# Joint Content Placement and Storage Allocation in C-RANs for IoT Sensing Service

Jingjing Yao, *Student Member, IEEE*, and Nirwan Ansari, *Fellow, IEEE*

***Abstract*—The Internet of Things (IoT) sensing service allows systems and users to monitor environment states by transmitting the content sensed by a variety of sensors. Owing to billions of sensors and devices deployed in the IoT system, a huge amount of data (big data) are generated, thus injecting tremendous traffic into the network. Cloud radio access network (C-RAN) is a promising wireless network architecture to accommodate the fast growing IoT traffic and improve the performance of IoT services. Caching in C-RAN, which brings content to the edges, not only alleviates the network traffic, thus improving the end-to-end user quality of service (QoS), but also avoids activating the sensors too frequently, thus reducing their energy consumption. The content placement problem determines what and where to cache in C-RAN. However, the caching performance is highly related to the caching storages. The storage allocation problem determines the storage capacities of network entities. In our work, we jointly optimize the storage allocation problem and content placement problem in a hierarchical cache-enabled C-RAN architecture for IoT sensing service. We formulate the joint problem as an integer linear programming (ILP) model with the objective to minimize the total network traffic cost. The storage allocation problem and content placement problem are constrained by caching storage budgets and cache capacities, respectively. Two heuristic algorithms are proposed in order to reduce the computational complexity of ILP. Extensive simulations have been conducted to demonstrate that the performances of our proposed algorithms approximate the optimal solutions.**

***Index Terms*—Internet of Things, Caching, Cloud Radio Access Network (C-RAN), Content Placement, Storage Allocation.**

## I. Introduction

Internet of Things (IoT) interconnects billions of smart sensors, devices, actuators, as well as human, over a distributed environment to work together toward a smarter physical world. One of the most common applications of IoT is the sensing service that allows systems and users to monitor environment states (e.g., temperature and air pollution level) via a variety of sensors [1]. The IoT sensing data is usually transmitted to the users through the wireless networks for flexible communications. The International Data Corporation (IDC) forcasted that IoT will grow to 50 billion connected devices by 2020 [2]. Owing to billions of sensors and devices deployed in the IoT system, a huge amount of data are generated and injected to the network, which may cause a great quality of service (QoS) degradation.

Cloud radio access network (C-RAN) has been proposed as a novel architecture for 5G cellular networks to provision flexibility with the aid of cloud computing [3] and to accommodate the fast growing dynamic IoT traffic [4]–[6]. C-RAN deploys multiple radio remote heads (RRHs) at cell sites to address the coverage and capacity issues, while the computational functionalities of traditional base stations (BSs) are aggregated in centralized common cloud processing unit, i.e., baseband unit (BBU) pool. With densely deployed RRHs, all user equipments (UEs) in the IoT sensing service have easier access to the core network. The centralized BBU structure in C-RAN facilitates cross-cell cooperation, improve spectrum efficiency, and can improve the QoS for all UEs in IoT [4]. The fronthaul links with high bandwidth and low latency connect RRHs with BBU pool while backhaul links refer to the links between the mobile core network and BBUs. Although C-RAN provides strong computing abilities by sharing computing and storage resources at BBU pool, it still suffers from performance limitation due to the limited capacity of the fronthaul and backhaul links [7]. To overcome this shortage, traffic through the backhaul links and fronthaul links should be alleviated and offloaded.

Owing to the extremely large and heterogenous IoT data, the wireless backhaul may be congested, thus leading to the degradation of user QoS. Caching the IoT data at network entities is regarded as an effective technique to maintain high quality of data transmission [8], reduce content access latency and cut the network traffic cost [9]. On the other hand, sensors usually do not have a fixed power supply and are constrained by their limited energy capacity. In order to reduce the energy consumption, caches can be utilized to store sensing data temporarily so that the frequent activation of sensors (consuming considerable energy) can be avoided. When a user requests sensing data, the network entity that has pre-cached that content can directly provide them to the user instead of obtaining them from the sensors. The fundamental problem of caching is the content placement problem which determines which content should be placed in which cache nodes (e.g., cache storages attached to BBU pools and RRHs).

The caching performance is, however, highly related to capacities of caching storages. With smaller cache storages, limited content can be cached, that may result in degraded service quality as compared with larger cache storages. It is therefore imperative to design efficient strategies to distribute storages across different network cache nodes, in order to best utilize storage resources. Storage allocation decides how many storages should be assigned to each RRH and BBU. Although several research efforts have been devoted to the caching problem in C-RAN, the joint optimization of the storage allocation and content placement has not been investigated. To fill this gap, we study this joint problem in a hierarchical cache-enabled C-RAN architecture (Fig. 1) for IoT sensing service with the objective to minimize the network cost. A cache management module (CMM), which is attached to each

BBU pool, makes storage allocation and caching strategies. We characterize the network cost as the network traffic cost [10], which is incurred by the traffic in network links.

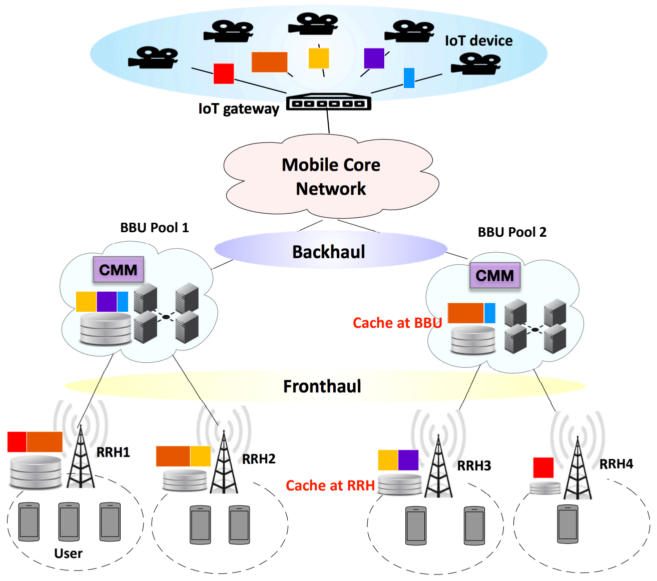


Fig. 1. Hierarchical cache-enabled C-RAN architecture.

The rest of this paper is organized as follows. Section II provides an overview of current works. In Section III, we present the detailed description of our system model. The problem formulation is discussed in Section IV. Owing to the high computational complexity of the problem, we design low-complexity heuristic algorithms in Section V. Simulation results are compared and analyzed in Section VI. Finally, Section VII concludes the paper.

## II. Related Works

Caching has been studied in IoT services. A cache can be employed at Internet routers to reduce the network traffic load [11]. Vural *et al.* [8] introduced in-network caching for IoT services to minimize the content transmission delay. Quevedo *et al.* [12] incorporated caching in the IoT Information Centric Networking (ICN) and demonstrated that caching reduces both energy consumption and bandwidth usage. Niyato et al. [13] studied caching in energy harvesting aided IoT networks to maximize the cache hit ratio for IoT sensing service. Sun and Ansari [14] proposed an energy efficient caching strategy for IoT sensing service to minimize the system delay. However, the above works only consider caching at the IoT gateways. If the IoT gateway is located far from the user, obtaining data from the cache may still inject a large amount of traffic into the network. Caching in C-RAN for IoT sensing service, which brings IoT data closer to users, still requires further investigation.

There are several use cases for caching in IoT networks. Sun and Ansari [14] discussed a smart parking system where parking meters generate IoT data to indicate whether the parking spots are available. Those data are periodically cached at the broker attached to the IoT gateway. Vural *et al.* [8] cached the average temperature of a certain location at the several candidate routers of the Internet. Niyato et al. [13] investigated the noise level measurement system in IoT networks where the measurement data sensed by sensors can be cached at the IoT gateway. Fan *et al.* [15] designed a cache to store a recent visited key of tags in IoT networks to improve the efficiency of authentication.

Several prior works have studied content caching in C-RAN. Effective caching strategies for fog radio access network (F-RAN) was given in [16] where F-RAN refers to the C-RAN architecture that utilizes distributed edge caching techniques [17]. Tao *et al.* [18] proposed to store popular contents in RRHs beforehand and to cooperatively transmit the data to many users at the same time, hence reducing the total transmit power required to satisfy the user requests. Mosleh *et al.* [19] investigated the content caching and beamforming design to minimize the transmission power and backhaul cost. Stephen *et al.* [20] worked on energy efficient transmission in orthogonal frequency division multiple access (OFDM)-based C-RAN with the constraints of fronthaul capacity and user minimum rates. From the the perspective of the social relationship of users, Wang *et al.* [21] discussed the impact of mobile social networks (MSNs) on the performance of edge caching schemes in F-RANs. However, the above works only consider caching at the RRH level. As the caches at RRHs are relatively small, caching at BBUs can provide higher storage capacity and also reduce the traffic through backhaul links.

Joint consideration of caching at both BBU and RRH level has been investigated in [22]. They proposed a novel caching framework, Octopus, which has distributed edge-caches at RRHs and cloud-cache at BBUs. Octopus provides an optimal solution for caching at multiple layers to minimize the total content access delay. However, their system only contains one BBU pool, and hence does not scale well in a large network. Our previous work [23] addressed the content placement problem in C-RAN with multiple BBU pools with the objective to minimize the average file download latency.

The caching performance is constrained by the capacities of caching storages. Too small caching storages may degrade the caching performance while too large ones may waste storage resources. Therefore, it is important to allocate storages according to the demand of users. To the best of our knowledge, this is the first work to jointly consider the storage allocation and content placement problem in the cache-enabled C-RAN architecture for the IoT sensing service. In our work, we investigate the joint optimization of the storage allocation and content placement in a hierarchical cache-enabled C-RAN architecture with the objective to minimize the network traffic cost.

## III. System Model

### A. System Architecture

In our architecture, as shown in Fig. 1, we consider the traffic monitoring application where $F$ cameras are deployed at different locations of a city. Before drivers (i.e., users) pass the city, they may want to know the traffic conditions (e.g., congested, blocked, or smooth) of several areas in order to choose the right paths to drive. These cameras can send IoT data (e.g., pictures or videos) to the drivers through the IoT gateway and C-RANs. If thousands of drivers get the data from these cameras, tremendous traffic will be injected into the network. However, if the data have been cached, they can be sent directly to the users, hence alleviating the network traffic.

We consider the hierarchical cache-enabled C-RAN architecture consisting of a set $\mathcal{R} = \{1, 2, ..., R\}$ of $R$ RRHs and a set $\mathcal{B} = \{1, 2, ..., B\}$ of $B$ BBUs. RRHs are connected with their associated BBU pools which exchange data with the mobile core network through backhaul links. Fronthaul links connect BBU pools and densely deployed RRHs which communicate with the UEs through wireless channels. We consider each user only receives data from one RRH and each RRH only connects with one BBU pool. The BBU pool, associated with the RRH $r$, is denoted as $M_r$. Moreover, we consider assigning the caching storages to each RRH and BBU within certain storage budgets. We assume the total caches of RRHs and BBU pools should not surpass the upper limit budget $C^{th}$ and $D^{th}$, respectively, which are specified by the mobile network operator (MNO).

## B. Caching Process

CMMs can monitor all the requests generated from the users so that learning and prediction algorithms can be utilized to estimate the content popularity distribution and caching strategies can be made. A global indexing table can be maintained in each CMM to enable content lookup. A request from a user will first be forwarded to CMM, which searches its indexing table to determine the location of the content. If the content has been cached at the local RRH, CMM will inform RRH to send the content to the user directly without incurring fronthaul and backhaul traffic overhead. Otherwise, CMM will inform BBU pool to delivery the content. If CMM cannot locate the requested file in any cache, it will forward the request to the IoT gateway to obtain the content. As dividing the IoT data into small chunks and caching them at different levels increases the system complexity, we assume that each datum is indivisible and can only be cached at one cache node.

Since network traffic is dynamic, caches should be updated periodically (e.g., an hour). At the beginning of every period, CMM first re-optimizes the content placement and storage allocation strategies. If the re-optimized strategies are different from those of the last period, the caches can be updated and caching storages can be reallocated accordingly. We assume the traffic conditions do not change much within each period. Although the real IoT data is dynamically changing, the cached data can still reflect the traffic conditions. For example, the traffic conditions of certain roads are always congested during the rush hours.

## IV. Problem Formulation

We formulate our problem as a joint optimization problem of the storage allocation and content placement with the objective to minimize the expected network traffic cost. Specifically, we provide insights on how many storages, denoted as $c_r$ and $d_b$, should be allocated for caching at RRH $r$ and BBU pool $b$, respectively. We also investigate which IoT data should be cached at each RRH and BBU pool. We define two Boolean variables $x_{rf}$ and $y_{bf}$ to indicate whether IoT data from device $f$ should be cached at RRH $r$ and BBU pool $b$, respectively. We denote the data popularity as $P_{rf}$, i.e., the probability of users from RRH $r$ in requesting IoT data from device $f$. Thus, the problem is formulated as follows:

$$\textbf{P0:} \min_{\boldsymbol{x},\boldsymbol{y},\boldsymbol{c},\boldsymbol{d}} \sum_{f=1}^{F}\sum_{r=1}^{R}\{P_{rf}N_rL_f[W^R x_{rf} + (W^R+W^B)(1-x_{rf})y_{M_rf} + (W^R+W^B+W^C)(1-x_{rf})(1-y_{M_rf})]\} \tag{1}$$

s.t.

$$\sum_{r=1}^{R} c_r \le C^{th}, \tag{2}$$

$$\sum_{b=1}^{B} d_b \le D^{th}, \tag{3}$$

$$\sum_{f=1}^{F} x_{rf}L_f - c_r \le 0,\ \forall r \in \mathcal{R}, \tag{4}$$

$$\sum_{f=1}^{F} y_{bf}L_f - d_b \le 0,\ \forall b \in \mathcal{B}. \tag{5}$$

There are three items in the square brackets of Eq. (1). The first one indicates that IoT data from device $f$ are cached at RRH $r$, thus constituting the traffic through wireless channels between users and RRHs. The second one denotes that IoT data from device $f$ are cached at BBU pool $b$ and the traffic goes through wireless channels and fronthaul links. The third one implies that data from device $f$ are obtained from the mobile core network and the traffic comes from wireless channels and both fronthaul links and backhaul links. Note that if IoT data from device $f$ are cached at both RRH and BBU pool, they can only be obtained from the RRH. Eqs. (2) and (3) imply that the allocated cache at all RRHs and BBU pools should not surpass the storage budget $C^{th}$ and $D^{th}$. Eqs. (4) and (5) indicate that all the data in each RRH and BBU pool should not exceed their storage capacities. Due to the product terms in Eq. (1), this problem is non-linear which makes it difficult to solve. Therefore, we introduce another Boolean variable $z_{rf}$ to enable $z_{rf} = x_{rf}y_{M_rf}$. To ensure the transformed problem and the original problem is equivalent, Eqs. (6), (7) and (8) should be satisfied. Then, the transformed problem can be written as follows.

$$\textbf{P1:} \min_{\boldsymbol{x},\boldsymbol{y},\boldsymbol{c},\boldsymbol{d},\boldsymbol{z}} \sum_{f=1}^{F}\sum_{r=1}^{R}\{P_{rf}N_rL_f[(W^R+W^B+W^C) - (W^B+W^C)x_{rf} - W^C y_{M_rf} + W^C z_{rf}]\}$$

s.t. (2), (3), (4), (5),

$$z_{rf} \le x_{rf}, \quad \forall r \in \mathcal{R}, f \in \{1, ..., F\}, \tag{6}$$

$$z_{rf} \leq y_{M_r f}, \quad \forall r \in \mathcal{R}, f \in \{1, ..., F\}, \tag{7}$$

$$z_{rf} \geq x_{rf} + y_{M_r f} - 1, \quad \forall r \in \mathcal{R}, f \in \{1, ..., F\}. \tag{8}$$

The transformed problem **P1** is an integer linear programming (ILP) problem which can be addressed via the exhaustive search algorithm or by CPLEX at the expense of high computational complexity. In order to reduce the computational complexity, we design several low-complexity suboptimal heuristics in the next section and compare their performances with the optimal solutions of ILP.

## V. Heuristic Algorithms

Although ILP can obtain the optimal solutions of the joint problem of storage allocation and content placement, it suffers from high computational complexity. In order to improve time efficiency, we propose two heuristic algorithms to obtain fast solutions in this section. We decompose this joint optimization problem into two subproblems, including the storage allocation problem and content placement problem. We try to solve the storage allocation problem first and then utilize its results as the input to the content placement problem.

### A. Storage Allocation Problem

In the storage allocation problem, storages are allocated to each RRH and BBU pool subject to the storage budgets of total RRHs and BBU pools to maximize the resource utilization. In order to maximize the resource utilization, the algorithm should be designed according to different traffic demands, which are related to the number of users, the data popularity and the length of IoT data. The RRHs and BBU pools with higher traffic demands should be allocated with more caching storages. Hence, we propose the traffic based allocation algorithm, which allocates storages proportionally based on different traffic demands. The specific process of this algorithm is shown in Alg. 1. Lines 2-8 calculate the storage for each RRH and Lines 9-15 calculate the storage for each BBU pool.

**Algorithm 1:** Traffic based Allocation Algorithm

```
Input  : $R, B, N_r, P_{rf}, L_f, C^{th}, D^{th}$
Output: storage allocation strategy $c_r, d_b$
1  Total RRHs' traffic $sumt = 0$;
2  for each RRH $r$ do
3      Calculate traffic demand $t_r$;
4      $sumt = sumt + t_r$;
5  end
6  for each RRH $r$ do
7      $c_r = t_r / sumt * C^{th}$;
8  end
9  Total BBU pools' traffic $sumT = 0$;
10 for each BBU pool $b$ do
11     Calculate traffic demand $T_b$;
12     $sumT = sumT + T_b$;
13 end
14 for each BBU pool $b$ do
15     $d_b = T_b / sumT * D^{th}$;
16 end
17 return $\boldsymbol{c}, \boldsymbol{d}$ ;
```

### B. Content Placement Problem

The content placement problem determines which file should be cached at each RRH and BBU pool to minimize the traffic cost. We propose two heuristics to solve the content placement problem.

*1) Global Popularity based Greedy Algorithm:* As the data popularity plays an important role in designing caching strategies, caching data with higher popularity provides better performance. We leverage the greedy idea to cache as many popular data as possible at each caching node. Specifically, our global popularity based greedy algorithm first caches the most popular data at each RRH up to its caching storage capacity. Then, each BBU pool caches the most popular data except those that are already cached at one of the connected RRHs up to the allowable storage capacity. This is because when a RRH caches the requested content, it can directly transmit the data to the UE and hence caching at the connected BBU pool is a waste of the BBU storage. Alg. 2 details the procedure. Lines 1-5 depict the caching process of each RRH and Lines 6-12 describe the caching process of each BBU pool. The shortcoming of this algorithm is that when a RRH caches data from device $f$, the data cannot be cached at the associated BBU pool $b$. Hence, other RRHs under BBU pool $b$ cannot obtain these data from BBU $b$.

**Algorithm 2:** Global Popularity based Greedy Algorithm

```
Input  : $B, R, M_r, F, L_f, P_{rf}, N_r, W^R, W^B, W^C, \boldsymbol{c}, \boldsymbol{d}$
Output: Traffic cost $P$, content placement solutions
        $\boldsymbol{x}$, $\boldsymbol{y}$
1  for each RRH $r$ do
2      for data from each device $f$ in descending order
         of $P_f$ do
3          Cache data from device $f$ until RRH $r$ is full;
4      end
5  end
6  for each BBU pool $b$ do
7      for data from each device $f$ in descending order
         of popularity $P_f$ do
8          if data from device $f$ is not cached at one of
             RRHs under BBU pool $b$ then
9              Cache data from device $f$ until BBU pool
                 $b$ is full;
10         end
11     end
12 end
13 Calculate the traffic cost $P$;
14 return $P, \boldsymbol{x}, \boldsymbol{y}$
```

*2) Local Popularity based Knapsack Algorithm:* In order to alleviate the network traffic cost, it is important to reduce the traffic across the network. Hence, it is beneficial to cache as many data as possible at the RRHs and then the BBU pools. The proposed local popularity based knapsack algorithm can avoid the shortage of Alg. 2 by considering the local popularity for each RRH and BBU pools. Specifically, we denote $P_{bf}$ as the data popularity of data from device $f$ seen by BBU pool $b$. We first decide the contents to be cached for each RRH according to the data popularity $P_{rf}$ and data length $L_f$ within the storage capacity $c_r$. This content placement decision for RRH $r$ can be formulated as

$$\textbf{P2:} \quad \max_{\boldsymbol{x}} \quad \sum_{f=1}^{F} P_{rf} L_f x_{rf} \tag{9}$$

$$s.t. \quad \sum_{f=1}^{F} L_f x_{rf} - c_r \leq 0 \tag{10}$$

where Eq. (10) indicates that all the cached data in RRH $r$ should not surpass its capacity. Eq. (9) implies that we prefer to cache the data with higher popularity and larger data length. In order to calculate the file popularities seen by each BBU pool, we update the local popularity $P_{r\mathcal{F}} = 0$ where $\mathcal{F}$ denotes the files cached at RRH $r$, after caching decisions have been made for RRH $r$. This is because files $\mathcal{F}$ from RRH $r$ can be considered not needed (may still be required by other RRHs) for the connected BBU pool if they already exist in RRH $r$. Then, the data popularity $P_{bf}$ can be calculated as $P_{bf} = \sum_{r \in R_b} P_{rf}, \forall f$, where $R_b$ denotes the RRHs covered by BBU pool $b$. By the same token, according to data popularity $P_{bf}$ and file length $L_f$, the content decision for each BBU pool $b$ can be formulated as

$$\textbf{P3:} \quad \max_{\boldsymbol{y}} \quad \sum_{f=1}^{F} P_{bf} L_f y_{bf} \tag{11}$$

$$s.t. \quad \sum_{f=1}^{F} L_f y_{bf} - d_b \leq 0, \tag{12}$$

where Eq. (12) indicates the storage capacity constraint of BBU pool $b$.

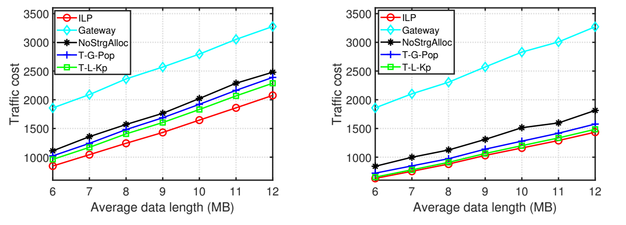
Alg. 3 depicts the processes of the local popularity based knapsack algorithm. RRHs choose their contents to cache in Lines 1-23. The data popularity $P_{bf}$ is obtained in Lines 24-25. Lines 26-28 decide which data should be cached at each BBU pool. We can observe that problem **P2** and problem **P3** have the same form of the 0-1 knapsack problem [24], where $L_f$ is the weight of item $f$, $c_r$ and $d_b$ are the knapsack capacities and $P_{rf} L_f$ is the value of each item. We can use the dynamic programming [24] to solve these two problems. The idea of dynamic programming is to decompose the problem into smaller problems and then find a relation between the structure of the optimal solution and the solutions of the smaller problems. For example, Lines 2-23 describe the dynamic programming method for problem **P2**. To decompose the problem into smaller problems, we construct matrix $m$ whose entry $m(f, j)$ refers to the maximal objective value we can obtain when storage $j$ is used by caching files from $\{1, 2, ..., f\}$. Hence, the optimal solution is $m(F, c_r)$. The relation between the larger problem and the smaller problems is

**Algorithm 3:** Local Popularity based Knapsack Algorithm

```
Input  : $B, R, M_r, F, L_f, P_{rf}, N_r, W^R, W^B, W^C, \boldsymbol{c}, \boldsymbol{d}$
Output: Traffic cost $P$, content placement solutions
        $\boldsymbol{x}$, $\boldsymbol{y}$
1  for each RRH $r$ do
2      Initialize $m$ as a $(F+1) \times (c_r+1)$ zero matrix;
3      for $f = 1 \to F$ do
4          for $j = 1 \to c_r$ do
5              if $L_f > j$ then
6                  $m(f+1, j+1) = m(f, j+1)$;
7              else
8                  $m(f+1, j+1) = \max\{m(f, j+1), m(f, j+1-L_f) + P_{rf} L_f\}$;
9              end
10         end
11     end
12     $opt = m(F+1, c_r+1), f = F, j = c_r$;
13     while $opt > 0$ do
14         while $m(f+1, j+1) == opt$ do
15             $j = j - 1$;
16         end
17         $f = f + 1$;
18         $x_{rf} = 1$;
19         $j = j - L_f$;
20         $f = f - 1$;
21         $opt = m(f+1, j+1)$;
22     end
23 end
24 Update local popularity $P_{rf}$;
25 Calculate the popularity $P_{bf}$;
26 for each BBU pool $b$ do
27     Solve the 0-1 knapsack problem P3;
28 end
29 Calculate the traffic cost $P$;
30 return $P, \boldsymbol{x}, \boldsymbol{y}$
```

$$m(f, j) = \begin{cases} m(f-1, j), & \text{if } j < L_f, \\ \max\{m(f-1, j), m(f-1, j-L_f) + P_{rf} L_f\}, & \text{otherwise.} \end{cases}$$

When the storage capacity is $j$ and less than the data length $L_f$ from device $f$, the data cannot be cached. Therefore, we can remove data from device $f$ in our solution and only consider caching data from $\{1, 2, ..., f-1\}$, i.e., $m(f, j) = m(f-1, j)$. Otherwise, we choose the larger value between caching or not caching the data from device $f$. The first item in the braces means the data from device $f$ is not cached in the RRH, hence contributing no value to the larger problem nor occupying any storage. The second item implies the data from device $f$ is cached in the RRH, hence adding $P_{rf} L_f$ value to the larger problem and consuming $L_f$ of the storage capacity. We recursively calculate all the entries of $m$ according to Lines 5-8. In Lines 12-23, we backtrack the entries of $m$ and decide



(a) $\delta = 0.1$ (b) $\delta = 0.9$

Fig. 2. Traffic cost versus average data length.

which files are cached at RRH $r$. We can utilize the similar dynamic programming method to solve problem **P3** in Line 27.

## C. Complexity Analysis

We combine algorithms for both the storage allocation problem and the content placement problem into two algorithms which are denoted as T-G-Pop and T-L-Kp. T-G-Pop combines Alg. 1 and Alg. 2 while T-L-Kp consists of Alg. 1 and Alg. 3. In Alg. 1, the for-loops in Lines 2-5 and Lines 6-8 are executed $2R$ times and the for-loops in Lines 10-13 and Lines 14-16 are run $2B$ times. Hence, the overall computational complexity of Alg. 1 is $\mathcal{O}(2R + 2B) = \mathcal{O}(R + B)$. In Alg. 2, the for-loop from Line 1 to Line 5 is run $RF$ times. At the worst case, the for-loop in Lines 6-12 is executed $BF$ times. Therefore, the overall computational complexity of Alg. 2 is $\mathcal{O}(RF + BF)$. In Alg. 3, the computational complexities of dynamic programming in solving 0-1 knapsack problems **P2** and **P3** are $\mathcal{O}(Fc_r)$ and $\mathcal{O}(Fd_b)$, respectively [24]. Hence, the for-loop from Line 1 to Line 11 is executed $\sum_{r=1}^{R} Fc_r = FC^{th}$ times. Similarly, the for-loop in Lines 26-28 is run $\sum_{b=1}^{B} Fd_b = FD^{th}$ times. Hence, the overall computational complexity of Alg. 3 is $\mathcal{O}(FC^{th} + FD^{th})$. Thus, the computational complexity of T-G-Pop is $\mathcal{O}(R + B + FR + FB) = \mathcal{O}(F(R + B))$ and the time complexity of T-L-Kp is $\mathcal{O}(R + B + FC^{th} + FD^{th}) = \mathcal{O}(F(C^{th} + D^{th}))$. We can observe that the computational complexity of T-L-Kp is much larger than that of T-G-Pop.

## VI. Performance Evaluation

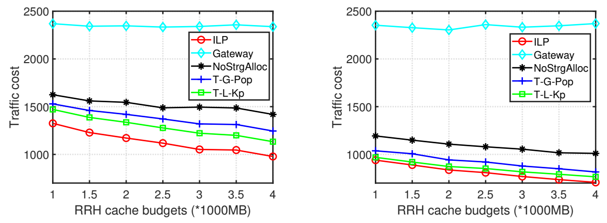
In this section, we evaluate the performance of our proposed heuristic algorithms, T-G-Pop and T-L-Kp, by comparing them with the optimal solutions of the ILP model obtained from CPLEX. We also utilize the existing works in [14] which only considers caching at the IoT gateway for IoT sensing service and [23] which only investigates the content placement problem for caching in C-RANs for comparison. We denote the two existing works as Gateway and NoStrgAlloc, respectively.

In our simulation, we assign the number of RRHs $R = 30$ and the number of BBU pools $B = 3$. As a backhaul link is shared by more users than a fronthaul link, the traffic through the backhaul linked is more likely to cause congestions. Hence, we assume the traffic cost of backhaul links $W^C$ should be greater than that of fronthaul links $W^B$. For the same reason, $W^B$ should be greater than the traffic cost of wireless links $W^R$. We normalize $W^R$, $W^B$ and $W^C$ as 1, 2 and 4 per *MB*. Note that these three parameters are consistent with those used in [10]. A total of $N = 1000$ users are randomly distributed among different RRHs. The number of IoT devices $F = 100$. In all the simulations, we specify the average data length $L$. The actual data lengths in all IoT devices are randomly chosen from $0.8L$ to $1.2L$ in all IoT devices. Predicting data popularity is a challenging task. Techniques such as [25], [26] involve analyzing requesting data around a specific RRH $r$ to estimate the data popularity distribution $P_{rf}$. The popularity variable can also be estimated by considering the position of the camera sensors (e.g., a principle road has higher popularity).

In our work, we assume the popularity profile of different data can be characterized by the Zipf distribution which is widely used to characterize the web caching data [27] and is recently utilized to characterize the IoT data [28]–[30]. The feature of Zipf distribution is that the first few popular data account for the majority of requests. This feature matches the scenario in our work when most users are interested in several principal roads. In the Zipf distribution, we can obtain the probability of IoT data from device $f$ as $P_f = \frac{f^{-\delta}}{\sum_{j=1}^{F} j^{-\delta}}$, where $\delta$ describes the skewness of data popularity and reflects different levels of skewness of the distribution. A higher $\delta$ implies larger differences among different $P_f$s, i.e., most users request the IoT data from a small number of devices. Users request the IoT data from all devices with equal probability when $\delta = 0$. As different RRHs have their own preferences, we denote the probability that users from RRH $r$ requesting IoT data from device $f$ as $P_{rf}$, which is uniformly distribution among different RRHs under the condition $P_f = \sum_{r=1}^{R} P_{rf}$.

We first compare the performances of T-G-Pop and T-L-Kp with different average data lengths. We assign the total storage budgets for RRHs $C^{th} = 1500$ *MB* and BBU pools $D^{th} = 1500$ *MB*. Fig. 2 shows the traffic cost with different average IoT data lengths ranging from 6 *MB* to 12 *MB*. We conduct the simulation under two different skewness parameters, 0.1 and 0.9. Fig. 2(a) depicts the comparisons with the smaller skewness parameter while Fig. 2(b) with the larger skewness parameter. We can observe from both Fig. 2(a) and Fig. 2(b) that the traffic cost increases with the average file length because larger file lengths bring more traffic to the network. ILP provides the lowest traffic cost because ILP from CPLEX can derive the exact optimal solutions. Caching only at IoT gateway incurs the highest traffic cost and caching in C-RANs without storage allocation the second highest. T-L-Kp performs better than T-G-Pop because T-L-Kp considers local popularity instead of global popularity and can avoid the drawback of global popularity which we discussed in Section V. In addition, both T-L-Kp and T-G-Pop perform close to the optimal solutions from ILP.

In comparing Fig. 2(b) with Fig. 2(a), we can observe that with the higher skewness parameter, the traffic cost reduces because with limited caching capacity, the cached popular data can serve more UEs and hence less traffic is incurred to the network. For example, each RRH cache can only accommodate the data from one device. If the data are much more popular than others (i.e., skewness parameter is quite large), each RRH will cache the same most popular data and can still serve most content requests. In Fig. 2(b), both T-L-Kp and T-G-Pop yield results closer to the optimal solutions. On that occasion, T-G-Pop would also be a good choice because it has much lower computational complexity than T-L-Kp.

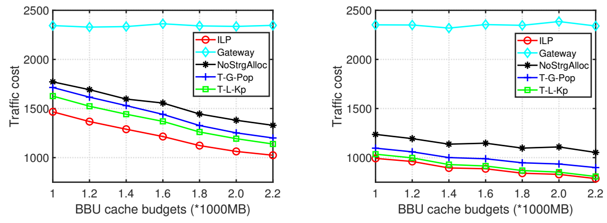


(a) $\delta = 0.1$ (b) $\delta = 0.9$

Fig. 3. Traffic cost versus RRH cache budgets.

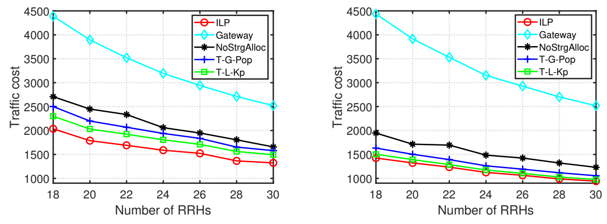
The cache storage is an important metric that should be considered in designing caching strategies. We then compare the performances of T-G-Pop and T-L-Kp with different RRH cache budgets $C^{th}$. We set the average data length to 10 *MB*. Fig. 3 shows the traffic cost with different RRH cache budgets ranging from 1000 *MB* to 5500 *MB*. Fig. 3(a) and Fig. 3(b) illustrate the performances with different skewness parameters. From both Fig. 3(a) and Fig. 3(b), we can observe that the traffic cost decreases with RRH cache budgets because more IoT data can be cached at RRHs and hence less traffic is incurred to both the fronthaul and backhaul links. ILP solutions are the best, caching at IoT gateway yields the worst solutions and caching in C-RANs without storage allocation the second worst. T-L-Kp performs better than T-G-Pop because T-L-Kp considers local popularity instead of global popularity. As compared to Fig. 3(a), Fig. 3(b) shows that with the higher skewness parameter, the traffic cost decreases because the cached data are requested by most users, and there is less traffic injected into the network. In Fig. 3(b), both T-L-Kp and T-G-Pop obtain closer solutions to those of ILP as compared with Fig. 3(a).

Similarly, we compare the performance of T-G-Pop and T-L-Kp with different BBU cache budgets $D^{th}$. We set the average data length to 10 *MB*. Fig. 4 shows the traffic cost for different BBU cache budgets ranging from 1000 *MB* to 2800 *MB*. Fig. 4(a) and Fig. 4(b) illustrate the performances with different skewness parameters. In both Fig. 4(a) and Fig. 4(b), the traffic cost reduces with the BBU cache budget because more data can be cached at BBU pools for larger



(a) $\delta = 0.1$ (b) $\delta = 0.9$

Fig. 4. Traffic cost versus BBU cache budgets.



(a) $\delta = 0.1$ (b) $\delta = 0.9$

Fig. 5. Traffic cost versus number of RRHs.

BBU cache budgets and hence less traffic is introduced to the backhaul links. ILP incurs the lowest traffic cost. Both T-G-Pop and T-L-Kp perform better than the existing works of caching only at IoT gateway and caching in C-RANs without storage allocation. T-G-Pop incurs more traffic cost than T-L-Kp because it only considers the global file popularity. As compared to Fig. 4(a), Fig. 4(b) illustrates that the traffic cost reduces with higher skewness parameter because the cached data are requested by more users, thus resulting in less traffic from the mobile core network. Fig. 4(b) shows that solutions obtained by both T-L-Kp and T-G-Pop are close. The reason is that the larger skewness parameter favors caching the popular files. In addition, both T-G-Pop and T-L-Kp perform very close to ILP as shown in Fig. 4(b).

We also investigate the impacts of different numbers of RRHs on the traffic cost. Fig. 5 depicts the traffic cost for the number of RRHs ranging from 18 to 30. The performances with different skewness are illustrated in Fig. 5(a) and Fig. 5(b). The traffic cost decreases as the number of RRHs increases because more RRHs imply that more popular data can be cached and thus less traffic will be injected into the network. Our proposed T-G-Pop and T-L-Kp perform better than the existing works Gateway and NostrgAlloc. The traffic cost decreases with a larger skewness for the same reason as shown in Fig. 2, Fig. 3 and Fig. 4. When the skewness is large, both T-G-Pop and T-L-Kp become closer to the ILP optimal solutions as compared with the small skewness, and T-L-Kp performs very close to ILP.

## VII. Conclusion

In this paper, we have studied the problem of jointly optimizing the storage allocation and content placement in a hierarchical cache-enabled C-RAN architecture for IoT sensing service where contents can be cached at both the RRH and BBU level by minimizing the total network traffic cost with the constraints of RRHs and BBU pools storage budgets. An ILP model has been proposed to address this joint optimization problem. However, the ILP model incurs high computational complexity and does not scale well as the problem size increases. In order to reduce the complexity, we have proposed two heuristic algorithms T-G-Pop and T-L-Kp, which provide near-optimal solutions with lower computational complexities. Simulation results have demonstrated that our proposed algorithms perform better than the existing works and can achieve near-optimal solutions. We have also justified that both T-L-Kp and T-G-Pop yield closer solutions to the optimal solutions of ILP when the skewness is large. On that occasion, T-L-Kp performs very close to the optimal solutions. However, T-G-Pop would also be a good choice because it provides near-optimal solutions but with much lower computational complexity than T-L-Kp.

## REFERENCES

[1] X. Sun and N. Ansari, "Traffic load balancing among brokers at the IoT application layer," *IEEE Trans. Netw. Service Manag.*, vol. 15, no. 1, pp. 489–502, Mar. 2018.

[2] M. K. Marcus Torchia, Ashutosh Bisht, "IDC's worldwide internet of things connectivity taxonomy," Tech. Rep., Oct. 2017.

[3] C. Mobile, "C-RAN: the road towards green RAN," *White Paper, ver 3*, vol. 2, 2011.

[4] K. Wang, W. Zhou, and S. Mao, "On joint BBU/RRH resource allocation in heterogeneous Cloud-RANs," *IEEE Internet Things J.*, vol. 4, no. 3, pp. 749–759, Jun. 2017.

[5] Y.-S. Jeong, J. S. Park, and J. H. Park, "An efficient authentication system of smart device using multi factors in mobile cloud service architecture," *INT J COMMUN SYST.*, vol. 28, no. 4, pp. 659–674, 2015.

[6] L.-W. Chen, Y.-F. Ho, W.-T. Kuo, and M.-F. Tsai, "Intelligent file transfer for smart handheld devices based on mobile cloud computing," *INT J COMMUN SYST.*, vol. 30, no. 1, pp. 2403–2415, 2017.

[7] M. Peng, Y. Sun, X. Li, Z. Mao, and C. Wang, "Recent advances in cloud radio access networks: System architectures, key techniques, and open issues," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 3, pp. 2282–2308, third quarter 2016.

[8] S. Vural, P. Navaratnam, N. Wang, C. Wang, L. Dong, and R. Tafazolli, "In-network caching of internet-of-things data," in *Proc. IEEE ICC 2014*, Jun. 2014, pp. 3185–3190.

[9] T. Han and N. Ansari, "Network utility aware traffic load balancing in backhaul-constrained cache-enabled small cell networks with hybrid power supplies," *IEEE Trans. Mobile Comput.*, vol. 16, no. 10, pp. 2819–2832, Oct. 2017.

[10] X. Li, X. Wang, and V. C. M. Leung, "Weighted network traffic offloading in cache-enabled heterogeneous networks," in *Proc. IEEE ICC 2016*, 2016, pp. 1–6.

[11] J. Li, Y. Shvartzshnaider, J. A. Francisco, R. P. Martin, and D. Raychaudhuri, "Enabling internet-of-things services in the mobility first future Internet architecture," in *Proc. IEEE WoWMoM 2012*, Jun. 2012, pp. 1–6.

[12] J. Quevedo, D. Corujo, and R. Aguiar, "A case for ICN usage in IoT environments," in *Proc. IEEE GLOBECOM 2014*, Dec. 2014, pp. 2770–2775.

[13] D. Niyato, D. I. Kim, P. Wang, and L. Song, "A novel caching mechanism for Internet of Things (IoT) sensing service with energy harvesting," in *Proc. IEEE ICC 2016*, May 2016, pp. 1–6.

[14] X. Sun and N. Ansari, "Dynamic resource caching in the IoT application layer for smart cities," *IEEE Internet Things J.*, vol. 5, no. 2, pp. 606–613, Apr. 2018.

[15] K. Fan, C. Liang, H. Li, and Y. Yang, "LRMAPC: A lightweight RFID mutual authentication protocol with cache in the reader for IoT," in *Proc. IEEE ICIST 2014*, Sep. 2014, pp. 276–280.

[16] R. Tandon and O. Simeone, "Cloud-aided wireless networks with edge caching: Fundamental latency trade-offs in fog radio access networks," in *Proc. IEEE ISIT 2016*, 2016, pp. 2029–2033.

[17] S.-H. Park, O. Simeone, and S. S. Shitz, "Joint optimization of cloud and edge processing for fog radio access networks," *IEEE Trans. Wireless Commun.*, vol. 15, no. 11, pp. 7621–7632, 2016.

[18] 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, 2016.

[19] 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 2016*, 2016, pp. 1–6.

[20] R. G. Stephen and R. Zhang, "Green OFDMA resource allocation in cache-enabled CRAN," in *Proc. IEEE OnlineGreenComm 2016*, 2016, pp. 70–75.

[21] X. Wang, S. Leng, and K. Yang, "Social-aware edge caching in fog radio access networks," *IEEE Access*, 2017.

[22] T. X. Tran and D. Pompili, "Octopus: A cooperative hierarchical caching strategy for cloud radio access networks," in *Proc. IEEE MASS 2016*, Oct. 2016, pp. 154–162.

[23] J. Yao and N. Ansari, "Joint caching in fronthaul and backhaul constrained C-RAN," in *Proc. IEEE GLOBECOM 2017*, Dec. 2017.

[24] D. P. H. Kellerer, U. Pferschy, *Knapsack Problems*. Springer Berlin Heidelberg, 2004.

[25] E. Batu, M. Bennis, E. Zeydan, M. A. Kader, I. A. Karatepe, A. S. Er, and M. Debbah, "Big data meets telcos: A proactive caching perspective," *J. COMMUN. NETW.*, vol. 17, no. 6, pp. 549–557, 2015.

[26] E. Zeydan, E. Bastug, M. Bennis, M. A. Kader, I. A. Karatepe, A. S. Er, and M. Debbah, "Big data caching for networking: moving from cloud to edge," *IEEE Commun. Mag.*, vol. 54, no. 9, pp. 36–42, 2016.

[27] L. Breslau, C. Pei, F. Li, G. Phillips, and S. Shenker, "Web caching and Zipf-like distributions: evidence and implications," in *Proc. IEEE INFOCOM '99*, vol. 1, 1999, Conference Proceedings, pp. 126–134 vol.1.

[28] L. Wang, H. Wu, Z. Han, P. Zhang, and H. V. Poor, "Multi-hop cooperative caching in social IoT using matching theory," *IEEE Trans. Wireless Commun.*, vol. 17, no. 4, pp. 2127–2145, Apr. 2018.

[29] ——, "Dynamic resource matching for socially cooperative caching in iot networking," in *Proc. IEEE GLOBECOM 2017*, Dec. 2017, pp. 1–6.

[30] Z. Zhang, H. Ma, and L. Liu, "Cache-aware named-data forwarding in internet of things," in *Proc. IEEE GLOBECOM 2015*, Dec. 2015, pp. 1–6.