Fast Multi-Channel Gibbs-Sampling for Low-Overhead Distributed Resource Allocation in OFDMA Cellular Networks

Po-Kai Huang[1], Xiaojun Lin[1], Ness B. Shroff[2], and David J. Love[1]

Abstract— **A major challenge in OFDMA cellular networks is to efficiently allocate scarce channel resources and optimize global system performance. Specifically, the allocation problem across cells/base-stations is known to incur extremely high computational/communication complexity. Recently, Gibbs sampling has been used to solve the downlink inter-cell allocation problem with distributed algorithms that incur low computational complexity in each iteration. In a typical Gibbs sampling algorithm, in order to determine whether to transit to a new state, one needs to know in advance the performance value after the transition, even before such transition takes place. For OFDMA networks with many channels, such computation of future performance values leads to a challenging tradeoff between convergence speed and overhead: the algorithm either updates a very small number of channels at an iteration, which leads to slow convergence, or incurs high computation and communication overhead. In this paper, we propose a new multi-channel Gibbs sampling algorithm that resolves this tradeoff. The key idea is to utilize perturbation analysis so that each base-station can accurately predict the future performance values. As a result, the proposed algorithm can quickly update many channels in every iteration without incurring excessive computation and communication overhead. Simulation results show that our algorithm converges quickly and achieves system utility that is close to the existing Gibbs sampling algorithm.**

I. Introduction

In order to accommodate the exponential growth of data traffic in mobile wireless networks [1], current and future cellular systems will increasingly rely on high-density small-cells (such as pico- and femto-cells) to significantly increase the traffic-carrying capacity [2]. However, the proliferation of small-cells leads to a challenging problem of how to manage inter-cell interference. First, since the placement of small cells often faces significant constraints, the resulting network topology becomes highly irregular. Second, the traffic density across cells can be highly non-uniform and may exhibit time-varying patterns. As a result, traditional resource planning strategies based on *static* and *regular* reuse patterns are no longer adequate [3]. There is thus a pressing need to develop adaptive and even distributed resource-allocation mechanisms that can dynamically adapt to irregular topology and non-uniform traffic patterns to best utilize the limited spectrum resources.

This work has been partially supported by the NSF through grants CNS-0643145, CNS-1012700 and a MURI grant from the Army Research Office W911NF-08-1-0238.

[1]P.-K. Huang, X. Lin, and D. J. Love are with the School of ECE, Purdue University, West Lafayette, IN 47907 USA (E-mail: huang113@purdue.edu, linx@purdue.edu, djlove@ecn.purdue.edu)

[2]N. B. Shroff is with the Ohio State University, Columbus, OH 43210 USA (E-mail: shroff@ece.osu.edu)

In this paper, we focus on the downlink of OFDMA (Orthogonal Frequency-Division Multiple Access) systems (as in 4G LTE [4]), and our goal is to design such an adaptive and distributed algorithm for allocating frequency and time resources across cells and users under irregular topology and non-uniform load. Note that such a resource allocation problem can be decomposed into two parts. Within a cell, based on the set of frequency channels allocated to the cell and to its neighboring cells, the base-station can decide which user to serve over each channel at each time. This *intra-cell* resource allocation problem can be cast as a convex optimization problem, which can be easily solved [5,6]. Then, across cells, the *inter-cell* resource allocation problem decides which set of channels should be allocated to each cell. Due to the inherent non-convex nature of wireless interference, this inter-cell problem incurs extremely high complexity, and existing studies mostly focus on solutions without optimality or efficiency guarantees [7,8].

Recently, a class of randomized algorithms based on Gibbs sampling has been used to solve the downlink inter-cell resource allocation problem in OFDMA cellular networks [8,9]. At a high level, in each iteration of a Gibbs-sampling algorithm, a subset of base-stations distributively decide how they will adjust their channel allocation. Each base-station in the set will evaluate all possible adjustments of channel allocation and learn how these adjustments will affect the performance (which is often modeled as "utility" values [8,9]) of itself and its neighboring cells. Then, the base-station chooses one of the "proposed" adjustments with a certain probability. This probability is carefully designed so that the entire system follows a reversible Markov chain and has high probability to move towards the globally-optimal channel allocation. In theory, a Gibbs-sampling algorithm can be used to develop distributed solutions to any non-convex optimization problems with a suitable structure (although the convergence time to the global optimal solution may still be large [8,10]).

However, when applied to OFDMA cellular networks with a large number of frequency channels, the above generic version of Gibbs sampling leads to a difficult tradeoff between convergence speed and computation/communication overhead. Note that the allocation decision of every channel can potentially be adjusted to improve the global performance. Thus, to expedite convergence, each base-station should preferably evaluate a large set of "proposed" adjustments across multiple channels. However, each "proposed" adjustment affects its own cell and the neighboring cells in different ways. Hence, it will then incur significant computation/communication overhead if each base-station has to calculate the potential increase/decrease of the utility values at the affected cells *for every possible adjustment*.

In this paper, we propose a fast multi-channel Gibbs sampling algorithm that addresses this difficulty. In our proposed algorithm, each base-station can evaluate the "proposed" adjustments covering multiple channels without incurring excessive computation and communication overhead. If we view the utility value under each adjustment as the solution to an optimization problem, then the key idea behind our proposed algorithm is to view each adjustment as a perturbation to a *common* optimization problem. We can then quickly and accurately estimate the perturbed utility values for each base-station without actually solving additional optimization problems. We derive conditions under which the transition probabilities of the resulting Markov chain approach those of the original Markov chain under standard Gibbs sampling. Our simulation results confirm that the proposed algorithm achieves both fast convergence and low computation/communication overhead. Further, it can dynamically adjust the global channel allocations to adapt to different network conditions and load patterns.

There have been a number of recent studies that use Gibbs sampling to design distributed algorithms for the complex optimization problems in wireless networks [8,9,11]–[16]. Under a graph-based interference model, Gibbs sampling has been used to study link scheduling [12]–[14], channel selection and user association [11]. However, the interference model used in these studies does not account for SINR and is impractical for cellular systems. With a more accurate SINR-based interference model, recent studies use Gibbs sampling to study scheduling and power control [15,17,18], joint power control and user association [16], channel allocation [9], and joint power-control/user-association/channel-allocation [8]. Our work differs from these studies by utilizing a new method to estimate the perturbed utility values across multiple channels and hence can achieve fast convergence with low computation/communication overhead. Our work is also closely related to [7]. However, the latter uses a greedy approach for inter-cell channel allocation and power control, and thus the resulting solution does not possess global optimality. Finally, for a comprehensive review of other existing algorithms for resource allocation problem in wireless networks and their relationship to Gibbs sampling, we refer the readers to [8].

The rest of the paper is organized as follows. The system model is presented in Section II. In Section III, we provide an overview of the Gibbs sampling and explain the difficulty in existing Gibbs-sampling algorithms. We propose and analyze the fast Gibbs sampling algorithm in Section IV and present the simulation results in Section V. Then, we conclude.

II. System Model

Consider a cellular system using OFDMA (as in LTE [4]). There are J base-stations sharing K channels. Each channel has a bandwidth $W_c = W_T/K$, where W_T is the total available bandwidth in the system. Assume that time is slotted[1]. Let P_j be the maximum power that a base-station j can apply over all channels. We do not consider power control in this paper. Hence, we assume that the power that a base-station j can apply on a channel is P_j/K. We assume that each mobile user is associated with a unique base-station. For each base-station j, let $\mathcal{S}_j$ be the set of $|\mathcal{S}_j|$ users associated with it. Correspondingly, for a user i, let $A(i)$ be the base-station that user i is associated with, i.e., $i \in \mathcal{S}_{A(i)}$. Let G_{ij} be the channel gain from base-station j to user i, which captures channel attenuation due to path loss and slowly-varying shadow fading.

In this paper, we will focus on the resource allocation problem for downlink transmission. This problem can be viewed in two parts: the intra-cell control problem and the inter-cell control problem. The intra-cell control determines user scheduling within a base-station, while the inter-cell control determines the channel allocation across base-stations. Since inter-cell control incurs a higher overhead, our goal is to find one fixed inter-cell channel-allocation pattern based on a given system setting. Then, based on the fixed channel allocation, each base-station can schedule different users across channels and time-slots. In practice, this decomposition means that the inter-cell channel allocation is updated at a slower time-scale than intra-cell user scheduling. Specifically, define the indicator variable $I_{jk} = 1$ if base-station j uses channel k, and $I_{jk} = 0$, otherwise. Let $\vec{I} = [I_{jk}]$ represent the global channel-allocation vector across all base-stations and all channels. For a user i associated with base-station $A(i)$, the rate r_{ik} that user i will receive on channel k (assuming $I_{A(i),k} = 1$) is a function of the signal-to-interference-plus-noise ratio (SINR). Without loss of generality, we use the Shannon formula [19], i.e.,

$$\begin{aligned} r'_{ik} &= W_c \log_2\left(1 + \frac{P_{A(i)} I_{A(i),k} G_{i,A(i)}/K}{W_T N_0/K + \sum_{j\neq A(i)} P_j I_{jk} G_{ij}/K}\right) \\ &= W_c \log_2\left(1 + \frac{P_{A(i)} I_{A(i),k} G_{i,A(i)}}{W_T N_0 + \sum_{j\neq A(i)} P_j I_{jk} G_{ij}}\right), \end{aligned} \tag{1}$$

where N_0 is the thermal noise density. Note that Equation (1) accounts for the interference from all base-stations. In reality, the interference between two distant base-stations will be small. Hence, in practice it is reasonable to approximate (1) by only considering the interference from neighboring base-stations. Specifically, let $\mathcal{N}_j$ denote the set of neighboring base-stations[2] for base-station j. Let $\eta^{\max}_{i,A(i)} = \sum_{\{z\neq A(i), z\notin \mathcal{N}_{A(i)}\}} P_z G_{iz}$. We can then approximate (1) by

$$r_{ik} = W_c \log_2\left(1 + \frac{P_{A(i)} I_{A(i),k} G_{i,A(i)}}{W_T N_0 + \eta^{\max}_{i,A(i)} + \sum_{\{j\in \mathcal{N}_{A(i)}\}} P_j I_{jk} G_{ij}}\right). \tag{2}$$

Note that $r_{ik} \le r'_{ik}$. Further, since $\eta^{\max}_{ij}$ is independent of $\vec{I}$, the value of r_{ik} only depends on the channel allocation of the neighboring base-stations of $A(i)$. Later, we will see that this property is critical for developing a distributed algorithm. Hence, in the rest of the paper, we will use (2) to model the

[1]In LTE, each channel in a time slot represents a resource block that can be assigned by each base-station to one associated user. [4]

[2]For example, $\mathcal{N}_j$ could be the base-stations within a certain distance from j. However, our result does not depend on a specific definition of $\mathcal{N}_j$.

rate of each user. Further, we assume that the neighboring relationship is symmetric, i.e., if $h \in \mathcal{N}_j$, then $j \in \mathcal{N}_h$.

For intra-cell control, we assume that at any given time-slot and for a given channel, the base-station can only serve one associated user. However, a base-station can use the same channel to serve different users across time slots, i.e, time-division multiplexing[3]. Let ϕ_{ik} be the fraction of time that base-station $A(i)$ serves user i on channel k. Note that we must have $\sum_{i \in \mathcal{S}_j} \phi_{ik} \leq 1$ for each base-station j and each channel k. The average rate R_i received by user i is given by $R_i = \sum_{k=1}^{K} \phi_{ik} r_{ik}$. We associate a utility function $U_i(R_i)$ for user i, which captures the satisfaction level of user i when its service rate is R_i. Recall from (2) that the rate r_{ik} of each user i is determined by the channel-allocation vector $\vec{I}$. With a fixed vector $\vec{I}$, the intra-cell control for base-station j can then be modeled as the following optimization problem:

$$\begin{aligned} &\max_{[\phi_{ik}] \geq 0} && \sum_{i \in \mathcal{S}_j} U_i(R_i), \\ &\text{subject to} && R_i = \textstyle\sum_{k=1}^{K} \phi_{ik} r_{ik}, \text{ and } \sum_{i \in \mathcal{S}_j} \phi_{ik} = 1. \end{aligned} \tag{3}$$

Problem (3) is a convex optimization problem and can be solved by standard technique [5]. Let $V_j(\vec{I})$ denote the optimal objective value of problem (3) for base-station j. Then, the inter-cell channel-allocation problem can be written as:

$$\max_{\vec{I} = [I_{jk}] \in \mathcal{I}} \quad \sum_{j=1}^{J} V_j(\vec{I}), \tag{4}$$

where $\mathcal{I}$ is the set of all channel-allocation vectors.

Unfortunately, Problem (4) is a combinatorial problem that is in general very difficult [8]. In particular, since the number of all channel-allocation vectors (2^{JK}) grows exponentially with the number of base-stations and the number of channels, even a centralized solution will incur extremely high complexity[4]. In the next section, we will introduce distributed and low-complexity algorithms based on Gibbs sampling.

III. Gibbs Sampling

In this section, we provide an overview of distributed algorithms based on Gibbs sampling, which can be used to solving Problem (4). We then review several types of existing Gibbs-sampling algorithms and explain their performance tradeoffs in terms of the convergence speed and the computation/communication overhead. This discussion motivates the new algorithm that we will propose in Section IV.

A. Overview of Gibbs Sampling

Suppose that the channel-allocation vector $\vec{I}$ is updated in an iterative manner, and let $\vec{I}_t$ be the vector at the t-th iteration. In Gibbs sampling (and many other randomized algorithms), one forms a Markov chain with $\vec{I}_t$ as the state such that the stationary distribution of the chain has the form of the Gibbs distribution:

$$P(\vec{I}_t = \vec{I}) \triangleq \pi(\vec{I}) = \frac{1}{Z_T} e^{\frac{1}{T}\sum_{j=1}^{J} V_j(\vec{I})}, \tag{5}$$

where T is called the temperature of the Gibbs distribution [20, Chapter 7], and Z_T is a normalization constant. Clearly, when T is small, the channel-allocation vector(s) $\vec{I}$ with the optimal utility will be reached with probability close to 1. Hence, such a Markov chain can be used to iteratively find the optimal $\vec{I}$.

To form such a Markov chain, Gibbs sampling sets the transition probabilities in a particular way [20, Chapter 7, Section 6.1]. We start with a few notations. Let $A = \{(j,k) | 1 \leq j \leq J, 1 \leq k \leq K\}$ denote the set of all base-station/channel pairs, and let $H \subset A$ be a subset for which the network considers to update the channel allocation in a given iteration. Recall that $\vec{I}$ is a vector of indicator variables, each of which corresponds to an element in A. We further use $\vec{I}(H)$ and $\vec{I}(A \backslash H)$ to denote those components of $\vec{I}$ that correspond to elements in H and $A \backslash H$, respectively. Similar to the vector $\vec{I}$, let $\vec{x}$ and $\vec{y}$ be the indicator vectors before and after a given iteration, respectively. Since only those components corresponding to H will be changed, we must have $\vec{x}(A \backslash H) = \vec{y}(A \backslash H)$. Further, for any $\vec{x}$ and $\vec{y}$, we use $(\vec{y}(H), \vec{x}(A \backslash H))$ to denote another indicator vector whose components corresponding to H and $A \backslash H$ are taken from $\vec{y}$ and $\vec{x}$, respectively.

At each iteration t, a set H is chosen with probability p_H, and only the elements in H will be updated. The requirement on the probability distribution $[p_H, H \subset A]$ is that:

> Every base-station/channel pair (j,k) must belong to at least one $H \subset A$ such that $p_H > 0$. (6)

Hence, every (j,k) has a non-zero probability of being chosen for update. Let $\mathcal{I}(H)$ be the set of all possible values of $\vec{I}(H)$. Based on the channel-allocation vector $\vec{I}_t = \vec{x}$, Gibbs sampling uses the following transition probability:

$$\begin{aligned} & P(\vec{I}_{t+1} = \vec{y} | \vec{I}_t = \vec{x}) \\ = \; & \textstyle\sum_{\{H : \vec{y}(A \backslash H) = \vec{x}(A \backslash H)\}} p_H \; \pi(\vec{y}(H) | \vec{x}(A \backslash H)), \end{aligned} \tag{7}$$

where

$$\pi(\vec{y}(H) | \vec{x}(A \backslash H)) = \frac{e^{\frac{1}{T}\sum_{j=1}^{J} V_j(\vec{y}(H), \vec{x}(A \backslash H))}}{\sum_{\{\vec{z} \in \mathcal{I}(H)\}} e^{\frac{1}{T}\sum_{j=1}^{J} V_j(\vec{z}, \vec{x}(A \backslash H))}}. \tag{8}$$

In other words, if H is chosen, then update $\vec{x}(H)$ to $\vec{y}(H)$ with probability given by (8). Note that (8) is simply the conditional probability for $\vec{I}(H) = \vec{y}(H)$, given that $\vec{I}(A \backslash H) = \vec{x}(A \backslash H)$, when $\vec{I}$ follows the distribution in (5)[5]. It can be easily verified that the resulting Markov chain is aperiodic and irreducible. Further, it is reversible (satisfying the detailed balanced equations), and its stationary distribution is exactly (5).

[3]This is standard in LTE system.

[4]In a LTE system with 20MHz of total bandwidth, the number of channels is around 100. If we consider a 19 cells (two rings) layout, the number of channel-allocation vectors is 2^{1900}!

[5]Note that the normalization constant Z_T does not appear in (8). This is very useful because calculating Z_T would have incurred exponential complexity.

However, in general the transition probability in (7) depends on the update at all base-stations. Hence, this procedure requires centralized implementation. In order to derive a distributed algorithm, we exploit the fact that the value of $V_j(\vec{I})$ only depends on the channel allocation at base-station j and the base-stations in $\mathcal{N}_j$ (see (2)). In other words, let $A_j = \{(h,k) | h = j \text{ or } h \in \mathcal{N}_j, k = 1, ..., K\}$. Then, we can write $V_j(\vec{I}) = V_j(\vec{I}(A_j))$. We then use the following distributed procedure. At each iteration, first choose a set of base-stations $\mathcal{D}$[6] without common neighbors, i.e.,

$$\text{For any } j, j' \in \mathcal{D},\ \mathcal{N}_j \cap \mathcal{N}_{j'} = \emptyset. \tag{9}$$

Then, each base-station $j \in \mathcal{D}$ chooses a subset of channels $H_j \subset \{(j,k)|k = 1, ..., K\}$ to update. For $j \notin \mathcal{D}$, we let $H_j = \emptyset$. Let $H = \bigcup_{j=1}^{J} H_j$. Note that by this procedure, the value of $V_j(\cdot)$ in (8) can only change either because base-station j itself updates its own channels H_j (if $j \in \mathcal{D}$) or because at most *one* neighboring base-station $h \in \mathcal{N}_j$ updates its channels (if $j \in \mathcal{N}_h$ and $h \in \mathcal{D}$). We can then show that the expression (8) can be decoupled as follows:

$$\pi(\vec{y}(H)|\vec{x}(A\backslash H)) = \prod_{j\in\mathcal{D}} \pi(\vec{y}(H_j)|\vec{x}(A\backslash H_j)), \tag{10}$$

where $\pi(\vec{y}(H_j)|\vec{x}(A\backslash H_j))$ is equal to

$$\frac{e^{\frac{1}{T}\left[V_j(\vec{y}(H_j),\vec{x}(A_j\backslash H_j)) + \sum_{\{h\in\mathcal{N}_j\}} V_h(\vec{y}(H_j),\vec{x}(A_h\backslash H_j))\right]}}{\sum_{\{\vec{z}_j\in\mathcal{I}(H_j)\}} e^{\frac{1}{T}\left[V_j(\vec{z}_j,\vec{x}(A_j\backslash H_j)) + \sum_{\{h\in\mathcal{N}_j\}} V_h(\vec{z}_j,\vec{x}(A_h\backslash H_j))\right]}}. \tag{11}$$

(Details are available in our online technical report [21].) Note that for a base-station $j \in \mathcal{D}$, expression (11) does not depend on the updates at other base-stations in $\mathcal{D}$. Hence, each base-station j can do the update independently, i.e., base-station j uses the transition probability (11) to decide which new channel-allocation decision $\vec{y}(H_j)$ it will pick.

Finally, if each base-station j only updates one channel, i.e., $H_j = \{(j,k)\}$ for some $k \in 1, ..., K$, then $\vec{z}_j$ only contains one element z_{jk} that can take either 0 or 1. Hence, the expression (11) can be further simplified to:

$$\pi(\vec{y}(H_j)|\vec{x}(A\backslash H_j)) = \frac{\exp(y_{jk}\Delta_{jk})}{1+\exp(\Delta_{jk})}, \tag{12}$$

and the value of Δ_{jk} is given by

$$\begin{aligned}\Delta_{jk} &= \tfrac{1}{T}\left[V_j(1,\vec{x}(A_j\backslash H_j)) - V_j(0,\vec{x}(A_j\backslash H_j))\right] \\ &+ \tfrac{1}{T}\textstyle\sum_{\{h\in\mathcal{N}_j\}}\left[V_h(1,\vec{x}(A_h\backslash H_j)) - V_h(0,\vec{x}(A_h\backslash H_j))\right],\end{aligned}$$

where the first argument of $V_j(\cdot,\cdot)$ and $V_h(\cdot,\cdot)$ corresponds to the element $H_j = \{(j,k)\}$.

B. *Existing Multi-Channel Gibbs Sampling Algorithms*

The approach described in Section III-A immediately leads to distributed implementation. However, when the number of channels to be updated is large, this procedure can still lead to high computation/communication overhead. To see this, let $\mathcal{I}(H_j)$ denote the set of possible values of $\vec{I}(H_j)$ that base-station j may transit to. Then, in order to compute the transition probability in (11), each base-station $h \in \{j \bigcup \mathcal{N}_j\}$ needs to compute the value of $V_h(\vec{z}_j, \vec{x}(A_h\backslash H_j))$ for every possible value of $\vec{z}_j \in \mathcal{I}(H_j)$. Further, if $h \in \mathcal{N}_j$, the result also needs to be sent to base-station j. Recall that each computation of $V_j(\cdot)$ requires solving the optimization problem (3). Hence, depending on the different implementation options outlined below, there will be a challenging tradeoff between computation/communication overhead and convergence speed.

Option 1 (Single-channel update): In each iteration, each base-station $j \in \mathcal{D}$ only updates one channel, i.e., $|H_j| = 1$. Hence, $|\mathcal{I}(H_j)| = 2$, and the computation/communication overhead is the lowest. However, if K is large, updating one channel at a time may lead to slow convergence, as can be observed from our simulation results in Section V and in [8].

Option 2 (Multi-channel update): In each iteration, each base-station $j \in \mathcal{D}$ can update multiple channels, i.e., $|H_j| > 1$. The convergence speed will become faster as we increase $|H_j|$. However, $|\mathcal{I}(H_j)| = 2^{|H_j|}$. Hence, the computation overhead increases exponentially with $|H_j|$.

Option 3 (Sequential update): This option can be conceptually viewed as a hybrid between Option 1 and Option 2. In each iteration, like Option 2, each base-station $j \in \mathcal{D}$ can update multiple channels, i.e., $|H_j| > 1$. However, base-station j updates these channels sequentially. Specifically, base-station j chooses uniformly at random a permutation $(v_1, v_2, \cdots, v_{|H_j|})$ of the elements in H_j, i.e., each permutation is chosen with probability $1/|H_j|!$. Then, the iteration is divided into $|H_j|$ rounds, and only channel v_l is updated in each round $l = 1, ..., |H_j|$ as in Option 1. It can be shown that the resulting Markov chain still has the same stationary distribution as (5) [14]. Note that the computation overhead now increases linearly with $|H_j|$. However, after each round, the updated channel allocation must be communicated to the neighboring base-stations so that future computation of Problem (3) can use the most-recently updated channel allocation. As a result, more rounds lead to more control messages exchanged between base-stations and slower speed.

In sum, when K is large, all options either suffer slow convergence or high computation/communication overhead.

IV. Fast Gibbs Sampling Algorithms

In this section, we propose a fast Gibbs sampling algorithm with only *one round* of communication, low computation overhead, and *multi-channel* update in each iteration. We first sketch the basic skeleton of this algorithm below.

Fast Multi-Channel Gibbs Sampling: In each iteration,

1) Each base-station $j = 1, ..., J$ solves Problem (3) based on the existing channel-allocation decisions.
2) Randomly choose a subset $\mathcal{D}$ of base-stations and a subset H_j of channels for each base-stations $j \in \mathcal{D}$ such that the conditions in (6) and (9) are satisfied.
3) Base-station j chooses uniformly at random a permutation $(v_1, ..., v_{|H_j|})$ of the H_j channels.
4) The iteration is divided into $|H_j|$ rounds. In the l-th round, the base-station j updates channel v_l using the

[6]This set $\mathcal{D}$ is similar to the "decision schedule" in [13,14].

transition probability in (19) and (20) shown later.

The above procedure is similar to Option 3. However, the key difference is in Step 4, where we will propose a new method in (19) and (20) to compute the transition probability with low overhead. We explain the thought process as follows. Let $\vec{x}$ and $\vec{y}$ again denote the channel-allocation vectors before and after an iteration, respectively. At the l-th round, we can use $\vec{z}^l = (\vec{y}(\bigcup_{n=1}^{l-1} v_n), \vec{x}(A \backslash \bigcup_{n=1}^{l-1} v_n))$ to express the channel allocation to update from[7]. Recall that in Option 3 (and similarly in Option 1), the transition probability at round l is given by (12), which is rewritten below after taking into account the changes in previous rounds:

$$\pi(\vec{y}(H_{j,v_l})|\vec{z}^l(A\backslash H_{j,v_l})) = \frac{\exp(y_{j,v_l}\Delta_{j,v_l})}{1+\exp(\Delta_{j,v_l})}, \tag{13}$$

where $H_{j,v_l} = \{(j, v_l)\}$, and Δ_{j,v_l} can be written as

$$\begin{aligned}\Delta_{j,v_l} &= \tfrac{1}{T}\left[V_j(1,\vec{z}^l(A_j\backslash H_{j,v_l})) - V_j(0,\vec{z}^l(A_j\backslash H_{j,v_l}))\right] \\ &+\tfrac{1}{T}\sum_{\{h\in\mathcal{N}_j\}}\left[V_h(1,\vec{z}^l(A_h\backslash H_{j,v_l})) - V_h(0,\vec{z}^l(A_h\backslash H_{j,v_l}))\right].\end{aligned} \tag{14}$$

Note that the first argument in $V_j(\cdot,\cdot)$ and $V_h(\cdot,\cdot)$ corresponds to the channel allocation for v_l.

Now, when the number of channels is large, and the size of $|H_j|$ is not too large, the channel-allocation vector $\vec{z}^l$ differs from $\vec{x}$ by only a few elements. Let V_j^0 and V_h^0 be the optimal objective values of Problem (3) for base-station j and base-station $h \in \mathcal{N}_j$. Then, we can expect that V_j and V_h should be close to V_j^0 and V_h^0. The key idea is then to use the solutions to Problem (3) for V_j^0 and V_h^0 to predict the value in (14). In this way, we can approximate the transition probability without incurring additional computation/communication overhead in each round.

To illustrate the idea, we start with some basic properties of the solutions to Problem (3). Let $\vec{R} = [R_i]$ and $\vec{\phi} = [\phi_{ik}]$. Associate dual variables λ_i to the first equality constraint of (3). Let Φ be the domain of $\vec{\phi}$ given in (3). The dual objective function of Problem (3) is then [5]:

$$g(\vec{\lambda}) = \max_{\vec{R},\vec{\phi}\in\Phi} \sum_{i\in\mathcal{S}_j} U_i(R_i) - \sum_{i\in\mathcal{S}_j}\lambda_i(R_i - \sum_{k=1}^{K}\phi_{ik}r_{ik}). \tag{15}$$

Let us first focus on the change of utility at base-station j. Recall that the value of Δ_{j,v_l} is determined by the utility difference with or without a channel v_l allocated to base-station j. Suppose that Problem (3) is solved twice for a base-station $j \in \mathcal{D}$, firstly *with* a channel v_l assigned to base-station j and secondly *without* the channel v_l assigned to it. Let $(\vec{R}^*, \vec{\phi}^*)$ and $\vec{\lambda}^*$ be a pair of optimal primal and dual variables for Problem (3) when channel v_l is assigned to base-station j, and let V_j be the corresponding optimal objective value. Similarly, let $\vec{R}^{*(\text{wo})}, \vec{\phi}^{*(\text{wo})}$, $\vec{\lambda}^{*(\text{wo})}$, and $V_j^{(\text{wo})}$ be the corresponding values for Problem (3) when channel v_l is *not* assigned to base-station j. Note that $r_{i,v_l}^{(\text{wo})} = 0$ for all users i. Since Problem (3) is convex, the KKT condition [5] must hold, and we have:

$$\lambda_i^* = U_i'(R_i^*), \quad R_i^* = \sum_{k=1}^{K}\phi_{ik}^* r_{ik}. \tag{16}$$

If user i is served at channel k, i.e., $\phi_{ik}^* > 0$, we also have:

$$\lambda_i^* r_{ik} = \max_{u\in\mathcal{S}_j}\lambda_u^* r_{uk}. \tag{17}$$

A similar set of equations will also hold for Problem (3) without channel v_l. Intuitively, this set of equations suggest that if a channel v_l is taken away from a base-station j, it will be taken away from user i (see (17)). For this user, the corresponding utility decrease will be approximately $U_i'(R_i^*)r_{i,v_l} = \lambda_i^* r_{i,v_l} = \max_{u\in\mathcal{S}_j}\lambda_u^* r_{u,v_l}$. Hence, it suggests that the value of $\max_{u\in\mathcal{S}_j}\lambda_u^* r_{u,v_l}$ would be a good estimate of the utility difference between V_j and $V_j^{(\text{wo})}$. The following lemma makes this intuition more precise.

Lemma 1: Suppose that there exists a constant D such that $|\lambda_i^{*(\text{wo})} - \lambda_i^*| \le D$. Then,

1) $V_j \ge V_j^{(\text{wo})} + \max_{i\in\mathcal{S}_j}\lambda_i^* r_{i,v_l}$.
2) $V_j \le V_j^{(\text{wo})} + \max_{i\in\mathcal{S}_j}\lambda_i^* r_{i,v_l} + D\sum_{i\in\mathcal{S}_j}\phi_{i,v_l}^* r_{i,v_l}$.

Proof: Detailed proof can be found in [21] ■

When the number of channels K is large, and only a few channels $|H_j|$ are updated in one iteration, we would expect that the dual variables will not change much across rounds, i.e., the value of D will be small. Lemma 1 then states that $\max_{i\in\mathcal{S}_j}\lambda_i^* r_{i,v_l}$ becomes a good estimate of the utility difference when a channel v_l is removed from base-station j. Note that we can replace λ_i^* by $\lambda_i^{*(\text{wo})}$ and use $\max_{i\in\mathcal{S}_j}\lambda_i^{*(\text{wo})} r_{i,v_l}$ to estimate the utility difference. The latter will add another error term $D\max_{i\in\mathcal{S}_j} r_{i,v_l}$, which again will be small if D is small. Further, for any round l, we can replace λ_i^* or $\lambda_i^{*(\text{wo})}$ by the optimal dual variables before round 1. In that case, the error terms will accumulate linearly in l. However, as long as l is bounded, the error terms can still be bounded.

We have estimated the utility difference for a base-station $j \in \mathcal{D}$. For its neighboring base-station $h \in \mathcal{N}_j$, the situation is slightly different because base-station h does not change its own channel allocation. Instead, the user rates $r_{i,v_l}, i \in \mathcal{S}_h$, change due to the change of channel allocation by base-station j. If the base-station h does not use channel v_l, then this change will not affect its utility. If the base-station h does use channel v_l, suppose that the user rates without and with channel v_l assigned to base-station j are r_{i,v_l}^1 to r_{i,v_l}^2, respectively, for all $i \in \mathcal{S}_h$. We can then view the change at base-station h as the concatenation of two steps: base-station h first removes channel v_l with user rates r_{i,v_l}^1 and then adds back channel v_l with user rates r_{i,v_l}^2. In order to differentiate the base-stations, we use λ_{ij}^* and λ_{ih}^* to denote the optimal dual variables for Problem (3) at base-station j and h, respectively. By Lemma 1, we can then approximate the total utility difference by

$$V_h \approx V_h^{(\text{wo})} - \max_{i\in\mathcal{S}_h}\lambda_{ih}^* r_{i,v_l}^1 + \max_{i\in\mathcal{S}_h}\lambda_{ih}^* r_{i,v_l}^2, \tag{18}$$

[7]Of course, elements outside A_j and $\cup_{h\in\mathcal{N}_j}A_h$ may also change. However, their changes do not affect the transition probability at base-station j as can be seen in (13) and (14).

where V_h and $V_h^{(\mathrm{wo})}$ are the optimal utility for base-station h when base-station j uses or does not use channel v_l, respectively.

Taking all the above into account, the transition probability at round l at base-station j can be approximated as:

$$\tilde{\pi}(\vec{y}(H_{j,v_l})|\vec{z}^l(A\backslash H_{j,v_l})) = \frac{\exp(y_{j,v_l}\tilde{\Delta}_{j,v_l})}{1+\exp(\tilde{\Delta}_{j,v_l})}, \tag{19}$$

where $H_{j,v_l} = \{(j, v_l)\}$ and $\tilde{\Delta}_{j,v_l}$ can be written as

$$\begin{aligned}\tilde{\Delta}_{j,v_l} &= \tfrac{1}{T}\max_{i\in\mathcal{S}_j}\lambda^*_{ij}r_{i,v_l} \\ &+\tfrac{1}{T}\textstyle\sum_{\{h\in\mathcal{N}_j\}}\left[-\max_{i\in\mathcal{S}_h}\lambda^*_{ih}r^1_{i,v_l}+\max_{i\in\mathcal{S}_h}\lambda^*_{ih}r^2_{i,v_l}\right]I_{h,v_l}.\end{aligned} \tag{20}$$

where all the optimal dual variables λ^*_{ij} and λ^*_{ih} can be taken as those before round 1. Our proposed fast multi-channel Gibbs sampling algorithm then uses (19) and (20) in Step 4.

Remark: Despite the similarity between (13) and (19), we emphasize that they lead to significantly different speed and overhead. In (19) and (20), since the values of the optimal dual variables before round 1, which are calculated in Step 1 of the proposed algorithm, are used, base-station j can use one control message to send information about H_j to base-station h and ask base-station h to return the corresponding values in $\tilde{\Delta}_{j,v_l}$ for all channels v_l in H_j, again using one message. Then, base-station j can carry out the $|H_j|$ rounds of transitions without additional communication/computation involving neighboring base-stations. Thus, the computation/communication overhead is significantly reduced, and the channel updates can be carried out with much faster speed than Option 3 (using (13)).

A. Analysis

Next, we will analyze the performance. Ideally, we would like to show that as the number of channels K increases, the stationary distribution of the Markov chain using the approximate (19) and (20) will approach the stationary distribution of the original Markov chain using (13) and (14). However, since the stationary distribution of the original Markov chain also changes with K, it appears to be difficult to directly analyze the convergence in terms of the stationary probability. Instead, in this subsection, we will focus on showing how the approximate transition probability in (19) approaches (13).

As we have discussed, it is sufficient to focus on one base-station $j \in \mathcal{D}$ and study the utility difference with or without a single channel v_l (see Lemma 1). Intuitively, as the number of channels K increases, the contribution of any one channel, i.e., the term $\max_{i\in\mathcal{S}_j}\lambda^*_i r_{i,v_l}$ in Lemma 1, decreases as $\Theta(1/K)$. Hence, if we can show that the error term $D\sum_i \phi^*_{i,v_l}r_{i,v_l}$ in Lemma 1 decreases faster than $\Theta(1/K)$, then by setting the temperature $T = \alpha/K$, the effect of the error term will approach 0 as K increases. However, the bound D in general depends on the particular channel-allocation patterns. In the sequel, we will study conditions under which D can be bounded uniformly over all transitions. Due to space constraints, we will present the results without proofs. The interested readers are referred to our technical report in [21] for detailed proofs.

Again, we focus on the setting in Lemma 1, where Problem (3) is solved twice, firstly with channel v_l allocated to base-station j and secondly without. Our first lemma is:

Lemma 2: If the utility function is $U_i(R_i) = \log(R_i)$, then $|\lambda^*_i - \lambda^{*(\mathrm{wo})}_i| \le D$ if and only if $\frac{|R^*_i - R^{*(\mathrm{wo})}_i|}{R^*_i R^{*(\mathrm{wo})}_i} \le D$.

Proof: The result can be easily shown by (16). ■

This lemma shows that to bound the difference in λ^*_i and $\lambda^{*(\mathrm{wo})}_i$, we can instead focus on the difference in R^*_i and $R^{*(\mathrm{wo})}_i$. Hence, in the following, we will let $U_i(R_i) = \log(R_i)$ and focus on the difference between R^*_i and $R^{*(\mathrm{wo})}_i$.

Lemma 3: $R^*_i \ge R^{*(\mathrm{wo})}_i$, for all $i \in \mathcal{S}_j$.

Lemma 3 is quite intuitive. It states that after an additional channel v_l is allocated to base-station j, the optimal rate for all associated users of base-station j cannot decrease. Next, we would like to bound the increase $R^*_i - R^{*(\mathrm{wo})}_i$. Here, we need a condition on the heterogeneity of the user rates. Let

$$r_{\max} = W_T \max_{i\in\mathcal{S}_j}\log_2\Big(1+\frac{P_jG_{ij}}{W_TN_0+\eta^{\max}_{ij}}\Big) \tag{21}$$

and

$$r_{\min} = W_T \min_{i\in\mathcal{S}_j}\log_2\Big(1+\frac{P_jG_{ij}}{W_TN_0+\eta^{\max}_{ij}+\sum_{h\in\mathcal{N}_j}P_hG_{ih}}\Big). \tag{22}$$

Then, from (2), we have $\max_{\{i\in\mathcal{S}_j,k\}} r_{ik} \le r_{\max}/K$ and $\min_{\{i\in\mathcal{S}_j,k\}} r_{ik} \ge r_{\min}/K$. Let $B = r_{\max}/r_{\min}$.

Lemma 4: $R^*_i \le R^{*(\mathrm{wo})}_i + Br_{\max}/K$ for all $i \in \mathcal{S}_j$.

Lemma 4 implies that the increase $(R^*_i - R^{*(\mathrm{wo})}_i)$ must be bounded by $Br_{\max}/K$, which decreases as $\Theta(1/K)$. The bound can also be shown to be tight in the order sense. The interested readers can refer to [21] for details. Combining Lemmas 3 and 4, we then have

$$\frac{|R^*_i - R^{*(\mathrm{wo})}_i|}{R^*_i R^{*(\mathrm{wo})}_i} \le \frac{Br_{\max}}{K\min\limits_{i\in\mathcal{S}_j}R^*_i\min\limits_{i\in\mathcal{S}_j}R^{*(\mathrm{wo})}_i}. \tag{23}$$

It remains to bound the denominator.

Lemma 5: Let c be a fixed constant in $(0, 1)$. Suppose that at least a fraction c of the K channels is allocated to each base-station. Then, $\min[\min\limits_{i\in\mathcal{S}_j}R^*_i, \min\limits_{i\in\mathcal{S}_j}R^{*(\mathrm{wo})}_i] \ge cr_{\min}/|\mathcal{S}_j|$.

Note that by Lemma 5 we can conclude that the right-hand-side of (23) does decrease as $\Theta(1/K)$. Combining with Lemmas 1 and 2, we then obtain the following main result. Let $\tilde{\mathcal{I}}$ denote the set of all possible $\vec{I}$ such that the condition in Lemma 5 holds.

Proposition 6: Suppose that $U_i(R_i) = \log(R_i)$, $T = \alpha/K$, and at least a fraction c of the K channels is allocated to each base-station. Further, suppose that $|\mathcal{S}_j|$ and $|H_j|$ are upper-bounded by some constants. Then, we have

$$\lim_{K\to\infty}\sup_{\vec{x},\vec{y}\in\tilde{\mathcal{I}}}|\Delta_{j,v_l} - \tilde{\Delta}_{j,v_l}| = 0. \tag{24}$$

Proposition 6 confirms that when K is large, and each update involves a relatively small number $|H_j|$ of channels, then the transition probability of the proposed fast Gibbs sampling will approach that of the original Gibbs sampling.

Remark: Note that the condition in Lemma 5 suggests the following modification to the proposed algorithm. Suppose that the channel-allocation vector $\vec{x}$ before an iteration satisfies $\vec{x} \in \tilde{\mathcal{I}}$. Then, we only need to ensure that the vector $\vec{y}$ that the algorithm transits to is still in $\tilde{\mathcal{I}}$. Each base-station j can ensure this property distributively by setting the transition probability (19) to zero if the state after round l is not in $\tilde{\mathcal{I}}$.

Remark: Our analysis may be conservative since we focus on the worst-case scenario. However, it gives us some confidence that the proposed fast Gibbs sampling algorithm will likely follow the trajectory of the original Gibbs sampling. In fact, our simulation results in Section V show that such convergence occurs even for a moderate number of channels. A detailed discussion is provided in [21].

Finally, in our simulations, we found that the standard gradient algorithms [5] are still quite slow for solving Problem (3) in each iteration. To resolve this problem, we develop a second-order algorithm [5, Chapter 10] [22] to expedite the convergence. For details, please refer to [21].

V. Simulation

In this section, we use simulation to evaluate the fast Gibbs sampling algorithms proposed in Section IV. We first simulate a LTE network with 19 same-sized hexagon cells arranged in a three-ring structure. (There are a center cell, an inner ring of 6 cells, and an outer ring of 12 cells.) Each base-station is at the center of a cell, and the inter-site distance (ISD) is 500m. A total bandwidth of 20Mhz is divided into 50 channels. The power of each base-station is 47dbm and is equally shared over the 50 channel. Each cell has 10 users randomly placed inside its coverage area. The utility function of each user is $\log(\cdot/10^6)$. The channel gains G_{ij} are modeled by a path-loss component with exponent $n = 2.2$ and log-normal shadow fading with standard deviation $\sigma = 8$dm. The neighboring set $\mathcal{N}_j$ for each base-station j consists of any base-stations within 500m range.

We first compare the convergence of the proposed fast Gibbs-sampling algorithm with the "sequential-update" version of standard Gibbs sampling, i.e., Option 3 in Section III-B.[8] For all versions of Gibbs sampling, the temperature T of the Gibbs distribution is 0.002. We also compare with the Metropolis-Hastings algorithm [8]. The Metropolis-Hastings algorithm is similar to Gibbs sampling, but it chooses the transition probability differently. Specifically, after each base-station chooses a subset of channels H_j to update, the Metropolis-Hastings algorithm randomly chooses a feasible "proposed" state $\vec{y}(H_j)$ and compares $\pi(\vec{y}(H_j), \vec{x}(A\backslash H_j))$ with $\pi(\vec{x})$ according to (5), where $\vec{x}$ is the original state. If the former is larger, the base-station accepts state $\vec{y}(H_j)$ with probability one. Otherwise, the base-station accepts state $\vec{y}(H_j)$ with probability $\pi(\vec{y}(H_j), \vec{x}(A\backslash H_j))/\pi(\vec{x})$.

[8]We do not report the "multi-channel update" version, i.e., Option 2, due to the exponential computation/communication overhead. However, the convergence speed of Option 2 should be comparable to Option 3.

The simulation result is shown in Fig. 1(a), where we plot the total system utility as a function of the iterations. As readers can see, for all versions of Gibbs sampling algorithms, updating fewer channels in each iteration leads to a slower convergence speed. Further, although the Metropolis-Hastings algorithm can update a larger number of channels with low computation overhead, the proposed change in each iteration may not always lead to larger utility. Thus, its convergence speed is still slow. In contrast, the proposed fast Gibbs-sampling algorithm enjoys much faster convergence when updating multiple channels at a time. By comparing the curves "sequential-5" and "fast-5", we can observe that the utility evolution of the "fast" algorithm is close to that of the "sequential update" version. Finally, the "fast-15" curve converges even more quickly than other curves. These results confirm that our proposed algorithm is sufficiently accurate for finding the optimal channel-allocation. Further, it leads to fast convergence and low overhead.

Next, we simulate a setting with non-uniform load and evaluate whether the proposed algorithm can adapt to load patterns. Specifically, we compare fast Gibbs-sampling algorithm with (1) universal 1-reuse, where each base-station uses all the channels, and (2) strict FFR (Fractional Frequency Reuse), where $1/4$ of the channels are shared by all base-stations, and the rest $3/4$ of the channels are allocated according to a 3-reuse pattern [3]. Note that according to the 3-reuse pattern, we can divide the cells into three groups. We then let the cells in group 1, 2, and 3 have 20 users, 10 users, and 1 user, respectively. For all schemes, once the inter-cell channel allocation is determined, each cell solves Problem (3) for intra-cell user scheduling. The CDF of the resulting user rates under different schemes are shown in Fig. 1(b). As we can see, the user rates under universal 1-reuse are much poorer due to the strong inter-cell interference. Strict FFR performs better. However, since it does not react to the non-uniform load, its performance is still significantly worse than our proposed algorithm. A deeper analysis of the results reveals that under our algorithm, roughly 27 channels, 20 channels, and 3 channels are allocated to the base-stations in group 1, 2, and 3, respectively. As a result, the achieved user-rates are consistently better.

We next simulate our algorithm for a heterogenous network with both macro- and pico-cells (see Fig. 1(d)). Specifically, there are three macro-cells with the ISD of 500m. On the boundary of every two macro-cells, there is a pico-cell. Further, one additional pico-cell is placed in the interior of the coverage area of each macro-cell and is placed away from the boundary pico-cells. The power and coverage radius of each pico base-station is 27dbm and 75m, respectively. Each macro- or pico-cell has 10 users randomly placed in the cell. The neighboring set $\mathcal{N}_j$ for each macro and pico base-station consists of those base-stations that are immediate neighbors (refer to [21] for details).

We compare fast Gibbs-sampling with (1) universal 1-reuse, where each base-station uses all the channels, and (2) strict FFR, where macro base-stations use strict FFR described earlier [3], while pico base-stations use 1-reuse.

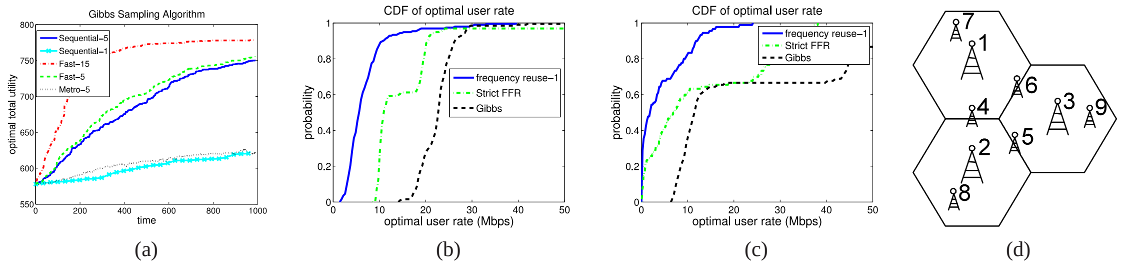

Fig. 1. (a): The utility evolution of different Gibbs-sampling algorithms. "Sequential" represents Option 3 (sequential update). "Fast" represents fast Gibbs-sampling. "Metro" represents the Metropolis-Hastings algorithm. The number behind the description of the algorithm is the number of channels updated by the chosen base-station in each iteration. (b): The cumulative distribution function (CDF) of the user rates under non-uniform load. (c): The cumulative distribution function (CDF) of the user rates under heterogeneous topology. (d): The topology of a heterogeneous network with three macro-cells and six pico-cells.

The simulation result is shown in Fig. 1(c), where our proposed algorithm demonstrates even more significant performance gains. As we can see, the user rates under universal 1-reuse is again very poor due to the strong inter-cell interference. The performance of pico-cells under strict FFR is also poor because pico-cells still receive strong inter-cell interference. In contrast, with our fast Gibbs-sampling algorithm, the channel allocation is automatically adjusted to manage interference, and thus the user rates are improved significantly. Specifically, the upper-right-hand side of the CDF represents interior pico-cells whose user rates are nearly doubled. The lower-left-hand side of the CDF represents the macro-cells and boundary pico-cells whose user rates are also significantly improved. We select the macro-cell $\#1$, the pico-cell $\#4$, and the pico-cell $\#7$ in Fig. 1(d) and observe the number of channels allocated to each of them to be 7, 12 and 43, respectively. Moreover, only two channels allocated to the boundary pico-cell $\#4$ are shared by the two neighboring pico-cells or the three neighboring macro-cells. Such an adaptive channel allocation not only improves the overall system utility, but also helps to improve the CDF of the user rates consistently across all users.

VI. Conclusion

In this paper, we propose a fast multi-channel Gibbs sampling algorithm for frequency resource allocation in the OFDMA cellular networks. The key idea of our proposed algorithm is to view the update of one channel allocation decision as a perturbation to the optimization problem. Hence, we can utilize the perturbation analysis to let each base-station quickly and accurately update many channels without excessive computation/communication overhead. Our simulation results show that fast Gibbs sampling algorithm can adapt to non-uniform load pattern and irregular cell deployment. In the future, we will refine our analysis bound and extend this novel idea to power control and user association problems.

References

[1] "Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2012-2017," *Cisco white paper*, 2013.

[2] J. G. Andrews, "Seven Ways that HetNets are a Cellular Paradigm Shift," *IEEE Communications Magazine*, vol. 51, no. 3, pp. 136–144, 2013.

[3] T. Novlan, J. G. Andrews, I. Sohn, R. K. Ganti, and A. Ghosh, "Comparison of Fractional Frequency Reuse Approaches in the OFDMA Cellular Downlink," in *IEEE GLOBECOM*, 2010.

[4] S. Sesia, I. Toufik, and M. Baker, *LTE: the UMTS long term evolution*. Wiley Online Library, 2009.

[5] S. P. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, 2004.

[6] J. Huang, V. G. Subramanian, R. Agrawal, and R. Berry, "Joint Scheduling and Resource Allocation in Uplink OFDM Systems for Broadband Wireless Access Networks ," *IEEE Journal on Selected Areas in Communications (JSAC)*, vol. 27, no. 2, pp. 226–234, 2009.

[7] I. Hou and C. S. Chen, "Self-Organized Resource Allocation in LTE Systems with Weighted Proportional Fairness," in *IEEE ICC*, 2012.

[8] S. C. Borst, M. G. Markakis, and I. Saniee, "Nonconcave Utility Maximization in Locally Coupled Systems, With Applications to Wireless and Wireline Networks," *IEEE/ACM Trans. on Networking*, 2013.

[9] C. S. Chen, F. Baccelli, and L. Roullet, "Joint Optimization of Radio Resources in Small and Macro Cell Networks," in *IEEE VTC*, 2011.

[10] M. Chen, S. C. Liew, Z. Shao, and C. Kai, "Markov Approximation for Combinatorial Network Optimization," in *IEEE INFOCOM*, 2010.

[11] B. Kauffmann, F. Baccelli, A. Chaintreau, V. Mhatre, K. Papagiannaki, and C. Diot, "Measurement-Based Self Organization of Interfering 802.11 Wireless Access Networks," in *IEEE INFOCOM*, 2007.

[12] L. Jiang and J. Walrand, "A Distributed CSMA Algorithm for Throughput and Utility Maximization in Wireless Networks," *IEEE/ACM Trans. on Networking*, vol. 18, no. 3, pp. 960 – 972, June 2010.

[13] J. Ni, B. Tan, and R. Srikant, "Q-CSMA: queue-length based CSMA/CA algorithms for achieving maximum throughput and low delay in wireless networks," in *IEEE INFOCOM*, 2010.

[14] P.-K. Huang and X. Lin, "Improving the Delay Performance of CSMA. Algorithms: A Virtual Multi-Channel Approach," in *IEEE INFOCOM*, 2013.

[15] S. Zhou, X. Wu, and L. Ying, "Distributed Power Control and Coding-Modulation Adaptation in Wireless Networks using Annealed Gibbs Sampling," in *IEEE INFOCOM*, 2012.

[16] C. S. Chen and F. Baccelli, "Self-Optimization in Mobile Cellular Networks: Power Control and User Association," in *IEEE ICC*, 2010.

[17] D. Qian, D. Zheng, J. Zhang, N. B. Shroff, and C. Joo, "Distributed CSMA Algorithms for Link Scheduling in Multihop MIMO Networks Under SINR Model," *IEEE/ACM Trans. on Networking*, vol. 21, no. 3, June 2013.

[18] M. Andrews and L. Zhang, "Utility optimization in heterogeneous networks via CSMA-based algorithms," in *WiOpt '13*, Tsukuba Science City, Japan, May 2013.

[19] T. M. Cover and J. A. Thomas, *Elements of Information Theory*, 2nd ed. Wiley, 2006.

[20] P. Bremaud, *Markov Chains, Gibbs Fields, Monte Carlo Simulation, and Queues*. New York: Springer-Verlag, 1999.

[21] P.-K. Huang, X. Lin, N. B. Shroff, and D. J. Love, "Fast Multi-Channel Gibbs-Sampling for Low-Overhead Distributed Resource Allocation in OFDMA Cellular Networks," *Technical Report, Purdue University*, 2013. [Online]. Available: http://docs.lib.purdue.edu/ecetr/450/

[22] J. Liu, C. H. Xia, N. B. Shroff, and H. D. Sherali, "Distributed Cross-Layer Optimization in Wireless Networks: A Second-Order Approach," in *IEEE INFOCOM*, 2013.