# Content Caching at the Wireless Network Edge: A Distributed Algorithm via Belief Propagation

Juan Liu[†], Bo Bai[‡], *Member, IEEE*, Jun Zhang[†], *Senior Member, IEEE*, and Khaled B. Letaief[†*], *Fellow, IEEE*
[†]Department of Electrical and Computer Engineering
The Hong Kong University of Science and Technology, Clear Water Bay, Hong Kong
[‡]Department of Electronic Engineering, Tsinghua University, Beijing 100084, China
[*]Hamad bin Khalifa University, Doha, Qatar
Email: eejliu@ust.hk, eebobai@tsinghua.edu.cn, eejzhang@ust.hk, eekhaled@ust.hk

***Abstract*—Caching popular contents at the edge of wireless networks has recently emerged as a promising technology to improve the quality of service for mobile users, while balancing the peak-to-average transmissions over backhaul links. In contrast to existing works, where a central coordinator is required to design the cache placement strategy, we consider a distributed caching problem which is highly relevant in dense network settings. In the considered scenario, each Base Station (BS) has a cache storage of finite capacity, and each user will be served by one or multiple BSs depending on the employed transmission scheme. A belief propagation based distributed algorithm is proposed to solve the cache placement problem, where the parallel computations are performed by individual BSs based on limited *local* information and very few messages passed between neighboring BSs. Thus, no central coordinator is required to collect the information of the whole network, which significantly saves signaling overhead. Simulation results show that the proposed low-complexity distributed algorithm can greatly reduce the average download delay by collaborative caching and transmissions.**

## I. Introduction

With the increasing demand of high-speed data traffics, especially due to Video on Demand (VoD) streaming, wireless networks are expected to provide extremely high throughput and ultra low latency services for massive mobile users. To deal with the stringent Quality of Service (QoS) requirements, the dense deployment of wireless small cells and fog style access points has attracted much attention from both academia and industry [1]–[3]. However, undesirably large latency could be induced during peak-traffic hours due to huge traffic requirement from vast users. A promising solution is to push the popular contents towards users by caching them at the edge of wireless networks, such as Base Stations (BSs) and terminal devices with computation and storage capacities. With the aid of distributed caching, the file delivery service of mobile users consists of two phases [4]: a *cache placement* phase, which determines the cache content at each BS, and a *content delivery* phase, which allows the BSs to deliver the requested files to the users over wireless channels.

This work was supported in part by the Hong Kong Research Grant Council under Grant No. 610113. This work is also supported in part by national Nature Science Foundation of China (NSFC) project under Grant No. 61401249 and 61322111, Specialized Research Fund for the Doctoral Program of Higher Education (SRFDP) under Grant 20130002120001, and National basic research program of China (973 Program) No. 2013CB336600.

Recently, many research interests have been aroused to investigate the content caching problem for wireless networks [5]–[9]. In [5], the cache placement problem was studied based on the statistics of users' file requests under the constraints of storage capacities in femtocell networks, where femtocell BSs are deployed to act as helper nodes to cache popular files. In [6], the design of optimal cache placement was pursued for wireless networks, considering the extra delay induced via backhaul links and physical-layer transmission parameters. Taking physical-layer transmission into account, the authors of [7] investigated the caching performance in terms of average delivery rate and outage probability for wireless small-cell networks, where the random deployment of cache-enabled BSs follows a Poisson point process. In [8], the throughput-outage tradeoff performance was investigated for wireless networks by exploiting clustered device caching via Device-to-Device (D2D) communications. In [9], given the caching placement, joint backhaul data assignment and beamforming was designed for the file delivery phase.

A carefully designed cache placement strategy is expected to provide flexible transmission opportunities for users with different QoS demands in the delivery phase. Usually, a coordinator is required in the placement phase to collect network-wide information (e.g. the storage capacities of different BSs, the connectivity between BSs and users), and perform calculations during algorithm design. However, coordination may not be possible in some self-organized networks, where no central controller exists. Without a central coordinator, [10] developed a decentralized caching algorithm to create simultaneous coded-simulcasting opportunities among users.

In this paper, we propose a novel distributed approach for the cache placement in the cache-enabled wireless networks. Each BS has a cache storage with finite capacity, and each user can be served by one or multiple BSs. Considering multiple candidate transmission schemes for each user, we formulate an optimization problem to minimize the average download delay subject to the BS's storage capacities, which, however, is NP-hard. To deal with this difficulty, we develop a belief propagation based distributed algorithm to perform distributed caching without a central coordinator. Based on

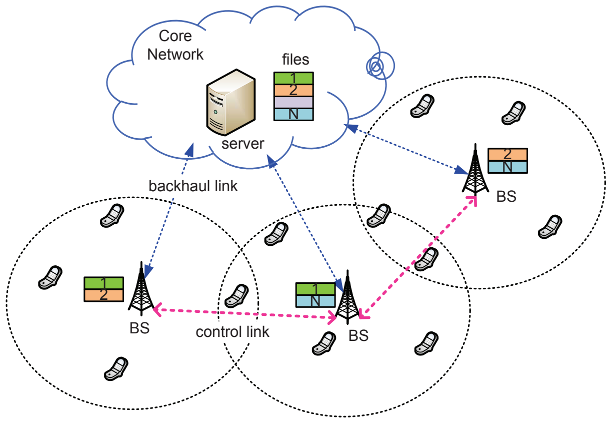


Fig. 1. An example of the considered network.

local information of its storage capacity, the users in its serving cell and their file request statistics, each BS will perform iterative computations and exchange its belief on the local caching strategy with neighboring BSs, which are collaborators of file transmissions to their common users. Through iterations, the distributed algorithm converges to a suboptimal caching solution. Simulation results show that the proposed distributed algorithm can significantly reduce the average download delay, thanks to collaborative caching and transmission. It is also shown that the performance of the proposed distributed algorithm is comparable to the centralized greedy algorithm in [5].

## II. Network Model

As shown in Fig. 1, we consider a wireless network consisting of $M$ BSs and $K$ user terminals. Let $\mathcal{A} = \{a_1, \cdots, a_M\}$ and $\mathcal{U} = \{u_1, \cdots, u_K\}$ denote the BS set and the user set, respectively. Each user can be served by one or multiple BSs, depending on the employed transmission scheme. The connectivity between the users and BSs is denoted by a $K \times M$ matrix $\boldsymbol{L}$, where each binary element $l_{km}$ indicates whether user $u_k$ can be served by BS $a_m$. That is, $l_{km} = 1$ if user $u_k$ is located in the coverage range of BS $a_m$. Otherwise, $l_{km} = 0$. The set of users in the coverage range of BS $a_m$ is denoted by $\mathcal{U}_m = \{u_k \in \mathcal{U} | l_{km} = 1\}$. Similarly, the set of service BSs of user $u_k$ is denoted by $\mathcal{A}_k = \{a_m \in \mathcal{A} | l_{km} = 1\}$.

Assume that the library of $N$ files, denoted by $\mathcal{F} = \{f_1, \cdots, f_N\}$, is stored at one or multiple content servers which could be far away in the core network. Suppose that all the files have the same size, *i.e.*, $|f_n| = |f|$ $(\forall f_n \in \mathcal{F})$. The file popularity distribution conditioned on the event that user $u_k$ makes a request is denoted by $p_{nk}$, which can be viewed as the user preference indicator and estimated via some learning procedure [11]. The user's file preferences are normalized such that $\sum_{n=1}^{N} p_{nk} = 1$. We also assume that each BS $a_m$ has a finite-capacity storage storing some of the popular files. Denote by $Q_m$ the normalized storage capacity of BS $a_m$. This means that each BS $a_m$ can store at most $Q_m$ files. Let $x_{nm}$ be a binary variable indicating whether file $f_n$ is cached at BS $a_m$. That is, $x_{nm} = 1$ if file $f_n$ is stored at the buffer of BS $a_m$, and otherwise $x_{nm} = 0$. Therefore, the matrix $\boldsymbol{X}$ consisting of the variables $\{x_{nm}\}$ $(f_n \in \mathcal{F}, a_m \in \mathcal{A})$ can be used to denote the caching strategy.

Each time when one user $u_k$ makes a request for file $f_n$, the associated BSs $\mathcal{A}_k$ can jointly decide how to transmit to this user based on the caching strategy $\boldsymbol{X}$. If file $f_n$ is cached in one BS $a_m \in \mathcal{A}_k$, this BS will transmit the file to user $u_k$ directly. If the file is cached in more than one BS, the BSs can cooperatively transmit to the user in different ways. For example, when the instantaneous channel state information is available, the BSs can transmit their cached file $f_n$ to the user with cooperative beamforming. Otherwise if the file has not been cached in any of the associate BSs $\mathcal{A}_k$, one BS will fetch the file from a content server via the backhaul link and then transmit it to the user.

## III. Distributed Cache Placement Problem

In this section, we will formulate the cache placement problem to minimize the average delay of all the users.

Let $\bar{D}_{nk}(\boldsymbol{X})$ denote the average delay for user $u_k$ to download file $f_n$ from its service BSs $\mathcal{A}_k$ or a far-away content server given any caching strategy $\boldsymbol{X}$. The cache placement problem can be formulated as follows

$$\begin{aligned} \min_{\{x_{nm}\}} \quad & \frac{1}{K} \sum_{k=1}^{K} \sum_{n=1}^{N} p_{nk} \bar{D}_{nk}(\boldsymbol{X}) \\ s.t. \quad & \begin{cases} \sum_{n=1}^{N} x_{nm} \le Q_m, \forall a_m \in \mathcal{A}, & (a) \\ x_{nm} \in \{0,1\}, \forall f_n \in \mathcal{F}, a_m \in \mathcal{A}, & (b) \end{cases} \end{aligned} \tag{1}$$

where constraint (1.a) means that each BS $a_m$ is allowed to cache at most $Q_m$ files. Since the variable $x_{nm}$ is binary, problem (1) is a constrained integer programming problem and is generally NP-hard [12]. Thus, it is challenging to find the optimal solution $\boldsymbol{X}^*$ to problem (1).

In the considered system, the serving BSs of user $u_k$ can cooperatively deliver the files to this user in different ways if they have cached the requested files, as discussed in Section II. As a matter of fact, the value of $\bar{D}_{nk}(\boldsymbol{X})$ can be obtained in an oracle way for any fixed or cooperative transmission scheme given the caching strategy $\boldsymbol{X}$. We will give some examples of transmission schemes and show how to calculate the average delay $\bar{D}_{nk}(\boldsymbol{X})$ below. Denote by $h_{km}(i)$ the channel gain between user $u_k$ and BS $a_m$ in time slot $i$. Suppose that $h_{km}(i)$ is identically and independently distributed (i.i.d.) across the time slots $i$. Notice that the users' file delivery rate, measured by the channel capacity, is highly related to a specific transmission scheme applied.

*1) Fixed transmission:* Each user $u_k$ gets file $f_n$ from a fixed BS $a_m$. The delivery rate in slot $i$ can be computed as

$$R_{nk}(\boldsymbol{X}, i) = B \log\left(1 + |h_{km}(i)|^2 l_{km} \gamma_m x_{nm}\right), \tag{2}$$

where $B$ is the system bandwidth and $\gamma_m$ is the equivalent transmit Signal-to-Noise Ratio (SNR), respectively.

*2) Cooperative beamforming:* If cooperative beamforming is applied by the associated BSs $\mathcal{A}_k$, the delivery rate of user

$u_k$ in slot $i$ is given by

$$R_{nk}(\boldsymbol{X}, i) = B \log \left( 1 + \sum_{a_m \in \mathcal{A}_k} h_{km}(i) v_{km}(i) l_{km} \gamma_m x_{nm} \right), \tag{3}$$

where $v_{km}(i)$ is the beamforming coefficient in slot $i$. In this context, the delivery rate $R_{nk}(\boldsymbol{X}, i)$, as a function of the random channel gains $h_{km}(i)$ for $m = 1, 2, \ldots, M$, is a random variable and also i.i.d. across time slots for any given physical-layer transmission scheme.

Given the caching strategy $\boldsymbol{X}$, it takes at least $T^*_{nk}(\boldsymbol{X})$ time slots for user $u_k$ to download file $f_n$, where $T^*_{nk}(\boldsymbol{X})$ satisfies

$$T^*_{nk}(\boldsymbol{X}) = \arg\min \left\{ T : \sum_{i=1}^{T} R_{nk}(\boldsymbol{X}, i) \geq \frac{|f_n|}{\Delta t} \right\}. \tag{4}$$

In (4), $\Delta t$ is the duration of one time slot and $|f_n|$ is the size of file $f_n$. Since the delivery rates $R_{nk}(\boldsymbol{X}, i)$ $(i = 1, \cdots, T)$ are i.i.d. random variables, $T^*_{nk}(\boldsymbol{X})$ given by (4) is also a random variable. By martingale theory [13], we then evaluate the average download delay in the following theorem.

**Theorem 1.** *If user $u_k$ downloads file $f_n$ from the associated BSs, the average download delay is given by*

$$\bar{D}_{nk}(\boldsymbol{X}) = \mathbb{E}_{\boldsymbol{h}}\{T^*_{nk}(\boldsymbol{X}) \cdot \Delta t\} = \frac{|f_n|}{\mathbb{E}_{\boldsymbol{h}}\left\{R_{nk}(\boldsymbol{X})\right\}}. \tag{5}$$

*where $\mathbb{E}_{\boldsymbol{h}}\{\cdot\}$ is the expectation over the channel gain $\boldsymbol{h}$. Otherwise, if user $u_k$ has to download file $f_n$ from a content server, the download delay can be estimated by a large constant delay $D^*$.*

*Proof:* Based on the definition of channel capacity, we have $R_{nk}(\boldsymbol{X}, i) \geq 0$ for $i = 1, 2, \ldots, T^*_{nk}(\boldsymbol{X})$, where $T^*_{nk}(\boldsymbol{X})$ is a stopping time given by Eq. (4). According to Wald's Equation in martingale theory [13], we have $\mathbb{E}_{\boldsymbol{h}}\left\{\sum_{i=1}^{T^*_{nk}(\boldsymbol{X})} R_{nk}(\boldsymbol{X}, i)\right\} = \mathbb{E}_{\boldsymbol{h}}\left\{T^*_{nk}(\boldsymbol{X})\right\} \cdot \mathbb{E}_{\boldsymbol{h}}\left\{R_{nk}(\boldsymbol{X})\right\} = \frac{|f_n|}{\Delta t}$. Thus, Eq. (5) is established. ■

Given any transmission scheme, we can evaluate the average download delay by (5).

## IV. Belief Propagation based Distributed Caching

To avoid heavy overhead on collecting global network information, the BSs have to learn the network information, such as the connectivity, users' file preference statistics, and candidate transmission schemes with their associated delay metrics, etc., and carry out distributed caching autonomously, relying on local interactions between BSs in the neighborhood. To this end, we propose a belief propagation (BP) based distributed algorithm to perform cooperative caching.

### A. Message Passing Procedure

First, we briefly introduce the factor graph model and the max-product algorithm. A factor graph is a bipartite graph which consists of $I$ variable nodes $\{\mu_1, \cdots, \mu_I\}$ and $J$ function nodes $\{F_1, \cdots, F_J\}$. Let $\Gamma^\mu_i$ and $\Gamma^F_j$ denote the set of indices of the neighboring function nodes of a variable node $\mu_i$ and that of the neighboring variable nodes of a function node $F_j$, respectively. Max-product is a belief propagation algorithm based on factor graph model, which is widely applied to find the optimum of the global function taking the form as $F(\boldsymbol{\mu}) = \prod_{j=1}^{J} F_j(\mu_{\Gamma^F_j})$ in a distributed manner. A comprehensive tutorial can be found in [14].

In each iteration, each variable node sends one updated message to one of its neighboring function nodes and receives one updated message from this node. Let $m^t_{\mu_i \to F_j}(x)$ denote the message from a variable node $\mu_i$ to a function node $F_j$, and $m^t_{F_j \to \mu_i}(x)$ denote the message from a function node $F_j$ to a variable node $\mu_i$, respectively. According to the max-product algorithm [14], the message $m^t_{\mu_i \to F_j}(x)$ is updated as

$$m^{t+1}_{\mu_i \to F_j}(x) = \prod_{l \in \Gamma^\mu_i \setminus \{j\}} m^t_{F_l \to \mu_i}(x), \tag{6}$$

which collects all the beliefs on the value of $\mu_i = x$ from the neighboring function nodes $F_l (l \in \Gamma^\mu_i \setminus \{j\})$ except $F_j$. The message $m^t_{F_j \to \mu_i}(x)$ is updated as

$$m^{t+1}_{F_j \to \mu_i}(x) = \max_{l \in \Gamma^F_j \setminus \{i\}} \left\{ F_j(\boldsymbol{X}) \prod_{l} m^t_{\mu_l \to F_j}(x_l) \right\}, \tag{7}$$

which achieves the maximization of the product of the local function $F_j(\boldsymbol{X})$ and incident messages over configurations in $\Gamma^F_j \setminus \{i\}$. In iteration $t$, the belief on $\mu_i = x$ is obtained as

$$b^{t+1}_i(x) = \prod_{j \in \Gamma^\mu_i} m^t_{F_j \to \mu_i}(x), \tag{8}$$

which is the product of all the messages incident to $\mu_i$.

### B. Factor Graph Model for Cache Placement

To apply the belief propagation algorithm, we transform problem (1) into an unconstrained optimization problem, and then present the factor graph model for this problem.

**Lemma 2.** *Let $\mathcal{C} = \{(f_n, u_k) | p_{nk} > 0,\, f_n \in \mathcal{F},\, u_k \in \mathcal{U}\}$ denote the set of all possible pairs of user $u_k$ and file $f_n$. The problem (1) is equivalent to the following problem*

$$\hat{\boldsymbol{X}} = \arg \max_{\boldsymbol{X} \in \{0,1\}^{NM}} \prod_{(f_n, u_k) \in \mathcal{C}} \eta_{nk}(\boldsymbol{X}) \prod_{m=1}^{M} g_m(\boldsymbol{X}), \tag{9}$$

*where $\eta_{nk}(\boldsymbol{X})$ and $g_m(\boldsymbol{X})$ are defined as*

$$\eta_{nk}(\boldsymbol{X}) = \exp\left(-p_{nk} \bar{D}_{nk}(\boldsymbol{X})\right), \tag{10}$$

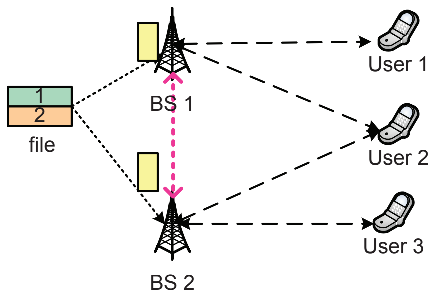
*and*

$$g_m(\boldsymbol{X}) = \begin{cases} 1, & \sum_{n=1}^{N} x_{nm} \leq Q_m, \\ 0, & \textit{otherwise}, \end{cases} \tag{11}$$

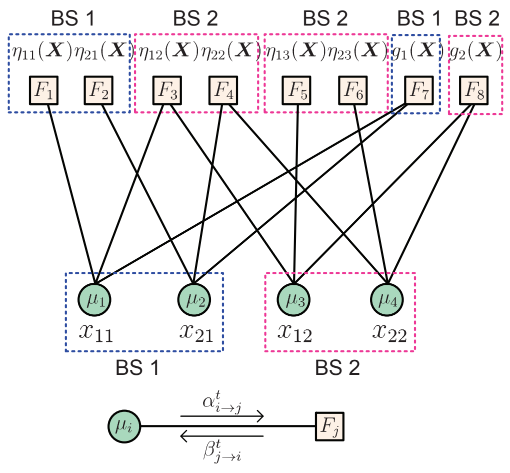
*respectively.*

*Proof:* At first, problem (1) is equivalent to maximizing $-\sum_{k=1}^{K} \sum_{n=1}^{N} p_{nk} \bar{D}_{nk}(\boldsymbol{X})$ subject to the constraints $\sum_{n=1}^{N} x_{nm} \leq Q_m$ for all $m$. Then, exponential function $\eta_{nk}(\boldsymbol{X})$ and the indicator function $g_m(\boldsymbol{X})$ are introduced to convert the equivalent optimization problem into a product form, as presented in (9). ■

In (9), $\eta_{nk}(\boldsymbol{X})$ is used to measure the delay performance when transmitting file $f_n$ to user $u_k$, and $g_m(\boldsymbol{X})$ imposes a strict constraint on the cache capacity of each BS $a_m$.



(a) Connectivity between BSs and users



(b) The factor graph model

Figure 2. An example: (a) a system with 2 BSs, 3 users, and a library of 2 files, (b) the factor graph model.

According to the optimization problem (9) and the network topology (e.g., Fig. 2(a)), we define a variable node $\mu_i$ for each element $x_{nm}$ and a function node $F_j$ for each function $\eta_{nk}(\boldsymbol{X})$ or $g_m(\boldsymbol{X})$, as shown in Fig. 2(b). We can express the mapping rule as

$$\mu_i \doteq x_{nm},\ i=(m-1)N+n, \tag{12}$$

$$F_j \doteq \begin{cases} \eta_{nk}, & j=\sum_{k'=1}^{k-1}|\mathcal{F}_{k'}|+\xi(n,k),\\ g_m, & j=\sum_{k=1}^{K}|\mathcal{F}_k|+m, \end{cases} \tag{13}$$

where $\mathcal{F}_k$ is the set of files requested by user $u_k$, defined as $\mathcal{F}_k=\{f_n|p_{nk}>0\}$, $|\mathcal{F}_k|$ denotes the number of elements in the set $\mathcal{F}_k$, and $\xi(n,k)$ denotes the index of file $f_n$ (with positive $p_{nk}$) in the set $\mathcal{F}_k$.

In the bipartite factor graph (e.g., Fig. 2(b)), each variable node $\mu_i \doteq x_{nm}$ connects to the function nodes $\{F_j\} \doteq \{\eta_{nk}\}\cup\{g_m\}$ for all $u_k\in\mathcal{U}_m$. Similarly, each function node $F_j\doteq\eta_{nk}$ connects to the variable nodes $\{\mu_i=x_{nm}\}$ for all $a_m\in\mathcal{A}_k$. Each function node $F_j\doteq g_m$ is adjacent to the variable nodes $\{\mu_i\doteq x_{nm}\}$ for all $f_n\in\mathcal{F}$. Hence, there are $I=NM$ variable nodes and $J=M+\sum_{k=1}^{K}|\mathcal{F}_k|$ function nodes in this factor graph model.

## C. Message Passing Procedure for Cache Placement

Our goal is to design a message-passing procedure which allows us to gradually approach the optimal solution to (9).

*1) Message Update :* Since all the variables $\{x_{nm}\}$ are binary, it is sufficient to pass the scalar ratio of the messages between each pair of nodes in practice. Moreover, we can express the message ratios in the logarithmic domain as

$$\alpha^t_{i\to j}=\log\left(\frac{m^t_{\mu_i\to F_j}(1)}{m^t_{\mu_i\to F_j}(0)}\right), \beta^t_{j\to i}=\log\left(\frac{m^t_{F_j\to\mu_i}(1)}{m^t_{F_j\to\mu_i}(0)}\right), \tag{14}$$

where $\alpha^t_{i\to j}$ is sent from the variable node $\mu_i$ to the function node $F_j$, and $\beta^t_{j\to i}$ is sent from the function node $F_j$ to the variable node $\mu_i$, as shown in Fig. 2(b). In this way, the product operations in (6) and (7) become simple additive operations in the logarithmic domain. And only half of the messages are actually calculated and passed. Thus, the computation complexity and communication overhead are greatly reduced. The messages $\alpha^t_{i\to j}$ and $\beta^t_{j\to i}$ also reflect the beliefs for the value of $\mu_i$ in each iteration $t$.

Then, we present the practical message passing procedure in the following theorem.

**Theorem 3.** *The message $\alpha^t_{i\to j}$ is updated as*

$$\alpha^{t+1}_{i\to j}=\sum_{l\in\Gamma^{\mu}_i\backslash\{j\}}\beta^t_{j'\to i}. \tag{15}$$

*When $F_j\doteq\eta_{nk}$, the message $\beta^{t+1}_{j\to i}$ is given by*

$$\beta^{t+1}_{j\to i}=p_{nk}\left(\bar{D}_{nk}(\boldsymbol{X}^t_{i,0})-\bar{D}_{nk}(\boldsymbol{X}^t_{i,1})\right), \tag{16}$$

*where the caching vectors $\boldsymbol{X}^t_{i,0}$ and $\boldsymbol{X}^t_{i,1}$ can be obtained by assigning their elements as*

$$x_{nm}\doteq\mu_l=\begin{cases}1, & l\in E^t_i=\{i_1\in\Gamma^F_j\backslash\{i\}|\alpha^t_{i_1\to j}>0\},\\ 0, & \textit{otherwise},\end{cases}$$

*and*

$$x_{nm}\doteq\mu_l=\begin{cases}1, & l\in E^t_i\cup\{i\},\\ 0, & \textit{otherwise},\end{cases}$$

*respectively. When $F_j\doteq g_m$, the message $\beta^t_{j\to i}$ is updated as*

$$\beta^{t+1}_{j\to i}=\min\left\{0,-\alpha^{(Q_m)}_{l\to j}(t)\right\}, \tag{17}$$

*where $\alpha^{(Q_m)}_{l\to j}(t)$ is the $Q_m$-th message among the messages $\{\alpha^t_{l\to j}\}$ ($l\in\Gamma^F_j\backslash\{i\}$) sorted in the descending order.*

*Proof:* The proof is omitted due to limited space. ■

*2) Belief Update :* Similarly, we obtain the ratio of the belief (c.f. (8)) in the logarithmic domain as

$$\tilde{b}^t_i=\log\left(\frac{b^t_i(1)}{b^t_i(0)}\right)=\sum_{j\in\Gamma^{\mu}_i}\beta^t_{j\to i}, \tag{18}$$

where $\beta^t_{j\to i}$ is given by (17) for $F_l\doteq g_m$, and by (16) for $F_j\doteq\eta_{nk}$ ($j\in\Gamma^{\mu}_i\backslash\{l\}$), respectively. As a result, the estimation of $\mu_i$ can be expressed as

$$\hat{\mu}^t_i=\begin{cases}1, & \text{if } \tilde{b}^t_i>0,\\ 0, & \text{if } \tilde{b}^t_i<0.\end{cases} \tag{19}$$

In each iteration $t$, each variable node $\mu_i$ updates its belief on its associated variable $x_{nm}$ according to (18) and makes an estimate of $x_{nm}$ according to (19) till it converges.

**Algorithm 1** BP based distributed cache placement algorithm

```
1: Map $\eta_{nk}$, $g_m$ to $F_j$ and $x_{nm}$ to $\mu_i$ for $\forall n, k, m$,
2: Set $t = 0$ and $\alpha^t_{i\to j} = \beta^t_{j\to i} = 0$, $\forall i, j$,
3: Set $t_{max}$ as a sufficiently large constant.
4: while Not convergent and $t \le t_{max}$ do
5:     for $m = 1 : M$ do
6:         for $n = 1 : N$ do
7:             Calculate $\alpha^t_{i\to j}$ by (15);
8:             for $k \in \widetilde{\mathcal{U}}_m$ do
9:                 Calculate $\beta^t_{j\to i}$ by (16) for $F_j \doteq \eta_{nk}$;
10:            end for
11:        end for
12:        Calculate $\beta^t_{j\to i}$ by (17) for $F_j \doteq g_m$;
13:        Calculate the belief $\tilde{b}^t_i$ by (18);
14:        Estimate the variable $\hat{\mu}_i$ by (19);
15:    end for
16:    Check the convergence, and set $t = t + 1$;
17: end while
18: Obtain the optimal estimate $\hat{\boldsymbol{X}}$ to the solution of (9).
```

### *D. Distributed Algorithm Design*

When we map the message passing procedure derived on the factor graph (e.g., Fig. 2(b)) back to the original network graph (e.g., Fig. 2(a)), we notice that all the messages are updated at the BSs and only some of them will be exchanged between neighboring BSs.

*1) Scenario I:* When user $u_k$ is connected to one single BS $a_m$, as shown in Fig. 2(b), the update of messages $\alpha^t_{i\to j}$ and $\beta^t_{j\to i}$ is performed at this BS for the variable nodes $\mu_i \doteq x_{nm}$, the function nodes $F_j \doteq \eta_{nk}$, and $F_j \doteq g_m$. In this case, each BS $a_m$ performs the message calculation and belief update for all the users within its coverage, *i.e.*, $u_k \in \mathcal{U}_m$.

*2) Scenario II:* When user $u_k$ is connected to multiple BSs $\mathcal{A}_k$, the update of messages $\alpha^t_{i\to j}$ and $\beta^t_{j\to i}$ associated with the function node $F_j \doteq \eta_{nk}$ is performed at one BS $a_m$ and will be exchanged between the service BSs of this user $\mathcal{A}_k$ over control links, as shown in Fig. 1.

Notice that message exchanges just take place in Scenario II. Thus, the communication overhead induced in this scenario depends on the number of common users covered by multiple BSs. From the above discussion, we summarize the message passing based distributed caching algorithm in Algorithm 1. In this algorithm, the message update for each user should be performed just once by one single BS in each iteration. To avoid confusion, $\widetilde{\mathcal{U}}_m$ is used to denote the set of users whose messages are processed by BS $a_m$ in Algorithm 1.

## V. Simulation Results

In this section, we present simulation results to demonstrate the performance of the proposed caching algorithm. We consider a small-cell or fog style networking, where each of the $M$ circular cells has a radius of 150m and the distance between neighboring BSs is set as 200m. $K$ users are uniformly and independently distributed in the area covered by the $M$ cells. The users' file requests follow the Zipf distribution with parameter $\gamma_k = 0.3 + \frac{k}{K}(3 - 0.3)$, i.e., we assume different users have different request distributions. The users in the middle of a cell are served by just one single BS, while the users in the overlapping area of cells are covered by multiple BSs and thus cooperative transmission may be enabled. The connectivity between BSs and users is thus established. Suppose that the system bandwidth is 5MHz, and the length of each slot is 20ms. The file size is set to be 100Mbits. The path-loss exponent is set as 3.5. The small-scale channel gain $|h_{km}|^2$ follows independently standard exponential distribution in each slot. Assuming that no inter-cell interference is induced by adopting appropriate scheduling policies, the transmit power is set to make sure that the average received SNR at the cell edge is equal to 0dB. We also set $K = 100$, $M = 10$, $N = 100$, and $D^* = 40$s.

In Fig. 3, we compare the average download delay of the distributed algorithm with that of the centralized greedy algorithm in [5] by applying cooperative beamforming and fixed transmissions, respectively. For performance comparison, we also demonstrate the popular caching algorithm as a baseline, which caches the most popular files based on the statistical preference of the users in the whole network. Considering different file preferences, the users request files with randomly permuted probability $\frac{n^{-\gamma_k}}{\sum_{n=1}^{N} n^{-\gamma_k}}$ over $n = 1, \cdots, N$. From Fig. 3, the average download delay monotonically decreases with the increase of the cache capacity $Q_m = Q$. This is due to the fact that with the increase of storage capacity, more files are cached in each BS and more users can download files from local BSs instead of the content server. Accordingly, the average download delay is greatly reduced.

One can also see that the average delay performance is significantly improved, when cooperative transmission instead of fixed transmission is used. And the performance gap between cooperative transmission based caching and fixed transmission based caching becomes larger as the cache capacity increases, since more files can be cached to facilitate cooperative transmission for cell-edge users. In the scenario of either cooperative or fixed transmission, cooperative caching performs much better than popularity-based caching, except the case when all the files are cached in each BS with $Q = N$. This is because that cooperative caching is performed based on the more accurate estimation of the diverse file preferences of individual users. While the popularity-based caching algorithm performs caching based on the statistical preference of a very large number of users, which could not reflect the file preferences of individual users.

Observed from Fig. 3, the proposed belief propagation based algorithm can achieve a nearly identical delay performance as compared to the centralized greedy algorithm which gives a performance guarantee [5], *i.e.*, $\frac{1}{2}$ of the optimal value. It has a slightly larger delay performance in the small-capacity region when $Q$ is less than 40, and achieves almost the same performance as the greedy algorithm in other scenarios.

We plot in Fig. 4 the iterative procedure of the belief propagation based algorithm for different storage capacities $Q_m = Q$, when cooperative transmission is adopted. It is observed that the average delay starts from an initial value, and gradually converges to an appropriate solution through a few iterations up to hundreds of iterations, depending on the

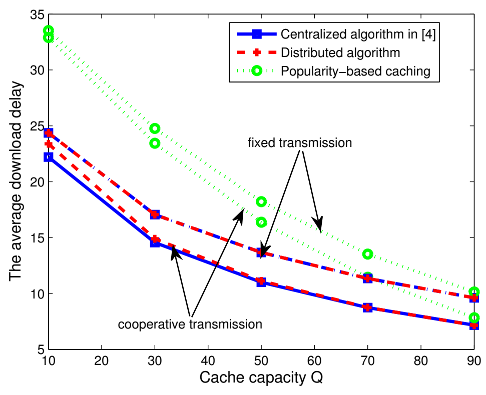


Figure 3. The average download delay vs. cache capacity $Q$.

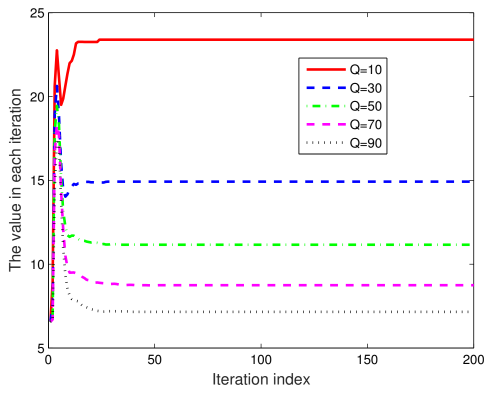


Figure 4. The iterative procedure of the proposed distributed algorithm.

system parameters.

In Fig. 5, we demonstrate the computational complexity of the proposed algorithm and the centralized greedy algorithm, when cooperative transmission is adopted. Here, we measure the computational complexity by the number of calculations used in the algorithm. It is observed that the total number of calculations of the proposed distributed algorithm required by all the BSs is much smaller than that of the greedy algorithm. By sharing computing tasks, each individual BS does much fewer calculations when running the distributed algorithm. Moreover, with the increase of cache capacity, more elements are added greedily when the greedy algorithm is adopted. Hence, the cache capacity has a great impact on the computational complexity when running the greedy algorithm. When applying the distributed algorithm, the cache capacity is a parameter which only adjusts the value of the messages during iterations. It does not change the factor graph model, and hence may not cause a significant impact on its computational complexity.

## VI. Conclusions

In this work, we studied the cache placement problem for small-cell wireless networks, considering different physical-layer transmission schemes for each user. To avoid heavy overhead in collecting global network information, we proposed a belief propagation based algorithm to place files at

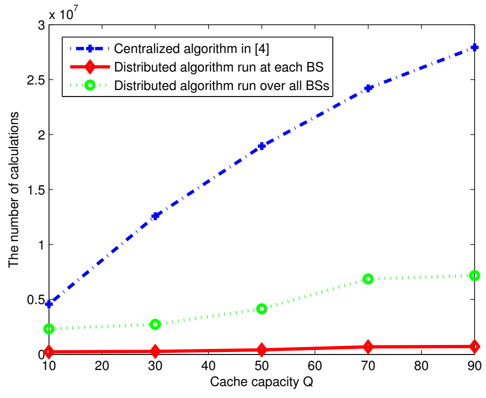


Figure 5. The number of calculations vs. cache capacity $Q$.

BSs in a distributed way. By collecting local information of storage capacities, the users' request statistics and candidate transmission schemes, each BS will run computations and exchange very few messages with its neighboring BSs iteratively till convergence. Simulation results demonstrated that the proposed simple distributed algorithm can significantly improve the file delivery performance by collabrative caching and transmissions.

## References

[1] J. Andrews, “Seven ways that HetNets are a cellular paradigm shift,” *IEEE Commun. Mag.*, vol. 51, no. 3, pp. 136–144, Mar. 2013.
[2] Y. Shi, J. Zhang, K. B. Letaief, B. Bai, and W. Chen, “Large-scale convex optimization for ultra-dense cloud-RAN,” *IEEE Wireless Communications*, vol. 22, no. 3, pp. 84–91, Jun. 2015.
[3] C. Li, J. Zhang, and K. B. Letaief, "Throughput and energy efficiency analysis of small cell networks with multi-antenna base stations," *IEEE Trans. Wireless Commun.*, vol. 13, no. 5, pp. 2502-2517, May 2014.
[4] M. Maddah-Ali and U. Niesen, “Fundamental limits of caching,” *IEEE Trans. Inform. Theory*, vol. 60, no. 5, pp. 2856–2867, May 2014.
[5] 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. 1107–1115.
[6] X. Peng, J.-C. Shen, J. Zhang, and K. B. Letaief, “Backhaul-aware caching placement for wireless networks,” in *Proc. IEEE Globecom*, Diego, CA, Dec. 2015.
[7] E. Baştuğ, M. Bennis, M. Kountouris, and M. Debbah, “Cache-enabled small cell networks: Modeling and tradeoffs,” *EURASIP J. Wirel. Commun. Netw.*, vol. 2015, no. 1, p. 41, Feb. 2015.
[8] M. Ji, G. Caire, and A. Molisch, “Wireless device-to-device caching networks: Basic principles and system performance,” *IEEE J. Sel. Areas Commun.*, vol. PP, no. 99, pp. 1–1, 2015.
[9] X. Peng, J.-C. Shen, J. Zhang, and K. B. Letaief, "Joint data assignment and beamforming for backhaul limited caching networks," in *Proc. IEEE PIMRC*, Washington, DC, Sept. 2014.
[10] M. Maddah-Ali and U. Niesen, “Decentralized coded caching attains order-optimal memory-rate tradeoff,” *IEEE/ACM Trans. Networking*, vol. 23, no. 4, pp. 1029–1040, Aug. 2015.
[11] B. N. Bharath, K. G. Nagananda, and H. V. Poor, “A learning-based approach to caching in heterogenous small cell networks,” Aug. 2015. [Online]. Available: http://arxiv.org/abs/1508.03517
[12] K. Shanmugam, N. Golrezaei, A. Dimakis, A. Molisch, and G. Caire, “FemtoCaching: Wireless content delivery through distributed caching helpers,” *IEEE Trans. Inform. Theory*, vol. 59, no. 12, pp. 8402–8413, Dec. 2013.
[13] David Williams, *Probability with Martingale*. Cambridge University Press, 1991.
[14] F. Kschischang, B. Frey, and H.-A. Loeliger, “Factor graphs and the sum-product algorithm,” *IEEE Trans. Inform. Theory*, vol. 47, no. 2, pp. 498–519, Feb. 2001.