mathrm{i}}$ denote the intersection of the sets of duplicate popular files at all the edges that connect vertex m and its adjacency vertices in $\mathcal{T}_m^{\mathrm{o}}$. Then, it can be expressed as follows:

$$\mathcal{F}_m^{\mathrm{i}} = \bigcap_{e = \{m, m'\}, m' \neq m, m' \in \mathcal{T}_m^{\mathrm{o}}} \mathcal{F}_e^{\mathrm{d}}, \quad m \in \mathcal{M}. \tag{37}$$

Let $\mathcal{F}_e^{\mathrm{r}}$ denote the set of files that will indeed cause cache redundancy at edge e after separating $\mathcal{F}_e^{\mathrm{d}}$. If $\left|\mathcal{F}_m^{\mathrm{i}}\right| \geq K_m$, $\mathcal{F}_e^{\mathrm{r}}$ will be constituted by the random K_m files in $\mathcal{F}_m^{\mathrm{i}}$. Otherwise, $\mathcal{F}_e^{\mathrm{r}}$ will be constituted by the random $(K_m - \left|\mathcal{F}_m^{\mathrm{i}}\right|)/(\left|\mathcal{T}_m^{\mathrm{o}}\right| - 1)$ files in $\mathcal{F}_e^{\mathrm{d}} \backslash \mathcal{F}_m^{\mathrm{i}}$ and all the files in $\mathcal{F}_m^{\mathrm{i}}$. Once $\mathcal{F}_e^{\mathrm{r}}$ is determined, update $K_{m'} = K_{m'} - \left|\mathcal{F}_e^{\mathrm{r}}\right|$ for vertex $m' \in \mathcal{T}_m^{\mathrm{o}}$ and update $\mathcal{F}_{e'}^{\mathrm{d}} = \mathcal{F}_{e'}^{\mathrm{d}} \backslash \mathcal{F}_e^{\mathrm{r}}$ if edge $e' \in \mathcal{E}^{\mathrm{r}}$ connects vertex m'.

4.2.3 Enhance the caching decisions: Let $\Delta x_{mf} \in \{-1, 0, 1\}$ denote the indicator of the caching-decision enhancement for file $f \in \mathcal{F}$ at vertex $m \in \mathcal{M}$, where $\Delta x_{mf} = 1$ indicates that the F-AP corresponding to vertex m is chosen as the caching location for file f, $\Delta x_{mf} = -1$ indicates that the F-AP corresponding to vertex m is not allowed to cache file f so as to eliminate redundancy, and $\Delta x_{mf} = 0$ indicates that the caching location for file f has not been determined yet. Initialise $\Delta x_{mf} = 0$ and set $K_m = K$.

Firstly, calculate the indicators of the caching-decision enhancements for file $f \in \mathcal{F}_e^{\mathrm{r}}$. For each $\mathcal{T}_m^{\mathrm{o}} \in \mathcal{T}$ and each $m' \in \mathcal{T}_m^{\mathrm{o}}$ with $m' \neq m$, find the files whose caching locations are at the F-AP corresponding to vertex m, and forbid these files to be cached at the F-AP corresponding to vertex m'. Then, the indicators of the corresponding caching-decision enhancements are set as follows:

$$\Delta x_{m'f} = -1, \quad f \in \left\{ f \,\middle|\, \Delta x_{mf} = 1 \right\} \cap \mathcal{F}_e^{\mathrm{r}}. \tag{38}$$

Update $\mathcal{F}_e^{\mathrm{r}}$ by removing these files. Furthermore, find the files whose caching locations are not allowed to be at the F-AP corresponding to vertex m, and choose the F-AP corresponding to vertex m' as the caching locations for these files. Then, the indicators of the corresponding caching-decision enhancements are set as follows:

$$\Delta x_{m'f} = 1, \quad f \in \left\{ f \,\middle|\, \Delta x_{mf} = -1 \right\} \cap \mathcal{F}_e^{\mathrm{r}}. \tag{39}$$

Input: $\mathcal{F}_m^{\mathrm{n}}, \mathcal{F}_n^{\mathrm{c}}, \mathcal{G}^{\mathrm{r}}$
Output: x_{mf}

Calculate $\mathcal{F}_e^{\mathrm{r}}$ for each edge e;
for each $\mathcal{T}_m^{\mathrm{o}} \in \mathcal{T}$ **do**
  for each $m' \in \mathcal{T}_m^{\mathrm{o}}$ with $m' \neq m$ **do**
    Determine Δx_{mf} and $\Delta x_{m'f}$ according to (38)-(41), and update the corresponding $\mathcal{F}_e^{\mathrm{r}}$, $\mathcal{F}_n^{\mathrm{c}}$, K_m, and K'_m;
  end for
end for
for each $n \in \mathcal{N}$ **do**
  for each $m \in \mathcal{M}_n^{\mathrm{c}}$ **do**
    Determine Δx_{mf} according to (42), and update the corresponding K_m and $\mathcal{F}_n^{\mathrm{c}}$;
  end for
  for each $m \in \mathcal{M}_n^{\mathrm{c}}$ **do**
    Determine Δx_{mf} according to (43), and update the corresponding K_m and $\mathcal{F}_n^{\mathrm{c}}$;
  end for
end for
for each $m \in \mathcal{M}^{\mathrm{n}}$ **do**
  Determine Δx_{mf} according to (44);
end for
for each $m \in \mathcal{M}$ **do**
  Set x_{mf} according to (45).
end for

Fig. 6 *Algorithm 3: graph-based content placement algorithm*

Update $K_{m'}$ and $\mathcal{F}_e^{\mathrm{r}}$ by removing these files. If F-AP $m' \in \mathcal{M}_n^{\mathrm{c}}$, update $\mathcal{F}_n^{\mathrm{c}}$ by removing these files. Let T_{em}^{p} denote the possible offloaded traffic due to caching the remaining files in $\mathcal{F}_e^{\mathrm{r}}$ at the F-AP corresponding to vertex m. Then, it can be expressed as follows:

$$T_{em}^{\mathrm{p}} = \sum_{m' \in m \cup \mathcal{S}_m} \sum_{f \in \mathcal{F}_e^{\mathrm{r}}} \lambda_{m'} p_{m'f} L. \tag{40}$$

Suppose $T_{em}^{\mathrm{p}} \geq T_{em'}^{\mathrm{p}}$. Then, set the indicators of the corresponding caching-decision enhancements as follows:

$$\Delta x_{mf} = 1, \quad \Delta x_{m'f} = -1, \quad f \in \mathcal{F}_e^{\mathrm{r}}. \tag{41}$$

Update $K_m = K_m - \left|\mathcal{F}_e^{\mathrm{r}}\right|$. If F-AP $m \in \mathcal{M}_n^{\mathrm{c}}$, update $\mathcal{F}_n^{\mathrm{c}}$ by removing these files.

Secondly, calculate the indicators of the corresponding caching-decision enhancements for the remaining files in $\mathcal{F}_n^{\mathrm{c}}$. For each $m \in \mathcal{M}_n^{\mathrm{c}}$, find the files whose caching locations can be at the F-AP corresponding to vertex m. Then, set the indicators of the corresponding caching-decision enhancements as follows:

$$\Delta x_{mf} = 1, \quad f \in \left\{ f \,\middle|\, \Delta x_{mf} = 0 \right\} \cap \mathcal{F}_n^{\mathrm{c}}. \tag{42}$$

Update K_m and $\mathcal{F}_n^{\mathrm{c}}$ by removing these files. For each $m \in \mathcal{M}_n^{\mathrm{c}}$, which satisfies $K_m > 0$, randomly select K_m files from $\mathcal{F}_n^{\mathrm{c}}$, and set the indicators of the corresponding caching-decision enhancements as follows:

$$\Delta x_{mf} = 1, \quad f \in \mathcal{F}_n^{\mathrm{c}}. \tag{43}$$

Update K_m and $\mathcal{F}_n^{\mathrm{c}}$ by removing these files.

Thirdly, calculate the indicators of the corresponding caching-decision enhancements at the vertices corresponding to non-clustered F-APs. For each $m \in \mathcal{M}^{\mathrm{n}}$ which satisfies $K_m > 0$, select K_m most popular files from the uncached files at the F-AP corresponding to vertex m and its co-operators according to their request probability, and set Δx_{mf} as follows:

$$\Delta x_{mf} = 1, \quad f \in \mathcal{F} \setminus \left\{ f \,\middle|\, \Delta x_{m'f} = 1, m' \in m \cup \mathcal{S}_m \right\}. \tag{44}$$

Finally, enhance the caching-decision for each $m \in \mathcal{M}$ as follows:

$$x_{mf} = \begin{cases} 1, & \Delta x_{mf} = 1, \\ 0, & \Delta x_{mf} \neq 1. \end{cases} \tag{45}$$

The detailed description of our proposed graph-based content placement algorithm is presented in Algorithm 3 (see Fig. 6).

4.3 Complexity analysis

Let $\bar{L}$ denote the average size of the adjacency tables of all the vertices in the node graph $\mathcal{G}^{\mathrm{n}}$. Then, the computational complexity of searching for maximal complete subgraphs in Algorithm 1 (Fig. 3) is $\mathcal{O}(M\bar{L})$. Furthermore, the computational complexity of obtaining all the complete subgraphs is $\mathcal{O}(P\bar{V})$, where P denotes the number of maximal complete subgraphs that have been found, and $\bar{V}$ denotes the average vertex number of all the complete subgraphs. Besides, the computational complexity of searching for the max-weight independent subset in Algorithm 2 (Fig. 4) is $\mathcal{O}(N'N)$. Therefore, the computational complexity of the proposed graph-based clustering approach is $\mathcal{O}(M\bar{L} + P\bar{V} + N'N)$.

Let δ denote the maximum degree of the redundancy graph. The computation complexity of the proposed graph-based content placement algorithm is $\mathcal{O}(M\delta + 2M)$. In summary, the computational complexity of the proposed graph-based cooperative caching scheme is $\mathcal{O}(M\bar{L} + P\bar{V} + N'N + M\delta + 2M)$. By considering $\bar{L} < M$, $\bar{V} < M$, $N < M$, and $\delta < M$, the computation complexity of the proposed graph-based cooperative caching scheme is $\mathcal{O}(M^2 + PM + N'M)$ for the worst case. It is obviously lower than that of $\mathcal{O}(M^3KF^2)$ in [8] and $\mathcal{O}(M^4K + MKF)$ in [22] by taking $M \ll F$, $P < F$, and $N' < F$ into account.

5 Simulation results

In this section, the performance of the proposed graph-based cooperative caching scheme is evaluated via simulations. In the simulations, the locations of the considered F-APs are randomly distributed, and the request probability at each F-AP is generated from the global request probability which follows Zipf distribution with the skewness parameter z. [Let p_f denote the global request probability of file f. Assume that the global request probability and the request probability at the considered M F-APs have the following relationship: $p_f = \sum_{m \in \mathcal{M}} w_m p_{mf}$ [24].] Unless otherwise stated, the system parameters are set as follows: $z = 0.6$, $M = 10$, $F = 5000$, $K = 250$, $L = 2$ Gb. We choose the random caching (RC) scheme [32], the globally popular caching (GPC) scheme [20], the exclusive most popular caching (ExMPC) scheme [33], and the locally popular caching (LPC) scheme [20], as four baselines. For the RC scheme, all files are picked from the content library randomly to cache into the F-APs, and neighbouring F-APs cannot cooperate with each other. For the GPC scheme, the K most popular files are cached at each F-AP based on the global request probability, and neighbouring F-APs cannot cooperative with each other. For the ExMPC scheme, the K most popular files are stored at each F-AP based on local content popularity, and the cloud server stores the F files based on global popularity, excluding the K most popular files which have been cached at each F-AP, and neighbouring F-APs cannot cooperate with each other. For the LPC scheme, the K most popular files are cached at each F-AP based on the local request probability, and neighbouring F-APs can cooperate with each other.

In Fig. 7, we show the offloaded traffic T of our proposed scheme and the two baselines versus the storage size K of each F-AP with different distance threshold γ^{d}. It can be observed that the offloaded traffic of all the three schemes increases with the storage size. It can also be observed that the performance of the proposed scheme is superior to that of the baselines. [Clearly, a centralised approach has been assumed in this study. This would certainly

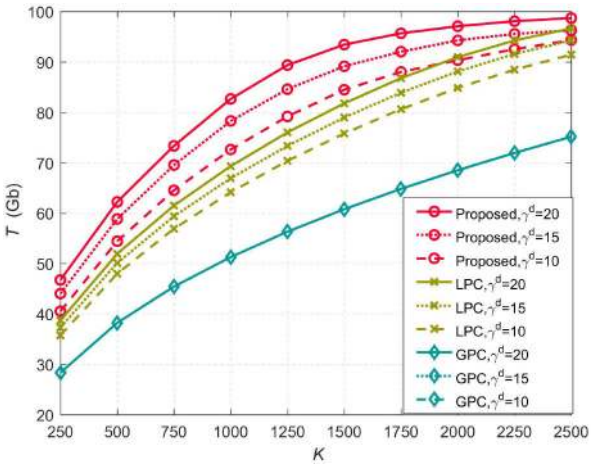

Fig. 7 *Offloaded traffic T versus the storage size K of each F-AP under different distance thresholds* γ^{d}

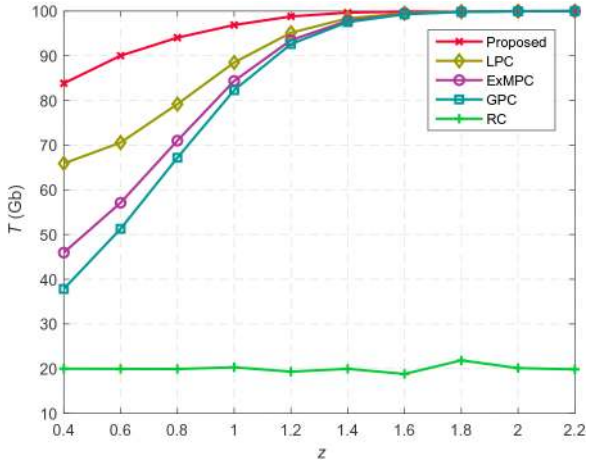

Fig. 8 *Offloaded traffic T versus the skewness parameter z of Zipf distribution with* $\gamma^{\mathrm{d}} = 20$ *and* $K = 1000$

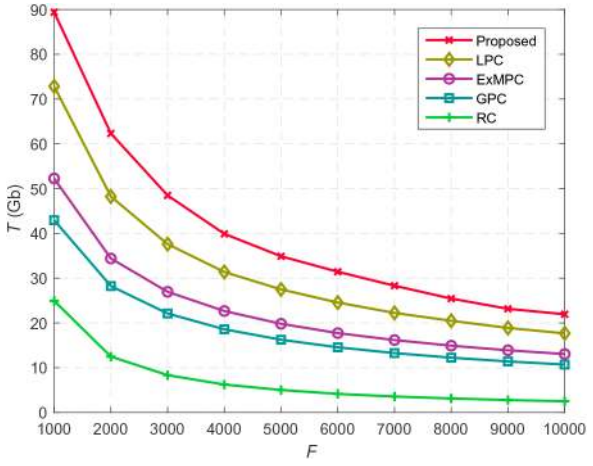

Fig. 9 *Offloaded traffic T versus the content library size F with* $\gamma^{\mathrm{d}} = 15$ *and* $z = 0.4$

incur the necessary signalling overhead, and the impact of such overhead will be an interesting issue for future research.] The reason is that the proposed scheme improves clustering and reduces the repetitive and redundant storage of files. Correspondingly, more user requests can be satisfied locally compared with the baselines. Furthermore, the offloaded traffic of the proposed and LPC schemes increases with distance threshold γ^{d}, and γ^{d} has a greater influence on the performance of the proposed scheme. The reason is that as γ^{d} becomes larger, the constraints of the clustering subproblem in our proposed scheme will be relaxed, the cluster size will become larger, more F-APs can cooperate with each other, and more files can then be successfully cached locally.

In Fig. 8, we show the offloaded traffic T of our proposed scheme and the four baselines versus the skewness parameter z of Zipf distribution with $\gamma^{\mathrm{d}} = 20$ and $K = 1000$. It can be observed that the offloaded traffic of all the four schemes except the RC scheme increases with z. The reason is that as z becomes larger, the most popular files will concentrate on fewer files and more traffic can then be offloaded. It can also be observed that the performance of the proposed scheme is superior to that of the baselines for all z.

In Fig. 9, we show the offloaded traffic T of our proposed scheme and the four baselines versus the content library size F with $\gamma^{\mathrm{d}} = 15$ and $z = 0.4$. It can be observed that the offloaded traffic of all the five schemes decreases with F. The reason is that as F becomes larger, the requested files will become more diverse and the number of requested files that are not cached locally will increase. It can also be observed that the performance of the proposed scheme is superior to that of the baselines for all F.

6 Conclusions

In this study, we have proposed a graph-based cooperative caching scheme including clustering and content placement in F-RANs. By constructing the relevant node graph and weighted graph, the objective cluster sets have been obtained by searching for the max-weight independent subset of the vertex set of the weighted graph. By constructing the redundancy graph, the final caching decisions have been obtained by calculating the indicators of the caching-decision enhancements. Both significant computational complexity reduction and remarkable offloaded traffic have been achieved by using our proposed graph-based cooperative caching scheme.

7 Acknowledgments

This work was supported in part by the Natural Science Foundation of China under grant no. 61971129, the Natural Science Foundation of Jiangsu Province under grant no. BK20181264, the Research Fund of the State Key Laboratory of Integrated Services Networks (Xidian University) under grant no. ISN19-10, the Research Fund of the Key Laboratory of Wireless Sensor Network and Communication (Shanghai Institute of Microsystem and Information Technology, Chinese Academy of Sciences) under grant no. 2017002, the National Basic Research Program of China (973 Program) under grant no. 2012CB316004, and the U.K. Engineering and Physical Sciences Research Council under grant no. EP/K040685/2.

8 References

[1] Peng, M., Yan, S., Zhang, K., *et al.*: 'Fog-computing-based radio access networks: issues and challenges', *IEEE Netw.*, 2016, **30**, (4), pp. 46–53

[2] Jiang, W., Feng, G., Qin, S.: 'Optimal cooperative content caching and delivery policy for heterogeneous cellular networks', *IEEE Trans. Mob. Comput.*, 2017, **16**, (5), pp. 1382–1393

[3] Li, X., Wang, X., Leung, V.C.M.: 'Weighted network traffic offloading in cache-enabled heterogeneous networks'. Proc. IEEE Int. Conf. on Communications (ICC), Kuala Lumpur, Malaysia, 2016, pp. 1–6

[4] Zhang, S., He, P., Suto, K., *et al.*: 'Cooperative edge caching in user-centric clustered mobile networks', *IEEE Trans. Mob. Comput.*, 2018, **17**, (8), pp. 1791–1805

[5] Chae, S., Quek, T., Choi, W.: 'Content placement for wireless cooperative caching helpers a tradeoff between cooperative gain and content diversity gain', *IEEE Trans. Wirel. Commun.*, 2017, **16**, (10), pp. 6795–6807

[6] Poularakis, K., Iosifidis, G., Argyriou, A., *et al.*: 'Caching and operator cooperation policies for layered video content delivery'. IEEE INFOCOM 2016 – 35th Annual IEEE Int. Conf. on Computer Communications, San Francisco, CA, USA, 2016, pp. 1–9

[7] Sun, Y., Chen, Z., Liu, H.: 'Delay analysis and optimization in cache-enabled multi-cell cooperative networks'. Proc. IEEE Global Communications Conf. (GLOBECOM), Washington DC, USA, 2016, pp. 1–7

[8] Shanmugam, K., Golrezaei, N., Dimakis, A., *et al.*: 'Femtocaching: wireless content delivery through distributed caching helpers', *IEEE Trans. Inf. Theory*, 2013, **59**, (12), pp. 8402–8413

[9] Zheng, G., Suraweera, H., Krikidis, I.: 'Optimization of hybrid cache placement for collaborative relaying', *IEEE Commun. Lett.*, 2016, **21**, (2), pp. 442–445

[10] ElBamby, M.S., Bennis, M., Saad, W., *et al.*: 'Content-aware user clustering and caching in wireless small cell networks'. Proc. IEEE Int. Symp. on Wireless Communication and Systems, Barcelona, Spain, 2014, pp. 945–949

[11] Huang, W., Song, T., Yang, Y., *et al.*: 'Cluster-based cooperative caching with mobility prediction in vehicular named data networking', *IEEE Access*, 2019, **7**, pp. 23442–23458

[12] Chen, Q., Wang, W., Wang, Y., *et al.*: 'Content caching clustering based on piecewise interest similarity'. 2017 IEEE Global Communications Conf., Singapore, 2017, pp. 1–6

[13] Han, W., Liu, A., Lau, V.K.N.: 'Dual-mode user-centric open-loop cooperative caching for backhaul-limited small-cell wireless networks', *IEEE Trans. Wirel. Commun.*, 2019, **18**, (1), pp. 532–545

[14] Jiang, W., Feng, G., Qin, S., *et al.*: 'Multi-agent reinforcement learning based cooperative content caching for mobile edge networks', *IEEE Access*, 2019, **7**, pp. 61856–61867

[15] Yang, Z., Liu, Y., Chen, Y: 'Q-learning for content placement in wireless cooperative caching'. 2018 IEEE Global Communications Conf., Abu Dhabi, UAE, 2018, pp. 1–6

[16] Müller, S., Atan, O., Schaar, M., *et al.*: 'Context-aware proactive content caching with service differentiation in wireless networks', *IEEE Trans. Wirel. Commun.*, 2017, **16**, (2), pp. 1024–1036

[17] Jiang, Y., Ma, M., Bennis, M., *et al.*: 'User preference learning based edge caching for fog radio access network', *IEEE Trans. Commun.*, 2019, **67**, (2), pp. 1268–1283

[18] Tamoor-ul-Hassan, S., Samarakoon, S., Bennis, M., *et al.*: 'Learning-based caching in cloud-aided wireless networks', *IEEE Commun. Lett.*, 2018, **22**, (1), pp. 137–140

[19] Liu, J., Yan, H., Li, Y., *et al.*: 'Cache behavior characterization and validation over large-scale video data', *IEEE Trans. Circuits Syst. Video Technol.*, 2018, **28**, (3), pp. 734–745

[20] Liu, J., Bai, B., Zhang, J., *et al.*: 'Cache placement in Fog-RANs: from centralized to distributed algorithms', *IEEE Trans. Wirel. Commun.*, 2017, **16**, (11), pp. 7039–7051

[21] Ashraf, M.I., Bennis, M., Saad, W.: 'Dynamic clustering and user association in wireless small cell networks with social considerations', *IEEE Trans. Veh. Technol.*, 2017, **66**, (7), pp. 6553–6568

[22] Wang, R., Zhang, J., Song, S., *et al.*: 'Mobility-aware caching in D2D networks', *IEEE Trans. Wirel. Commun.*, 2017, **16**, (8), pp. 5001–5015

[23] Chen, Z., Lee, J., Quek, T., *et al.*: 'Cooperative caching and transmission design in cluster-centric small cell networks', *IEEE Trans. Wirel. Commun.*, 2017, **16**, (5), pp. 3401–3415

[24] Li, X., Wang, X., Zhu, C.: 'Caching-as-a-service: virtual caching framework in the cloud-based mobile networks'. 2015 IEEE Conf. on Computer Communications Workshops (INFOCOM WKSHPS), Hong Kong, China, 2015, pp. 372–377

[25] Cui, X., Jiang, Y., Chen, X., *et al.*: 'Graph-based cooperative caching in Fog-RAN'. 2018 Int. Conf. on Computing, Networking and Communications (ICNC), Maui, Hawaii, USA, 2018, pp. 166–171

[26] Zhou, L., Hu, X., Ngai, E.C.H., *et al.*: 'A dynamic graph-based scheduling and interference coordination approach in heterogeneous cellular networks', *IEEE Trans. Veh. Technol.*, 2015, **65**, (5), pp. 3735–3748

[27] Bondy, J.A., Murty, U.S.R.: '*Graph theory with applications*' (Elsevier, New York, 1976)

[28] Sanghavi, S., Shah, D., Willsky, A.S.: 'Message passing for maximum weight independent set', *IEEE Trans. Inf. Theory*, 2009, **55**, (11), pp. 4822–4834

[29] Vondrák, J.: 'Submodularity and curvature: the optimal algorithm', *RIMS Kôkyûroku Bessatsu*, 2010, **B23**, pp. 253–266

[30] Hoepman, J.: 'Simple distributed weighted matchings'. Available at http://arxivorg/abs/cs/0410047, 2004

[31] Basagni, S.: 'Finding a maximal weighted independent set in wireless networks', *Telecommun. Syst.*, 2001, **18**, pp. 155–168

[32] Chen, M., Hao, Y., Hu, L., *et al.*: 'Green and mobility-aware caching in 5G networks', *IEEE Trans. Wirel. Commun.*, 2017, **16**, (12), pp. 8347–8361

[33] Tran, T.X., Le, D.V., Yue, G., *et al.*: 'Cooperative hierarchical caching and request scheduling in a cloud radio access network', *IEEE Trans. Mob. Comput.*, 2018, **17**, (12), pp. 2729–2743