

Cost-Optimal Caching for D2D Networks with User Mobility: Modeling, Analysis, and Computational Approaches

Tao Deng, *Student Member, IEEE*, Ghafour Ahani, Pingzhi Fan, *Fellow, IEEE*,
and Di Yuan, *Senior Member, IEEE*

Abstract

Caching popular files at user equipments (UEs) provides an effective way to alleviate the burden of the backhaul networks. Generally, popularity-based caching is not a system-wide optimal strategy, especially for user mobility scenarios. Motivated by this observation, we consider optimal caching with presence of mobility. A cost-optimal caching problem (COCP) for device-to-device (D2D) networks is modelled, in which the impact of user mobility, cache size, and total number of encoded segments are all accounted for. Compared with the related studies, our investigation guarantees that the collected segments are non-overlapping, takes into account the cost of downloading from the network, and provides a rigorous problem complexity analysis. The hardness of the problem is proved via a reduction from the satisfiability problem. Next, a lower-bounding function of the objective function is derived. By the function, an approximation of COCP (ACOCP) achieving linearization is obtained, which features two advantages. First, the ACOCP approach can use an off-the-shelf integer linear programming algorithm to obtain the global optimal solution, and it can effectively deliver solutions for small-scale and medium-scale system scenarios. Second, and more importantly, based on the ACOCP approach, one can derive the lower bound of global optimum of COCP, thus enabling performance benchmarking of any sub-optimal algorithm. To tackle large scenarios with low complexity, we first prove that the optimal caching placement of one user, giving other users' caching placements, can be derived in polynomial time. Then, based on this proof, a mobility aware user-by-user (MAUU) algorithm is developed. Simulation results verify the effectivenesses of the two approaches by comparing them to the lower bound of global optimum and conventional caching algorithms.

The paper is a significant extension of a previous work submitted to IEEE Globecom [1].

T. Deng and P. Fan are with the School of Information Science and Technology, Southwest Jiaotong University, Chengdu, Sichuan 610031, China (e-mail: dengtaoswjtu@foxmail.com and p.fan@ieee.org).

G. Ahani and D. Yuan are with the Department of Information Technology, Uppsala University, 751 05 Uppsala, Sweden (e-mail: {ghafour.ahani, di.yuan}@it.uu.se)

Index Terms

Backhaul networks, caching, D2D, integer linear programming, user mobility.

I. INTRODUCTION

A. Motivations

With rapid emergence of new services and application scenarios, such as social networks (e.g., Twitter and Facebook), multimedia contents (e.g., YouTube), and Internet of things (IoT) etc., explosive growth in mobile data traffic and massive device connectivity are becoming two main challenges for existing cellular networks. Hyper-dense small cell networks have been recognized as a promising technology to achieve higher network capacity in fifth-generation (5G) wireless networks [2], [3]. However, due to a large number of connections between the base stations (BSs) and core network (CN), the backhaul networks will face a heavy burden [4], calling for research from both the academia and industry. Caching is a promising technology to alleviate the burden of the backhaul networks by storing the required files or contents in advance at the edge devices [5]–[7], e.g., small cells and user equipments (UEs).

With caching, users can obtain their requested files from the edge devices so as to improve the network performance in terms of energy efficiency and file downloaded delay, and at the same time reduce the burden of backhaul [8]. The caching performance depends heavily on the cache placement strategy. Although the conventional strategy of caching popular files can improve the probability that the users will find the files of interest in their local caches, it is not a system-wide optimal solution, especially for user mobility scenarios. Therefore, it is necessary to revisit the caching problem with user mobility and investigate the following questions:

- How to make the best use of user mobility to design approaches for optimizing content caching?
- How much will mobility help?

To address the two questions, we consider caching at mobile users and investigate cost-optimal caching for device-to-device (D2D) networks. More specifically, the inter-contact model is used to describe the mobility pattern of mobile users. The mobile users can collect segments of files when they meet each other. If the total number of collected data segments is not enough to recover the requested content within a given period, the user has to download additional segments from the network.

B. Existing Studies

A number of studies have investigated caching placement optimization. The existing studies can be categorized into two groups.

The investigations in [9]–[15] considered caching at small cells. The works in [9]–[13] jointly considered the caching and multicast technologies to optimize system performance. In [9], the work investigated a multicast-aware caching problem. The hardness of this problem was proved, and an algorithm with approximation ratio was proposed. In [10], the study developed a random caching design with multicasting in a large-scale cache-enabled wireless network. An iterative algorithm was proposed to derive a local optimal solution. In order to reduce the computation complexity, an asymptotical optimal design was obtained. Based on [10], [11] further investigated caching and multicast design with backhaul constraints in heterogeneous networks (HetNets). In [12], the work considered a scenario with content-centric BS clustering and multicast beamforming. The authors target optimizing the weighted sum of backhaul cost and transmit power. In [13], a stochastic content multicast problem, originated from a Markov decision process, was formulated and a low-complexity algorithm was proposed. An assumption in [12] and [13] is that the content placement was given. Relaxing the assumption, [14] optimized the caching placement and proposed a mesh adaptive direct search algorithm.

Compared with caching at the small cells, the investigations in [16]–[22] considered caching at the UEs, e.g., D2D caching networks. The studies in [16]–[19] analyzed and investigated caching problems by using stochastic geometry tools. In [16], the study investigated the optimal caching placements to maximize the average successful receptions' density. In [17], the performance between caching at the small cells and UEs were analyzed and compared. Numerical results manifested that the performance varies by the user density and the content popularity distribution. In [18] and [19], the works investigated optimization problems with respect to probabilistic caching placement and average caching failure probability for each content. In [20], the study addressed a two-tier caching network in which a subset of UEs and small cells have cache capability. In [21] and [22], the authors proposed an accurate simulation model taking into account a holistic system design and investigated information theoretic bounds for D2D caching networks, respectively.

Although the above studies focused on the cache placement design to optimize network performance, they neglected the impact of user mobility on caching performance. This issue was

recognized in [23]. In [24] and [25], the authors investigated the caching placement problem taking into account user mobility in HetNets, with the objective of minimizing the probability that the macrocell has to serve a request. The intractability of this problem was proved, and the problem is then reformulated using mixed integer programming (MIP). Moreover, the authors derived an upper bound for the objective function and proposed a distributed algorithm. In [26] and [27], the studies investigated a mobility and popularity-based caching strategy (MPCS) and a seamless radio access network cache handover framework based on a mobility prediction algorithm (MPA), respectively. In [28], assuming that the trajectories of mobile users are known in advance, the authors investigated mobility-aware content caching and proposed an algorithm with approximation ratio. In [29], the work optimized caching placement to maximize the data offloading ratio. A dynamic programming algorithm was proposed to obtain the optimal solution in small-scale scenarios. Since the algorithm complexity increases exponentially, the authors first proved that the objective function is a monotone submodular function, and then proposed a greedy algorithm which can achieve an $1/2$ approximation.

The investigations in [25] and [29] are the most related works to our study. However, the system setup in [25] addresses caching at base stations, which is different from our study where we investigate caching at mobile users with mobility. In comparison to [29], our problem formulation takes into account the cost of downloading from network and guarantees that the collected segments are non-overlapping, along with giving a rigorous problem complexity analysis. In addition, our computational approach provides performance benchmarking of any sub-optimal algorithm for up to medium-size system scenarios.

C. Our Contributions

We investigate the cost-optimal caching problem with user mobility for D2D networks. Our objective is to optimize caching placement so as to minimize the expected cost of obtaining files of interest by collecting file segments. The main contributions are summarized as follows. First, a cost-optimal caching problem (COCP) is modelled, taking into account the impact of user mobility, cache size, and the total number of encoded segments. Accounting for this number is important in order to ensure no duplicates in the collected segments. Second, the hardness of the problem is proved. To the best of our knowledge, this is the first mathematical proof for the complexity of this type of problems. The proof is based on a reduction from the 3-satisfiability (3-SAT) problem [30]. Moreover, for problem-solving, due to the nonlinearity

and high complexity of the objective function in COCP, a linear lower-bounding function is derived, yielding an approximation of COCP (ACOCP). The ACOCP approach brings two advantages. On one hand, it enables the global optimal solution by using an off-the-shelf integer linear programming algorithm that can deliver solutions for small-scale and medium-scale system scenarios effectively. Second, and more importantly, it serves the purpose of performance benchmarking of any sub-optimal algorithm. To be specific, by this approach, the lower bound of global optimum of COCP can be obtained. We are hence able to gauge the deviation from optimum for any sub-optimal algorithm, whereas pure heuristics algorithm cannot be used for such a purpose. To tackle large-scale scenarios, it is proved that the optimal caching placement of one user, giving other users' caching placements, can be derived in polynomial time. Then, based on this proof, a mobility aware user-by-user (MAUU) algorithm is developed. Finally, Simulations are conducted to verify the effectivenesses of the ACOCP approach and the MAUU algorithm by comparing them to the lower bound of global optimum and conventional caching algorithms. Simulation results manifest that solving ACOCP leads to an effective approximation scheme – the solution of ACOCP does not deviate more than 4.4% from the global optimum of COCP. The true performance figure is likely to be better because the performance evaluation is derived using the lower bounds. For the MAUU algorithm, the gap value to global optimum is less than 9%. Thus, the algorithm achieves excellent balance between complexity and accuracy. In addition, the proposed algorithms also significantly outperform conventional caching algorithms, especially for large-scale scenarios.

The remainder of this paper is organized as follows. Section II introduces the system scenario, assumptions for caching placement, and cost model. Section III first derives the problem formulation, and then provides a rigorous complexity analysis. Section IV presents the lower bound approximation approach of COCP. Section V develops an fast yet effective mobility aware user-by-user algorithm. Performance evaluation is presented in Section VI. Finally, Section VII concludes this paper.

II. SYSTEM MODEL

A. System Scenario

There are a total of U mobile users in a network, whose index set is represented by $\mathcal{U} = \{1, 2, \dots, U\}$. Each user $i, i \in \mathcal{U}$, is equipped with a cache of size C_i . Fig. 1 shows the system

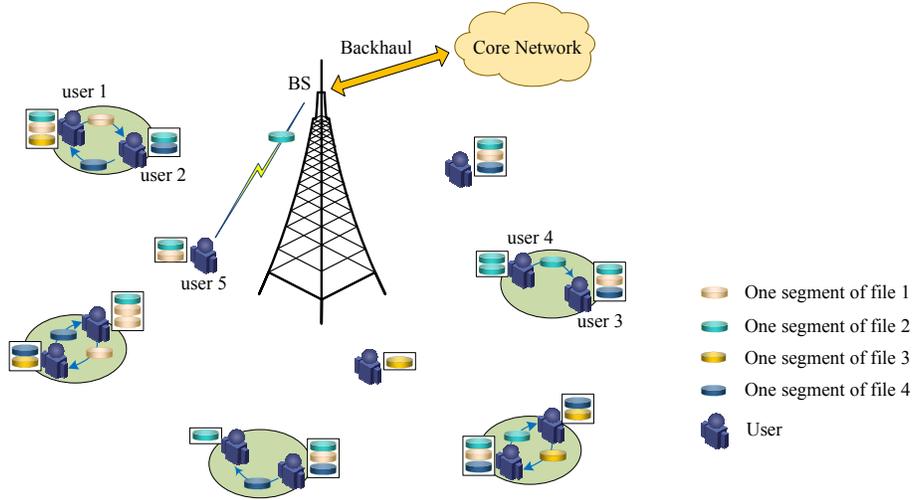


Figure 1. System scenario.

scenario in which mobile users are able to collect the content when they meet each other, e.g., user 1 and user 2.

The inter-contact model has been widely used to describe the mobility pattern of mobile users [31], [32]. In this model, the mobile users can communicate with each other when they meet. The contact process between any two mobile users is characterized by points along a timeline. Each point represents a time that the two users meet, and the inter-contact time represents the time between two consecutive points. The inter-contact time for any two users follows an exponential distribution. Moreover, it is assumed that the processes for the user pairs are independent. Hereafter, the term contact is used to refer to the event that two users meet each other.

B. Caching Placement

There are a total of F files, whose index set is represented by $\mathcal{F} = \{1, 2, \dots, F\}$. Each file f , $f \in \mathcal{F}$, is encoded into S_{\max}^f segments through a coding technique [25], [33]. File f can be recovered by collecting at least S_{rec}^f distinct segments. To describe the caching solution, we define a caching placement vector \mathbf{x} :

$$\mathbf{x} = \{x_{fi} \in \mathbb{N}, f \in \mathcal{F}, i \in \mathcal{U}\},$$

where x_{fi} represents the number of segments of file f stored at the user i . Denote by P_{fi} the probability that user i requests file f , with $\sum_{f=1}^F P_{fi} = 1$. When user i requests file f , it will collect the segments of the file from its own cache and from the encountered users through D2D communications. The latter is subject to a time period T_D . For example, in Fig. 1, user 1 will collect one segment of file 4 from user 2. At the same time, user 2 will collect one segment of file 1 from user 1. But user 4 cannot collect the content of file 3 from user 3, because the latter does not store any segment of file 3.

Each user will check the total number of collected segments of the requested file at the end of T_D . If the total number of collected segments of file f is at least S_{rec}^f , user i can recover this file. Otherwise, user i will have to download additional segments from the network in order to reach S_{rec}^f segments, e.g., user 5 in the figure. The file recovering process considers only segments that are distinct from each other in the cache. For example, user 4 stores two distinct segments of file 2.

C. Cost Model

Up to B segments can be collected by each user when two users meet. Denote by M_{ij} the number of contacts for users i and j . Here, M_{ij} follows a Poisson distribution with mean $\lambda_{ij}T_D$, where λ_{ij} represents the average number of contacts per unit time. The number of segments of file f collected by user i from user j within T_D , denoted by S_{fij} , is $\min(BM_{ij}, x_{fj})$. The number of segments of file f collected by user i from itself and all the other users via contacts within T_D , denoted by S_{fi} , is given as

$$S_{fi} = \sum_{j \in \mathcal{U}, j \neq i} \min(BM_{ij}, x_{fj}) + x_{fi}.$$

If $S_{fi} < S_{\text{rec}}^f$, user i will download $S_{\text{rec}}^f - S_{fi}$ segments from the network. This entity for file f and user i , denoted by S_{fi}^N , is thus $\max(S_{\text{rec}}^f - S_{fi}, 0)$. Denote by δ_D and δ_N the costs of obtaining one segment from a user and the network, respectively. The cost for user i to recover file f , denoted by Δ_{fi} , is $(S_{fi} - x_{fi})\delta_D + S_{fi}^N\delta_N$. Taking into account the distribution of file request probabilities, the cost for user i to recover its requested files, denoted by Δ_i , is $\sum_{f \in \mathcal{F}} P_{fi}\Delta_{fi}$. Thus, the expected average cost per user can be expressed as

$$\begin{aligned} \Delta &= \mathbb{E}\left\{\frac{1}{U} \sum_{i \in \mathcal{U}} \Delta_i\right\} \\ &= \mathbb{E}\left\{\frac{1}{U} \sum_{i \in \mathcal{U}} \sum_{f \in \mathcal{F}} P_{fi} [(S_{fi} - x_{fi})\delta_D + \max(S_{\text{rec}}^f - S_{fi}, 0)\delta_N]\right\}. \end{aligned}$$

III. PROBLEM FORMULATION AND COMPLEXITY ANALYSIS

A. Problem Formulation

Our problem is to minimize Δ by optimizing \mathbf{x} . Thus, the cost-optimal caching problem (COCP) can be formulated as

$$\min_{\mathbf{x}} \quad \mathbb{E}\left\{\frac{1}{U} \sum_{i \in \mathcal{U}} \sum_{f \in \mathcal{F}} P_{fi} [(S_{fi} - x_{fi})\delta_{\mathbb{D}} + \max(S_{\text{rec}}^f - S_{fi}, 0)\delta_{\mathbb{N}}]\right\} \quad (1a)$$

$$\text{s.t.} \quad \sum_{f \in \mathcal{F}} x_{fi} \leq C_i, \quad i \in \mathcal{U} \quad (1b)$$

$$\sum_{i \in \mathcal{U}} x_{fi} \leq S_{\text{max}}^f, \quad f \in \mathcal{F} \quad (1c)$$

$$x_{fi} \in \mathbb{N}, \quad i \in \mathcal{U}, \quad f \in \mathcal{F} \quad (1d)$$

Eq. (1b) requires the total number of cached segments to adhere to cache capacity limit. By Eq. (1c), the total number of segments of a file, cached by all users, does not exceed the number of encoded segments. This constraint guarantees that the collected segments of any file will be distinct from each other.

B. Complexity Analysis

Theorem 1. COCP is \mathcal{NP} -hard.

Proof: We adopt a polynomial-time reduction from the 3-satisfiability (3-SAT) problem that is \mathcal{NP} -complete. Consider any 3-SAT instance with m Boolean variables z_1, z_2, \dots, z_m , and n clauses. A variable or its negation is called a literal. Denote by \hat{z}_i the negation of z_i , $i = 1, 2, \dots, m$. Each clause consists of a disjunction of exactly three different literals, e.g., $\hat{z}_1 \vee z_2 \vee z_3$. The 3-SAT problem amounts to determining whether or not there exists an assignment of true/false values to the variables, such that all clauses are satisfied (i.e., at least one literal has value true in every clause). It is assumed that no clause contains both a variable and its negation; such clauses become always satisfied, thus they can be eliminated by preprocessing. Moreover, a literal appears in at least one clause as otherwise the corresponding value assignment is trivial. For the same reason, a literal is present in at most $n - 1$ clauses.

We construct a reduction from the 3-SAT instance as follows. The number of users is $U = 2m + n$, referred to as literal and clause users, respectively, i.e., $\mathcal{U} = \{1, 2, \dots, 2m + n\}$. There

are two files a and b , i.e., $\mathcal{F} = \{a, b\}$, each of them has m segments, i.e., $S_{\max}^a = S_{\max}^b = m$. File a or b can be recovered by collecting one segment, i.e., $S_{\text{rec}}^a = S_{\text{rec}}^b = 1$. The cache size of literal and clause users are one ($C_i = 1, i = 1, 2, \dots, 2m$) and zero ($C_j = 0, j = 2m + 1, \dots, 2m + n$), respectively.

The literal users are formed into m pairs. Denote by ϵ a small positive number. We set $\delta_N > 3n + \frac{n\epsilon(m-3)}{(1-\epsilon)^{m-2}}\delta_D$, and $\lambda_{ij} = \ln(\frac{1}{\epsilon})$ for users i and j in each of the m pairs. Then these users meet at least once with probability $1 - \epsilon$. We set $\lambda_{ij} = \ln \frac{1}{1-\epsilon}$ for the other literal users where i and j are from different pairs, so that these users meet at least once with probability ϵ . Each literal user is interested in downloading both files a and b with equal probability, i.e., $P_{ai} = P_{bi} = 1/2, i = 1, \dots, 2m$. First, suppose one of the users in each pair caches file a , and the other caches file b , or vice versa. It means that for any pair, the caching content is either ab or ba . This corresponds to the Boolean value assignment in the original 3-SAT instance. In such a case, the expected cost that both users of a pair recover both files a and b , denoted by Δ_1 , is given as

$$\Delta_1 = (1 - \epsilon)\delta_D + 2\epsilon(m - 1)\delta_D + \epsilon(1 - \epsilon)^{m-1}\delta_N.$$

Consequently, the total cost for all the literal users, denoted by Δ_1^l , is $m\Delta_1$.

Each clause user is interested in downloading file a with probability one, i.e., $P_{ai} = 1, i = 2m + 1, \dots, 2m + n$. If users i and j are clause users, λ_{ij} can be anything as their all have a cache size of zero. For a clause user i , if j is one of the three literal users in the corresponding clause in the 3-SAT instance, we set $\lambda_{ij} = \ln(\frac{1}{\epsilon})$. Otherwise, we set $\lambda_{ij} = \ln(\frac{1}{1-\epsilon})$. If at least one of the three literal users caches file a , then the expected cost for a clause user is at most $3(1 - \epsilon)\delta_D + \epsilon(m - 3)\delta_D + \epsilon^3(1 - \epsilon)^{m-3}\delta_N$. The corresponding values for the n clause users together, denoted by Δ_1^c , is $n(3(1 - \epsilon)\delta_D + \epsilon(m - 3)\delta_D + \epsilon^3(1 - \epsilon)^{m-3}\delta_N)$.

By the construction above, which is polynomial, the cost is no more than $\Delta_1^l + \Delta_1^c$ if the 3-SAT instance is satisfiable. Otherwise, at least one clause user has virtually no other option, than downloading from the network and the expected total cost is at least $m\Delta_1 + (n - 1)\Delta' + \epsilon(m - 3)\delta_D + (1 - \epsilon)^{m-3}\delta_N > \Delta_1^l + \Delta_1^c$, where $\Delta' = (1 - \epsilon)\delta_D + \epsilon(m - 3)\delta_D + \epsilon(1 - \epsilon)^{m-3}\delta_N$. Thus, whether or not there exists a caching placement strategy with a total expected cost of no more than $\Delta_1^l + \Delta_1^c$ gives the correct answer to 3-SAT.

Now, let's consider the case where some of the literal user pairs cache the same file. If there is one pair caching file a , i.e., the caching content is aa , another pair cache bb , and the remaining

pairs cache either ab or ba . The total literal users' cost, denoted by Δ_2^l , is given as

$$\Delta_2^l = 2((1 - \epsilon)\delta_D + 2\epsilon\delta_D + 2(m - 2)\epsilon\delta_D + (1 - \epsilon)^m\delta_N) + (m - 2)\Delta_1.$$

If all the clause users can obtain file a from the literal users, the total clause users cost, denoted by Δ_2^c , is no less than $n\Delta'$. The corresponding values for all the users together is $\Delta_2^l + \Delta_2^c$, and $\Delta_2^l + \Delta_2^c > \Delta_1^l + \Delta_1^c$. If there is more than one pair caching the same file, e.g., two pairs cache aa , the cost becomes even higher. Thus, the previous conclusion remains valid, namely whether or not there is an assignment with no more than $\Delta_1^l + \Delta_1^c$ gives the right answer even this case included.

Therefore, the recognition versions of COCP is \mathcal{NP} -complete and its optimization version is \mathcal{NP} -hard. \square

IV. LOWER BOUND APPROXIMATION APPROACH

Due to the COCP's intractability, generally it is difficult to obtain the global optimal solution. For problem-solving, we linearize the first part of objective function and derive a lower bound for the second part. These together give us a linear lower-bounding function, as an approximation to the original function. As a result, the problem can be reformulated as a mixed linear integer program.

Define

$$\Delta^{lb} \triangleq \frac{1}{U} \sum_{i \in \mathcal{U}} \sum_{f \in \mathcal{F}} P_{fi} [\Delta_{fi}^d + \max(\Delta_{fi}^n, 0)], \quad (2)$$

and

$$\begin{cases} \Delta_{fi}^d = \mathbb{E} \left(\sum_{j \in \mathcal{U}, j \neq i} \min(BM_{ij}, x_{fj}) \right) \delta_D, \\ \Delta_{fi}^n = S_{\text{rec}}^f \delta_N - \mathbb{E} \left[\sum_{j \in \mathcal{U}, j \neq i} \min(BM_{ij}, x_{fj}) + x_{fi} \right] \delta_N. \end{cases} \quad (3)$$

Theorem 2. Δ^{lb} is a lower-bounding function of Δ , i.e.,

$$\Delta \geq \Delta^{lb}.$$

Proof: See Appendix A. \square

Using Δ^{lb} , an approximation of COCP (ACOCP) can be formulated as

$$\min_{\mathbf{x}} \quad \frac{1}{U} \sum_{i \in \mathcal{U}} \sum_{f \in \mathcal{F}} P_{fi} [\Delta_{fi}^d + \max(\Delta_{fi}^n, 0)] \quad (4a)$$

$$\text{s.t.} \quad \sum_{f \in \mathcal{F}} x_{fi} \leq C_i, \quad i \in \mathcal{U} \quad (4b)$$

$$\sum_{i \in \mathcal{U}} x_{fi} \leq S_{\max}^f, \quad f \in \mathcal{F} \quad (4c)$$

$$x_{fi} \in \mathbb{N}, \quad i \in \mathcal{U}, \quad f \in \mathcal{F} \quad (4d)$$

To obtain the above problem's global optimal solution, we introduce binary variable y_{fi}^k that is one if and only if user i caches k segments of file f . Denote by \mathbf{y} the vector consisting of y_{fi}^k :

$$\mathbf{y} = \{y_{fi}^k, \quad i \in \mathcal{U}, \quad f \in \mathcal{F}, \quad k \in [0, S_{\text{rec}}^f]\}.$$

By definition, if $x_{fi} = k$, then $y_{fi}^k = 1$. For example, if $x_{fi} = 3$, then $y_{fi}^3 = 1$ and $y_{fi}^k = 0$ for the case that $k \neq 3$. Thus, the relationship between the optimization variables x_{fi} and y_{fi}^k can be expressed as

$$\begin{cases} x_{fi} = \sum_{k=0}^{S_{\text{rec}}^f} k y_{fi}^k, \quad i \in \mathcal{U}, \quad f \in \mathcal{F}, \\ \sum_{k=0}^{S_{\text{rec}}^f} y_{fi}^k = 1, \quad i \in \mathcal{U}, \quad f \in \mathcal{F}. \end{cases} \quad (5)$$

Define

$$\begin{aligned} e_{fij}^k &\triangleq \mathbb{E}(\min(BM_{ij}, k)) \\ &= \sum_{t=0}^k t \Pr(BM_{ij} = t) + k \Pr(BM_{ij} > k), \end{aligned} \quad (6)$$

where

$$\Pr(BM_{ij} = t) = \begin{cases} \frac{(\lambda_{ij} T_D)^{\frac{t}{B}} e^{-\lambda_{ij} T_D}}{\frac{t}{B}}, & \text{if } (t \bmod B) = 0, \\ 0, & \text{else.} \end{cases} \quad (7)$$

Thus, for any x_{fj} , $\mathbb{E}(\min(BM_{ij}, x_{fj}))$ can be expressed as

$$\mathbb{E}(\min(BM_{ij}, x_{fj})) = \sum_{k=0}^{S_{\text{rec}}^f} e_{fij}^k y_{fj}^k.$$

Moreover, by the proof in Appendix A, it follows that

$$\Delta_{fi}^{\text{n}2} = \max(\Delta_{fi}^{\text{n}}, 0).$$

Therefore, through the above mathematical analysis, ACOCP can be reformulated as

$$\min_{\mathbf{y}} \quad \frac{1}{U} \sum_{i \in \mathcal{U}} \sum_{f \in \mathcal{F}} P_{fi} (\Delta_{fi}^d + \Delta_{fi}^{n2}) \quad (8a)$$

$$\text{s.t.} \quad \Delta_{fi}^{n2} \geq \Delta_{fi}^n, \quad i \in \mathcal{U}, \quad f \in \mathcal{F} \quad (8b)$$

$$\Delta_{fi}^{n2} \geq 0, \quad i \in \mathcal{U}, \quad f \in \mathcal{F} \quad (8c)$$

$$\sum_{k=0}^{S_{\text{rec}}^f} y_{fi}^k = 1, \quad i \in \mathcal{U}, \quad f \in \mathcal{F} \quad (8d)$$

$$\sum_{f \in \mathcal{F}} \sum_{k=0}^{S_{\text{rec}}^f} k y_{fi}^k \leq C_i, \quad i \in \mathcal{U} \quad (8e)$$

$$\sum_{i \in \mathcal{U}} \sum_{k=0}^{S_{\text{rec}}^f} k y_{fi}^k \leq S_{\text{max}}^f, \quad f \in \mathcal{F} \quad (8f)$$

$$y_{fi}^k \in \{0, 1\}, \quad i \in \mathcal{U}, \quad f \in \mathcal{F}, \quad k \in [0, S_{\text{rec}}^f] \quad (8g)$$

where

$$\begin{cases} \Delta_{fi}^d = \sum_{j \in \mathcal{U}, j \neq i} \sum_{k=0}^{S_{\text{rec}}^f} (e_{fij}^k y_{fj}^k) \delta_{\text{D}}, \\ \Delta_{fi}^n = S_{\text{rec}}^f \delta_{\text{N}} - \sum_{j \in \mathcal{U}, j \neq i} \sum_{k=0}^{S_{\text{rec}}^f} (e_{fij}^k y_{fj}^k) \delta_{\text{N}} - \sum_{k=0}^{S_{\text{rec}}^f} (k y_{fi}^k) \delta_{\text{N}}. \end{cases} \quad (9)$$

Note that the definitions of Δ_{fi}^d and Δ_{fi}^n are the reformulations of that in (3).

The above objective function and constraints are linear with respect to \mathbf{y} . Thus, the ACOCP approach can use an off-the-shelf integer programming algorithm from optimization packages, e.g., Gurobi [34], to obtain the global optimal solution. Generally, it can deliver optimal solutions for the small-scale and medium-scale system scenarios effectively. What's more, it serves the purpose of performance benchmarking of any sub-optimal algorithm. Denote by \mathbf{y}^* the global optimal solution of ACOCP. By (5), \mathbf{y}^* can be converted into an approximation solution of COCP, referred to as \mathbf{x}^{lb} . Denote by \mathbf{x}^* the global optimal solution of COCP. By Theorem 2, it follows that

$$\begin{cases} \Delta(\mathbf{x}^{lb}) \geq \Delta(\mathbf{x}^*), \\ \Delta(\mathbf{x}^*) \geq \Delta^{lb}(\mathbf{x}^*), \\ \Delta^{lb}(\mathbf{x}^*) \geq \Delta^{lb}(\mathbf{x}^{lb}). \end{cases} \quad (10)$$

Therefore,

$$\Delta(\mathbf{x}^{lb}) \geq \Delta(\mathbf{x}^*) \geq \Delta^{lb}(\mathbf{x}^{lb}). \quad (11)$$

Eq. (11) indicates that if \mathbf{x}^{lb} is derived, a lower bound, $\Delta^{lb}(\mathbf{x}^{lb})$, of global optimum of COCP is obtained. The lower bound can be used to evaluate the optimality deviation of the solution of ACOCP. Namely, the gap between the approximation solution and the global optimal solution of COCP does not exceed $\Delta(\mathbf{x}^{lb}) - \Delta^{lb}(\mathbf{x}^{lb})$, while heuristic algorithms cannot provide this type of performance assessment. More importantly, it can evaluate the solution of any sub-optimal algorithm, such as the one presented in the next section, because the gap to the global optimum does not exceed the gap to the lower bound.

V. MOBILITY AWARE USER-BY-USER ALGORITHM

Although the ACOCP approach can obtain solutions for up to medium-size scenarios, the computation complexity does not scale well. Thus, we propose a fast yet effective algorithm, i.e., mobility aware user-by-user (MAUU) algorithm. A general description of MAUU is as follows. The users are treated one by one starting with the first user. Initially, the caching content of all the users are set to be empty. The algorithm optimizes the caching content of the first user, and then keeps this content fixed for this user in later iterations while performing the optimization for the other users. Once the cache content allocation of one user is optimized, the remaining segments of each file, denoted by S_{rem}^f , $f \in \mathcal{F}$, will be updated accordingly. The same process repeats for the next user.

A. Optimal Caching for One User

Theorem 3. Optimizing the caching placement of one user can be derived in polynomial time when the caching placements of the other users are given.

Proof: We compute a matrix, called cost matrix and denoted by \mathbf{V} , in which entry $v(f, k)$ represents the current expected total cost if this user caches k segments of file f . The entries of this matrix can be computed using Eq. (20) in Appendix B.

Below a recursive function is introduced to derive the optimal caching placement for the user. We define a second matrix, called the optimal cost matrix, and denote it by \mathbf{W} , in which $w(q, k')$ represents the cost of the optimal solution from considering the first q files using a cache size

Algorithm 1: The MAUU algorithm for COCP

Input: $\mathbf{S}_{\text{rem}}, \mathbf{S}_{\text{rec}}, \mathbf{x}, \mathbf{x}_1, \mathbf{C}, U, F, B, \delta_D, \delta_N$.

Output: \mathbf{x}

```

1: for  $i = 1 : U$  do
2:    $\mathbf{g} \leftarrow \emptyset, \mathbf{V} \leftarrow [0]_{F \times C_i}$ , and  $\mathbf{W} \leftarrow [0]_{C_i \times F}$ 
3:   for  $f = 1 : F$  do
4:     for  $k = 0 : \min(C_i, \mathbf{S}_{\text{rec}}(f), \mathbf{S}_{\text{rem}}(f))$  do
5:        $x_1(f, i) \leftarrow k$ 
6:        $v(f, k) \leftarrow \Delta(\mathbf{x}_1)$ 
7:        $x_1(f, i) \leftarrow 0$ 
8:     for  $q = 1 : F$  do
9:       if  $q < F$  then
10:        for  $k' = 0 : C_i$  do
11:          if  $q = 1$  then
12:             $w(1, k') \leftarrow v(1, \min(k', \mathbf{S}_{\text{rec}}(1), \mathbf{S}_{\text{rem}}(1)))$ 
13:             $\mathbf{g}_{1k'} \leftarrow \{\min(k', \mathbf{S}_{\text{rec}}(1), \mathbf{S}_{\text{rem}}(1))\}$ 
14:          else
15:             $w(q, k') \leftarrow \arg \min\{v(q, r_q) + w(q - 1, k' - r_q), r_q =$ 
16:               $0, 1, \dots, \min(k', \mathbf{S}_{\text{rec}}(q), \mathbf{S}_{\text{rem}}(q))\}$ 
17:             $\mathbf{g}_{qk'} \leftarrow \mathbf{g}_{q-1, k' - r_q^*} \cup \{r_q^*\}$ 
18:          else
19:             $w(F, C_i) \leftarrow \arg \min\{v(F, r_F) + w(F - 1, C_i - r_F), r_F =$ 
20:               $0, 1, \dots, \min(C_i, \mathbf{S}_{\text{rec}}(F), \mathbf{S}_{\text{rem}}(F))\}$ 
21:             $\mathbf{g}_{FC_i} \leftarrow \mathbf{g}_{F-1, C_i - r_F^*} \cup \{r_F^*\}$ 
22:             $\mathbf{S}_{\text{rem}} \leftarrow \mathbf{S}_{\text{rem}} - \mathbf{g}_{FC_i}$ 
23:             $\mathbf{x}^i \leftarrow \mathbf{g}_{FC_i}$ 
24:             $\mathbf{x}_1 \leftarrow \mathbf{x}$ 
25:          return  $\mathbf{x}$ 

```

of k' , $k' = 0, 1, \dots, C$; here C denotes the cache size of the user under consideration. The value of $w(q, k')$ is given by the following recursion:

$$w(q, k') = \arg \min_r \{v(q, r) + w(q - 1, k' - r)\}, \quad (12)$$

where r can vary from 0 to at most $\min\{k', S_{\text{rec}}^q, S_{\text{rem}}^q\}$ due to cache size k' , file recovery threshold S_{rec}^q , and the number of remaining segments S_{rem}^q of file q . Using Eq. (12), the optimal cost for file q is computed when the optimal cost of the first $q - 1$ files is given.

For the overall solution, the optimal cost can be computed using the above recursion for cache size of C and F files. We prove it by mathematical induction. First, when $q = 1$, obviously $w(1, k') = \arg \min_r \{v(1, r)\}$ for all k' . There are $\min\{k', S_{\text{rec}}^q, S_{\text{rem}}^q\} + 1$ possible values of r , and considering these values one by one gives the optimum r^* . Now, assume $w(l, k')$ is optimal for some l . We prove that $w(l + 1, k')$ is optimal. According to the recursive function,

$$w(l + 1, k') = \arg \min_r \{v(l + 1, r) + w(l, k' - r)\}.$$

The possible values for r is from 0 to $\min\{k', S_{\text{rec}}^q, S_{\text{rem}}^q\}$, and for each of the possible values of r , $w(l, k' - r)$ is optimal. This together gives the conclusion that the minimum will be obtained indeed by the $\arg \min$ operation. Thus, $w(q, k')$ is optimal.

Finally, we show that $w(F, C)$ can be computed in polynomial time. By Appendix B, the complexity of computing \mathbf{V} is of $O(CF^2U^2S_{\text{rec}}'^2)$. By the above, the computational complexity of \mathbf{W} is of $O(FC^2)$. Thus, optimizing the cache content of one user runs in $O(CF^2U^2S_{\text{rec}}'^2) + O(FC^2) = O(CF^2U^2S_{\text{rec}}'^2)$ because generally $FU^2S_{\text{rec}}'^2 > C$. \square

B. Algorithm Summary

The algorithmic flow is presented in Algorithm 1. The input parameters consist of \mathbf{S}_{rem} , \mathbf{S}_{rec} , \mathbf{x} , \mathbf{x}_1 , \mathbf{C} , U , F , B , δ_{D} , and δ_{N} . Here, \mathbf{S}_{rem} is a vector consisting of the remaining segments of all the files. The initialization step is to set $\mathbf{S}_{\text{rem}} = \{S_{\text{max}}^1, \dots, S_{\text{max}}^F\}$, $\mathbf{S}_{\text{rec}} = \{S_{\text{rec}}^1, \dots, S_{\text{rec}}^F\}$, and $\mathbf{C} = \{C_1, \dots, C_U\}$. The final caching placement solution is again denoted by \mathbf{x} . However, for the convenience of description, our algorithm treats it as a matrix of size $F \times U$. We also define \mathbf{x}_1 as an auxiliary matrix with the same size as \mathbf{x} . Initially, the algorithm sets all the entries of \mathbf{x} and \mathbf{x}' to zero, i.e., $\mathbf{x} = [0]_{F \times U}$ and $\mathbf{x}_1 = [0]_{F \times U}$.

For a generic iteration for one user, denote by r_q^* the optimal number of segments cached for file q , and denote by vector $\mathbf{g}_{qk'}$ the optimal caching placement for the user under consideration

with cache size of k' and first q files. By Line 1, the users are processed one by one. Line 2 initializes \mathbf{V} , \mathbf{W} , and \mathbf{g} . Lines 3-7 compute matrix \mathbf{V} . Lines 8-19 compute \mathbf{W} and \mathbf{g} . Lines 20-22 update \mathbf{S}_{rem} , the i th column of \mathbf{x} denoted by \mathbf{x}^i , and \mathbf{x}_1 , respectively.

VI. PERFORMANCE EVALUATIONS

We have developed two approaches that lead to solutions of COCP, i.e., the ACOCP approach and the MAUU algorithm. Next, simulations are conducted to evaluate the effectivenesses of the two approaches by comparing them to the lower bound of global optimum and conventional caching algorithms, i.e., random caching [35] and popular caching [36]. The two conventional algorithms consider users one by one. In the former, each user will cache files randomly with respect to the files' request probabilities. That is, the higher the request probability of a file is, the more likely this file will be cached. In the latter, each user will cache the files according to the popularity in terms of the files' request probabilities of this user. Besides, in implementing the two algorithms, to ensure that the collected segments are non-overlapping, the total number of cached segments of each file, for all the users together, does not exceed the number of available segments.

The file request probability follows a Zipf distribution [18], [24], i.e., $P_{fi} = \frac{f^{-\gamma_i}}{\sum_{k \in \mathcal{F}} k^{-\gamma_i}}$, where γ_i is the Zipf parameter for user i . The number of segments for recovering a file f , S_{rec}^f , is randomly selected in $[1, S^*]$, where S^* will vary in the simulations, and each file has the same $\alpha = S_{\text{max}}^f / S_{\text{rec}}^f$. The average number of contacts per unit time for users i and j , $i \neq j$, λ_{ij} , is generated according to a Gamma distribution $\Gamma(4.43, 1/1088)$ [32]. In the simulations, γ_i and C_i are uniform, namely, $\gamma_i = \gamma$ and $C_i = C$ for all i .

A. Performance Comparison

The performance of the ACOCP and the MAUU are shown in Figs. 2-6. The line in green and the line in blue denote the costs by using the MAUU algorithm and the solution of ACOCP (i.e., \mathbf{x}^{lb}), respectively. The line in red represents the cost of the lower bound of global optimum, i.e., $\Delta^{lb}(\mathbf{x}^{lb})$ in (11).

In general, the true optimality gaps of ACOCP approach and MAUU algorithm (or any sub-optimal algorithm) are hard to get, because it is difficult to know the value of global optimum. However, by Section IV, the ACOCP approach provides an effective bound for performance evaluation, because the gap to the global optimum does not exceed the gap to the lower bound.

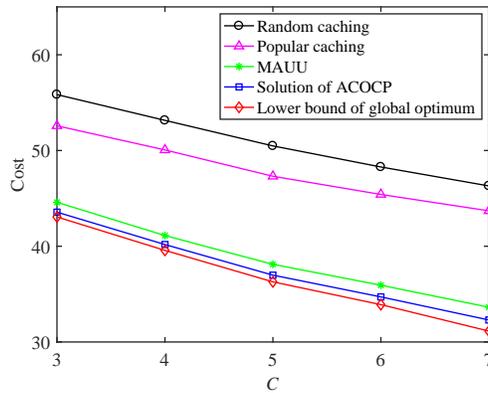


Figure 2. Impact of C on Δ when $U = 8$, $F = 80$, $B = 1$, $\delta_D = 1$, $\delta_N = 30$, $\gamma = 0.8$, $S^* = 4$, $\alpha = 3$, and $T_D = 600s$.

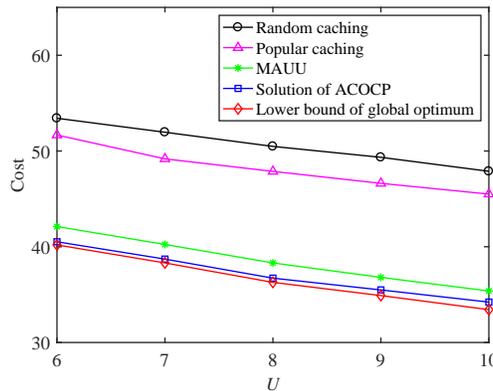


Figure 3. Impact of U on Δ when $F = 80$, $C = 5$, $B = 1$, $\delta_D = 1$, $\delta_N = 30$, $\gamma = 0.8$, $S^* = 4$, $\alpha = 3$, and $T_D = 600s$.

Fig. 2 and Fig. 3 show the impact of C and U , respectively. Overall, the cost linearly decreases with respect to C and U . This is expected, because the users can store more contents with the increase of cache size, and they have more choices and consequently more possibility to collect the needed segments when the number of users grows. In addition, when C and U increase, for the ACOCP approach, the solution is close to the lower bound of global optimum overall, but the gap to the bound increases slightly. For example, by increasing C from 3 to 7, the gap grows from 0.91% to 2.83%. The reason is that, although the global optimal solution of ACOCP can be derived, it is a sub-optimal solution for COCP. Increasing C and U leads to larger solution

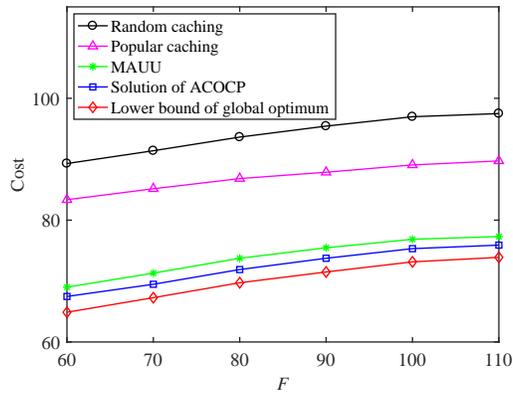


Figure 4. Impact of F on Δ when $U = 8$, $B = 2$, $\delta_D = 1$, $\delta_N = 30$, $C = 5$, $\gamma = 0.8$, $\alpha = 3$, and $T_D = 600s$.

space and may make the bound weaker. However, the worsening is not significant. Similarly, for the MAUU algorithm, the gap increases with the increase of the two parameters. This is because that, giving other users' caching placements, the MAUU algorithm achieves the optimal solution of one user under consideration, whereas this solution is sub-optimal for the system. However, although increasing C and U may slightly decrease the accuracy, the MAUU algorithm remains promising, as the gap is lower than 9%. Finally, the MAUU algorithm outperforms the conventional algorithms consistently in the two figures, especially for big U and C . When $C = 7$, it outperforms the popular caching algorithm by 23.5%, and outperforms the random caching algorithm by 27.8%. Note that U and C represent the system size. Thus, the MAUU algorithm is useful for large-scale system scenarios.

The effect of F is analyzed in Fig. 4. This figure shows results for which the number of segments for recovering a file f is uniform, namely, $S_{\text{rec}}^f = 4$ for any f . It can be observed that the cost first grows with the increase of F . If F becomes excessively big, the impact becomes insignificant due to the limit of cache size and the number of users. Besides, when F increases, the performance difference between the solution of ACOCP and the solution by using MAUU is fairly constant, but the popular caching algorithm outperforms the random caching algorithm significantly. Obviously, increasing F directly leads to higher diversity of files. Thus, for the random caching algorithm, the users are more likely to store the infrequently requested files.

The user average contact rate is proportional to the user average speed [29]. Thus, examining

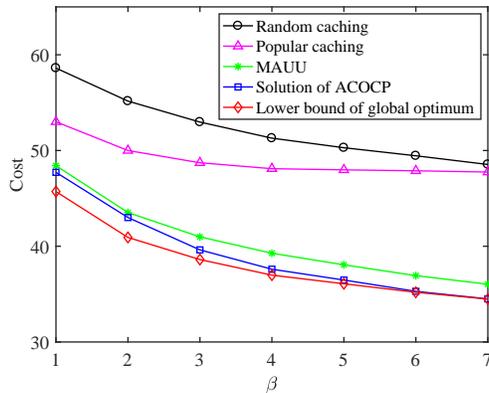


Figure 5. Impact of user average speed on Δ when $U = 8$, $F = 80$, $B = 1$, $\delta_D = 1$, $\delta_N = 30$, $C = 5$, $S^* = 4$, $\alpha = 3$, $\gamma = 0.8$, $\theta = 1/1088$, and $T_D = 600s$.

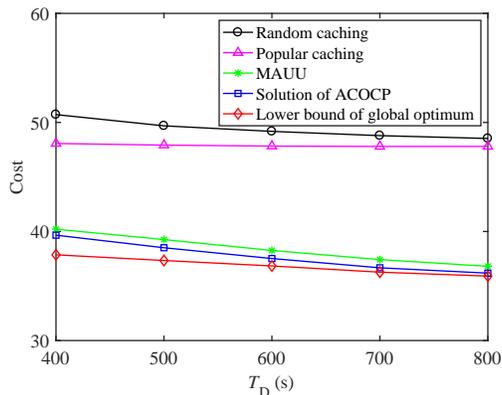


Figure 6. Impact of T_D on Δ when $U = 8$, $F = 80$, $C = 5$, $B = 2$, $\delta_D = 1$, $\delta_N = 30$, $\gamma = 0.8$, $S^* = 4$, and $\alpha = 3$.

the impact of the former reflects also that of the latter. We generate the contact rate for users i and j , λ_{ij} , $i \neq j$, according to a Gamma distribution $\Gamma(\beta, \theta)$. Thus, the average contact rate is $\beta\theta$. Fig. 5 fixes θ , and analyzes the impact of β on Δ . A large average contact rate means more frequent contacts among users, resulted from high mobility. The impact of T_D is shown in Fig. 6. A greater T_D indicates that the users have more time to collect the needed segments. There are two common insights for the two figures. First, the MAUU algorithm outperforms the popular caching algorithm. When the values of the two parameters increase, the improvement is significant. This is because the caching placement by MAUU can be dynamically adapted to

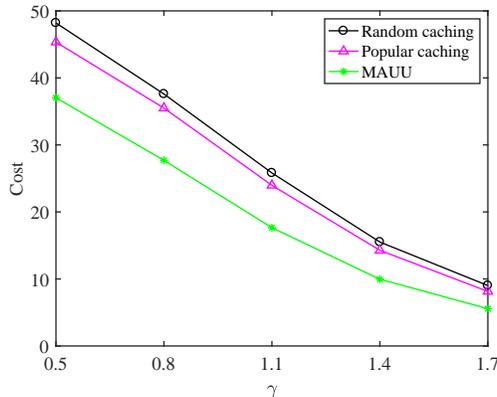


Figure 7. Impact of γ on Δ when $U = 20$, $F = 200$, $B = 1$, $\delta_D = 1$, $\delta_N = 30$, $C = 4$, $S^* = 3$, $\alpha = 3$, and $T_D = 600s$.

the variations in the parameters, whereas the caching placement by popular caching is fixed. Moreover, for the ACOCP approach, the gap to the lower bound progressively decreases. In particular, when $\beta = 1$, the gap is 4.39%. While $\beta = 6$, the gap decreases to 0.28%, indicating that the solution of ACOCP is very close to optimum. The reason is that in such cases, Δ^{lb} approaches Δ .

B. Algorithm Scalability

In Fig. 7, we show additional results for large-scale scenarios via increasing the number of users and files. Specifically, $U = 20$ and $F = 200$. For this case, we compare our scalable MAUU algorithm with the two conventional algorithms. Overall, the costs of the caching solution from the MAUU algorithm and conventional caching algorithms exhibit the same decreasing trend with respect to γ . The MAUU algorithm outperforms the two conventional caching algorithms as the latter algorithms neglect the effect of user mobility. However, there is an additional insight that the improvement of MAUU becomes smaller by increasing γ . The reason is that for high γ , the files' request probability has a large variation. As a result, the users are more inclined to request the popular files.

A general observation is that the ACOCP approach is more accurate than the MAUU algorithm – the cost by using the solution of ACOCP is always less than that of MAUU. Intuitively, this is expected, because the former pays the price of higher complexity due to the use of integer programming. In contrast, the latter is a polynomial time algorithm which is useful for large-scale

scenarios. Therefore, the MAUU algorithm illustrates excellent tradeoff between complexity and accuracy.

VII. CONCLUSIONS

This paper has investigated the caching problem with presence of user mobility, for which the inter-contact model is used to describe the mobility pattern of mobile users. An optimization problem, COCP, has been modelled, analyzed and formulated. The hardness of the problem has been thoroughly proved via a reduction from the 3-SAT problem. For problem-solving, two computational approaches, namely, the ACOCP approach and the MAUU algorithm, have been developed. Performance evaluation shows that the two approaches result in significant improvement in comparison to conventional caching algorithms. Moreover, solving ACOCP leads to an effective approximation scheme, and the MAUU algorithm achieves excellent balance between complexity and accuracy.

An extension of the work is the consideration of a more complicated hierarchical caching architecture with presence of mobility, i.e., caching at both users and base stations. This can be formulated as to minimize the expected delay for recovering one file, with constraints on the total number of encoded segments and cache capacity.

APPENDIX A

To facilitate presentation, define

$$\Delta_{fi}^{n1} \triangleq \mathbb{E}\{\max[S_{\text{rec}}^f - (\sum_{j \in \mathcal{U}, j \neq i} \min(BM_{ij}, x_{fj}) + x_{fi}), 0]\}, \quad (13)$$

and

$$\Delta_{fi}^{n2} \triangleq \begin{cases} S_{\text{rec}}^f - [\mathbb{E}(\sum_{j \in \mathcal{U}, j \neq i} \min(BM_{ij}, x_{fj})) + x_{fi}], \\ \text{if } S_{\text{rec}}^f > \mathbb{E}(\sum_{j \in \mathcal{U}, j \neq i} \min(BM_{ij}, x_{fj})) + x_{fi}, \\ 0, \text{ else.} \end{cases} \quad (14)$$

Given \mathbf{x} , we will prove the relationship between Δ_{fi}^{n1} and Δ_{fi}^{n2} .

(i) When $S_{\text{rec}}^f > \mathbb{E}(\sum_{j \in \mathcal{U}, j \neq i} \min(BM_{ij}, x_{fj})) + x_{fi}$, it follows that

$$\begin{aligned} & S_{\text{rec}}^f - [\mathbb{E}(\sum_{j \in \mathcal{U}, j \neq i} \min(BM_{ij}, x_{fj})) + x_{fi}] \\ &= \mathbb{E}[S_{\text{rec}}^f - (\sum_{j \in \mathcal{U}, j \neq i} \min(BM_{ij}, x_{fj}) + x_{fi})]. \end{aligned} \quad (15)$$

Due to the fact that

$$\begin{aligned} & S_{\text{rec}}^f - \left(\sum_{j \in \mathcal{U}, j \neq i} \min(BM_{ij}, x_{fj}) + x_{fi} \right) \\ & \leq \max[S_{\text{rec}}^f - \left(\sum_{j \in \mathcal{U}, j \neq i} \min(BM_{ij}, x_{fj}) + x_{fi} \right), 0], \end{aligned} \quad (16)$$

it follows that

$$\begin{aligned} & \mathbb{E}\{S_{\text{rec}}^f - \left(\sum_{j \in \mathcal{U}, j \neq i} \min(BM_{ij}, x_{fj}) + x_{fi} \right)\} \\ & \leq \mathbb{E}\{\max[S_{\text{rec}}^f - \left(\sum_{j \in \mathcal{U}, j \neq i} \min(BM_{ij}, x_{fj}) + x_{fi} \right), 0]\}. \end{aligned} \quad (17)$$

Combining (15), (16), with (17), we obtain

$$\begin{aligned} & S_{\text{rec}}^f - [\mathbb{E}\left(\sum_{j \in \mathcal{U}, j \neq i} \min(BM_{ij}, x_{fj})\right) + x_{fi}] \\ & \leq \mathbb{E}\{\max[S_{\text{rec}}^f - \left(\sum_{j \in \mathcal{U}, j \neq i} \min(BM_{ij}, x_{fj}) + x_{fi} \right), 0]\}. \end{aligned} \quad (18)$$

Thus, $\Delta_{fi}^{n1} \geq \Delta_{fi}^{n2}$.

(ii) When $S_{\text{rec}}^f \leq \mathbb{E}\left(\sum_{j \in \mathcal{U}, j \neq i} \min(BM_{ij}, x_{fj})\right) + x_{fi}$, $\Delta_{fi}^{n2} = 0$. Assume that m experiments are conducted. Denote by S_{fi}^r the number of segments of file f collected by user i in the r th experiment, $r = 1, 2, \dots, m$, and $S_{\text{rec}}^f \leq \frac{1}{m} \sum_{r=1}^m S_{fi}^r$. There are two cases. The first case is that user i can successfully recover the file f in each experiment, i.e., $S_{fi}^r \geq S_{\text{rec}}^f$, $r = 1, 2, \dots, m$. In this case, $\Delta_{fi}^{n1} = \Delta_{fi}^{n2}$. The second case is that user i unsuccessfully recovers the file f at least one experiment. For the second case, $\Delta_{fi}^{n1} > 0$. Thus, $\Delta_{fi}^{n1} > \Delta_{fi}^{n2}$.

Combining (i) with (ii), it follows that

$$\Delta_{fi}^{n1} \geq \Delta_{fi}^{n2}. \quad (19)$$

Therefore, $\Delta \geq \Delta^{lb}$.

APPENDIX B

Δ can be simplified as

$$\begin{aligned} \Delta &= \frac{1}{U} \sum_{i \in \mathcal{U}} \sum_{f \in \mathcal{F}} P_{fi} \left[\sum_{j \in \mathcal{U}, j \neq i} \mathbb{E}(S_{fi} - x_{fi}) \delta_{\text{D}} \right. \\ & \quad \left. + \sum_{k=x_{fi}}^{S_{\text{rec}}^f-1} (S_{\text{rec}}^f - k) \Pr(S_{fi} = k) \delta_{\text{N}} \right], \end{aligned} \quad (20)$$

where

$$\Pr(S_{fi} = k) = \Pr\left(\sum_{j \in \mathcal{U}, j \neq i} \min(BM_{ij}, x_{fj}) + x_{fi} = k\right).$$

Computing $\Pr(S_{fi} = k)$ directly by using multiple summations, the computational complexity exponentially increases with U . However, we can use a recursive function with polynomial-time complexity. Define

$$\begin{aligned} \Pr(U, k) &\triangleq \Pr(S_{fi} = k) \\ &= \Pr\left(\sum_{j \in \mathcal{U}, j \neq i} \min(BM_{ij}, x_{fj}) + x_{fi} = k\right). \end{aligned} \quad (21)$$

After some mathematical manipulations, the recursive function can reformulated as

$$\begin{aligned} \Pr(U, k) &= \sum_{t=0}^k [\Pr(\min(BM_{i,U}, x_{f,U}) = t) \\ &\quad * \Pr(U - 1, k - t)]. \end{aligned} \quad (22)$$

The above function manifests that if t segments are collected from user U , then user i will obtain $k - t$ segments from the other $U - 1$ users including itself. In general, $\Pr(U - \tau, \cdot)$ depends on $\Pr(U - \tau - 1, \cdot)$, $\tau = 0, 1, \dots, U - 2$, leading to a recursive process. The overall complexity of computing Δ is $O(FU^2 S'_{\text{rec}}{}^2)$, where $S'_{\text{rec}} = \max_{j \in \mathcal{F}} S_{\text{rec}}^j$.

REFERENCES

- [1] T. Deng, G. Ahani, P. Fan, and D. Yuan, "Cost-optimal caching for D2D networks with presence of user mobility," *submitted to IEEE GLOBECOM*, 2017.
- [2] J. Andrews, S. Buzzi, W. Choi, S. Hanly, A. Lozano, A. Soong, and J. Zhang, "What will 5G be?" *IEEE J. Sel. Areas Commun.*, vol. 32, no. 6, pp. 1065–1082, Jun. 2014.
- [3] I. Hwang, B. Song, and S. Soliman, "A holistic view on hyper-dense heterogeneous and small cell networks," *IEEE Commun. Mag.*, vol. 51, no. 6, pp. 20–27, Jun. 2013.
- [4] X. Ge, H. Cheng, M. Guizani, and T. Han, "5G wireless backhaul networks: Challenges and research advances," *IEEE Netw.*, vol. 28, no. 6, pp. 6–11, Nov. 2014.
- [5] P. Fan, "Coping with the big data: Convergence of communications, computing and storage," *China Commun.*, vol. 13, no. 9, pp. 203–207, Sept. 2016.
- [6] N. Golrezaei, K. Shanmugam, A. Dimakis, A. Molisch, and G. Caire, "Femtocaching: Wireless video content delivery through distributed caching helpers," in *Proc. IEEE INFOCOM*, Mar. 2012, pp. 25–30.
- [7] M. Peng, S. Yan, K. Zhang, and C. Wang, "Fog computing based radio access networks: Issues and challenges," *IEEE Netw.*, vol. 30, no. 4, pp. 46–53, Jul. 2016.
- [8] X. Wang, M. Chen, T. Taleb, A. Ksentini, and V. Leung, "Cache in the air: Exploiting content caching and delivery techniques for 5G systems," *IEEE Commun. Mag.*, vol. 52, no. 2, pp. 131–139, Feb. 2014.

- [9] K. Poularakis, G. Iosifidis, V. Sourlas, and L. Tassiulas, "Exploiting caching and multicast for 5G wireless networks," *IEEE Trans. Wireless Commun.*, vol. 15, no. 4, pp. 2995–3007, Apr. 2016.
- [10] Y. Cui, D. Jiang, and Y. Wu, "Analysis and optimization of caching and multicasting in large-scale cache-enabled wireless networks," *IEEE Trans. Wireless Commun.*, vol. 15, no. 7, pp. 5101–5112, Apr. 2016.
- [11] Y. Cui and D. Jiang, "Analysis and optimization of caching and multicasting in large-scale cache-enabled heterogeneous wireless networks," *IEEE Trans. Wireless Commun.*, vol. 16, no. 1, pp. 250–264, Jan. 2017.
- [12] M. Tao, E. Chen, H. Zhou, and W. Yu, "Content-centric sparse multicast beamforming for cache-enabled cloud RAN," *IEEE Trans. Wireless Commun.*, vol. 15, no. 9, pp. 6118–6131, Sept. 2016.
- [13] B. Zhou, Y. Cui, and M. Tao, "Stochastic content-centric multicast scheduling for cache-enabled heterogeneous cellular networks," *IEEE Trans. Wireless Commun.*, vol. 15, no. 9, pp. 6284–6297, Sept. 2016.
- [14] S. Mosleh, L. Liu, H. Hou, and Y. Yi, "Coordinated data assignment: A novel scheme for big data over cached cloud-RAN," in *Proc. IEEE GLOBECOM*, Dec. 2016, pp. 1–6.
- [15] S. Yan, M. Peng, and W. Wang, "User access mode selection in fog computing based radio access networks," in *Proc. IEEE ICC*, May 2016, pp. 1–6.
- [16] D. Malak, M. Al-Shalash, and J. Andrews, "Optimizing content caching to maximize the density of successful receptions in device-to-device networking," *IEEE Trans. Commun.*, vol. 64, no. 10, pp. 4365–4380, Oct. 2016.
- [17] Z. Chen and M. Kountouris, "D2D caching vs. small cell caching: Where to cache content in a wireless network?" in *Proc. IEEE SPAWC*, July 2016, pp. 1–6.
- [18] Z. Chen, N. Pappas, and M. Kountouris, "Probabilistic caching in wireless D2D networks: Cache hit optimal vs. through optimal," *IEEE Commun. Lett.*, vol. 21, no. 3, pp. 584–587, Mar. 2017.
- [19] H. Kang, K. Park, K. Cho, and C. Kang, "Mobile caching policies for device-to-device (D2D) content delivery networks," in *Proc. IEEE INFOCOM WKSHPs*, Apr. 2014, pp. 299–304.
- [20] J. Rao, H. Feng, C. Yang, Z. Chen, and B. Xia, "Optimal caching placement for D2D assisted wireless caching networks," in *Proc. IEEE ICC*, May 2016, pp. 1–6.
- [21] M. Ji, G. Caire, and A. Molisch, "Wireless device-to-device caching networks: Basic principles and system performance," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 1, pp. 176–189, Jan. 2016.
- [22] —, "Fundamental limits of caching in wireless D2D networks," *IEEE Trans. Inf. Theory*, vol. 62, no. 2, pp. 849–869, Jan. 2016.
- [23] W. Wang, X. Peng, J. Zhang, and K. Letaief, "Mobility-aware caching for content-centric wireless networks: Modeling and methodology," *IEEE Commun. Mag.*, vol. 54, no. 8, pp. 77–83, Aug. 2016.
- [24] K. Poularakis and L. Tassiulas, "Exploiting user mobility for wireless content delivery," in *Proc. IEEE ISIT*, July 2013, pp. 1017–1021.
- [25] —, "Code, cache and deliver on the move: A novel caching paradigm in hyper-dense small-cell networks," *IEEE Trans. Mobile Comput.*, vol. 16, no. 3, pp. 675–687, Mar. 2017.
- [26] T. Wei, L. Chang, B. Yu, and J. Pan, "MPCS: A mobility/popularity-based caching strategy for information-centric networks," in *Proc. IEEE GLOBECOM*, Dec. 2014, pp. 4629–4634.
- [27] H. Li and D. Hu, "Mobility prediction based seamless RAN-cache handover in HetNet," in *Proc. IEEE WCNC*, Apr. 2016, pp. 1–7.
- [28] Y. Guan, Y. Xiao, H. Feng, C.-C. Shen, and L. Cimini, "MobiCacher: Mobility-aware content caching in small-cell networks," in *Proc. IEEE GLOBECOM*, Dec. 2014, pp. 4537–4542.
- [29] R. Wang, J. Zhang, and K. Letaief, "Mobility-aware caching in D2D networks," *Submitted to IEEE Trans. Wireless Commun.*, Jun. 2016 (arXiv:1606.05282v1).

- [30] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.
- [31] V. Conan, J. Leguay, and T. Friedman, "Fixed point opportunistic routing in delay tolerant networks," *IEEE J. Sel. Areas Commun.*, vol. 26, no. 5, pp. 773–782, Jun. 2008.
- [32] A. Passarella and M. Conti, "Analysis of individual pair and aggregate intercontact times in heterogeneous opportunistic networks," *IEEE Trans. Mobile Comput.*, vol. 12, no. 12, pp. 2843–2495, Oct. 2013.
- [33] D. Leong, A. Dimakis, and T. Ho, "Distributed storage allocations," *IEEE Trans. Inf. Theory*, vol. 58, no. 7, pp. 4733–4752, Jul. 2012.
- [34] Gurobi Optimizer. [Online]. Available: <http://www.gurobi.com> .
- [35] B. Blaszczyszyn and A. Giovanidis, "Optimal geographic caching in cellular networks," in *Proc. IEEE ICC*, Jun. 2015, pp. 3358–3363.
- [36] H. Ahlehagh and S. Dey, "Video-aware scheduling and caching in the radio access network," *IEEE/ACM Trans. Netw.*, vol. 22, no. 5, pp. 1444–1462, Oct. 2014.