Context-Driven Power Management in Cache-Enabled Base Stations using a Bayesian Neural Network

Luhao Wang, Shuang Chen, Massoud Pedram
Department of Electrical Engineering
University of Southern California, Los Angeles, CA, 90089
Email: \{luhaowan, shuangc, pedram\}@usc.edu

Abstract—Aggressive network densification in next generation cellular networks is accompanied by an increase of the system energy consumption and calls for more advanced power management techniques in base stations. In this paper, we present a novel proactive and decentralized power management method for small cell base stations in a cache-enabled multi-tier heterogeneous cellular network. User contexts are utilized to drive the decision of dynamically switching a small cell base station between the active mode and the sleep mode to minimize the total energy consumption. The online control problem is formulated as a contextual multi-armed bandit problem. A variational inference based Bayesian neural network is proposed as the solution method, which implicitly finds a proper balance between exploration and exploitation. Experimental results show that the proposed solution can achieve up to 46.9% total energy reduction compared to baseline algorithms in the high density deployment scenario and has comparable performance to an offline optimal solution.

I. INTRODUCTION

With the proliferation of Internet-of-Things (IoTs) and smart-phones, there has been an explosion of mobile data. According to Cisco’s report [1], the annual mobile traffic volume will reach 587 exabytes by 2021, which is 122 times more than that in 2011. As a result of the demand for ever more network capacity, existing backhaul links and core networks are faced with greater challenge. Among all investigated solutions to the soaring mobile data traffic, deployment of ultra-dense small cells, a.k.a. network densification, as proposed in some preliminary 5G cellular network standards (e.g. METIS project [2]), is deemed the most promising solution. Network densification not only partially shifts the burden from the core network and backhaul links to the edge of the network and enable higher transmission speed [3], but also synergizes well with other key techniques in 5G cellular networks such as massive multiple-input multiple-output and millimeter wave transmission [4], [5] because it reduces the communication distance between a user and the nearest base station.

One major reason that network densification can help reduce the traffic volume on backhaul links is that the deployed small base stations (sBSs) introduce an additional level of content caching which further lowers the cost of acquiring the data requested by users. Since it has been observed that a small set of popular multimedia entities usually contribute to a large portion of the network traffic, saving popular files in base stations is preferred over repetitively fetching these files via backhaul links because of the saving of both backhaul capacity and energy consumption. It has been proved that caching of various types of data including text, image, and video is applicable in 3G [6] and 4G LTE [7] cellular networks and can reduce up to 50% of traffic volume in the core network. There has been a large body of research addressing the management of cache-enabled base stations [8]–[11], which try to improve the quality of service (QoS) in terms of round-trip time and/or throughput subject to limited backhaul bandwidth.

While network densification helps improve the QoS, it is associated with a significant power consumption overhead because of the large number of deployed small base stations. According to data in references [12], [13], there can be 100 times more sBSs than macro base stations (MBSs) in a cellular network in practice, resulting in twice as much the total peak power consumption. Hence, from the perspective of energy efficiency, sBSs need to be self-organizing, low-cost, and energy-efficient. Based on the observation that sBSs only help when their cached contents are requested, some opportunistic management policies have been proposed to switch an sBS into a “sleep mode” when no user request can be serviced [14], [15]. However, such techniques can only achieve limited energy saving because the sBS needs to monitor user traffic even in the “sleep mode” in order to resume normal operation upon arrival of new requests.

Further energy saving can be achieved by proactively putting some of sBSs into a deeper sleep mode in which sBSs do not serve or monitor user requests. While the coverage of the cellular network is not affected when some sBSs are not active because at least one MBS can be reached by any user, the system may suffer from reduced throughput and/or energy efficiency. In order to avoid the case in which an sBS is not active when its cached contents are frequently requested, an accurate estimation of the user demand in the future is required. In fact, designing a predictor to determine whether the cached contents of an sBS will be in demand in an upcoming time period is a challenging task, since the user traffic pattern is dependent on the user context comprised of a large variety of factors such as the number, age, and interests of the users in the service range. Moreover, as an sBS constantly refreshes its cache over time in order to include the emerging popular contents, the predictor needs to be able to adapt itself to new cached contents and user contexts.

In this paper, we present a decentralized online control algorithm that can dynamically predict the workload pattern and control the operation mode of sBSs to maximize the energy efficiency a cache-enabled multi-tier heterogeneous cellular network (HetNet) [16], [17]. While some recent research such
as Müller et al. [18] tries to address similar problems, all such prior work fails to connect the dynamic user context with the energy consumption of the network. We utilize the user information collected by an sBS including the gender and age of a user to estimate the energy efficiency benefit of switching an sBS to the sleep mode. To account for the uncertainty of user request pattern in the future, the control problem is formulated as a contextual multi-armed bandit problem. A variational inference based Bayesian neural network (BNN) is used as the solution, which outperforms some well-known algorithms.

The rest of this paper is structured as follows. In Section II, the system model of a cache-enabled multi-tier cellular network is introduced. In Section III, we propose the problem formulation and solution method. Experimental results are presented in Section IV. Section V concludes the paper.

II. SYSTEM MODEL

We consider a two-tier cache-enabled HetNet as shown in Fig. 1, where each tier-1 MBS is associated with a number of tier-2 sBSs within its coverage and each sBS is associated with a set of connected user equipments (UEs), e.g. smartphones, laptop computers, tablet PCs, etc. The MBS is also connected to a hub in the core network via a backhaul link to access contents on the Internet. sBSs are usually deployed to enhance capacity in locations of high demand under the coverage of MBSs. Therefore, sBSs don’t contribute to the coverage of MBSs. The spatial distribution of base stations follows homogeneous Poisson Point Process (PPP) [19] based on the stochastic geometry approach. The corresponding density of MBSs and sBSs are $\lambda_M$ and $\lambda_S$ respectively.

In practice, while it is possible that a UE is in the coverage of multiple sBSs, it will only connect to the sBS with the strongest signal strength. An sBS can choose to either operate in an “active mode” in which it can serve incoming user requests or enter a “sleep mode” in which it does not serve or monitor any user requests. If an sBS is in the sleep mode in order to achieve higher energy efficiency, all requests from the UEs connected to the sBS will be served by the MBS. Because the interest of this paper is the distributed control of sBSs, we will only focus on one sBS along with its connected UEs and associated MBS unless otherwise noted.

A. Channel Model

Let $S$ and $U$ denote the set of sBSs and UEs, respectively. For an interested sBS $S \in S$, the signal-to-interference-plus-noise ratio (SINR) when data is transmitted from the sBS to a connected UE $u \in U$, denoted by $SINR_S(u)$, is defined as

$$SINR_S(u) = \frac{P_S^u \cdot g_S(u)}{I_S(u) + \sigma^2},$$  

(1)

where $P_S^u$ is the transmission power of the sBS, $g_S(u)$ is the channel gain between the sBS and UE $u$, $I_S(u)$ is the interference power received by UE $u$ from other sBSs, and $\sigma^2$ is the noise power. $I_S(u)$ can be calculated as

$$I_S(u) = \sum_{S' \in S \setminus S} P_{S'} g_{S'}(u).$$

The channel gain, $g_S(u)$, can be further expanded as

$$g_S(u) = h_S(u) \cdot L_S(d_S(u)),$$

(2)

where $d_S(u)$ is the distance between the sBS and UE $u$, $L_S(\cdot)$ is the log-distance path loss function, and $h_S(u)$ accounts for the channel fading. Similarly, when data is transmitted from the MBS to a UE $u$, the SINR, denoted by $SINR_M(u)$, can be defined as

$$SINR_M(u) = \frac{P_M^u \cdot g_M(u)}{I_M(u) + \sigma^2},$$

(3)

where $P_M^u$ is the transmission power of the MBS, $I_M(u)$ is the interference power from other MBSs, and $g_M(u)$ is the channel gain between the MBS and UE $u$ which can be expressed the same way as in Eqn. (2).

The channel capacity, i.e. the maximum data throughput in a channel, denoted by $TP$, can be calculated based on the Shannon-Hartley theorem as follows

$$TP = BW \cdot \log_2(1 + SINR),$$  

(4)

where $BW$ is the effective bandwidth of the channel, and $SINR$ is calculated using Eqn. (1) or (3). $BW_S$ and $BW_M$ are denoted as the effective bandwidth of sBS and MBS, respectively. In software-defined networks (SDNs), $BW_M$ can be modeled as $BW_M = (1 - O_M) \cdot BW$, with $O_M$ is the overhead cost of MBS and $BW$ is the system bandwidth [20]. For sBSs, $BW_S \approx W$ can be achieved with the assistance of MBS. For the backhaul link between an MBS and the core network, the channel capacity is assumed to be constant.

B. Service Model

Fig. 2 shows the service model. It is assumed that both MBSs and sBSs are equipped with caches (which are usually solid state drives) with finite capacities. Generally speaking, an MBS has a larger cache than an sBS. When a UE initiates a request for a file, the request will first be served by its associated sBS if it is active. If the requested file is present in the cache of the sBS, it will be transferred via the downlink of the sBS to the UE. Otherwise, the request will be redirected to the associated MBS. If the requested file can be found in the cache of the MBS, then the cached copy will be transferred from the MBS. If the requested file can be served from neither the cache in the sBS and the MBS, the MBS is responsible for downloading it from the core network via the backhaul link and relaying it to the UE.
In addition, the sBS will routinely attempt to acquire the information of the connected UE such as the type and the location of the equipment. Please note that the sBS may not have access to some context information due to privacy reasons. However, our proposed framework doesn’t depend on any specific context information and it can utilize the information that can be collected from UEs.

C. Cache Model

The purpose of caching popular contents in the base stations is to reduce the traffic load of MBSs. It is critical for sBSs to store the most frequently requested contents locally in caches to possibly maximize the cache hit. However, learning the content popularity and deciding which contents to be stored in caches are difficult considering the huge amount of contents. Especially, the amount of contents is growing exponentially and it is impossible to cache all popular contents. It is hence crucial to decide which contents to cache taking popularity into account.

We consider a practical situation that each sBS and MBS is limited in the cache size, which is in terms of how many files can be stored. Many policies for refreshing caches have been widely studied by other works and they are not the main focus of our work. In our model, we assume that $P_S^S$ and $P_M^M$ are the percentage of total contents files in sBS and MBS that will be replaced based on the frequency of requests after every time $T_c^S$ and $T_c^M$ for sBSs and MBS, respectively. Please note that our proposed framework doesn’t rely on any specific cache replacing policy and it can adapt to rapid cache updating.

D. Power Consumption Model

We adopt a power consumption model for base stations similar to that in reference [13], in which the total power consumption of a base station is comprised of static power, transmission power, and data access power. Note that since the total amount of transmitted data is dominated by the requested content, we ignore the power consumption incurred from sending requests or other utility packets.

As discussed in Section II-B, an sBS will only transmit data if there is a cache hit for the requested file. Therefore, the power consumption of the sBS, denoted by $P_S^{tot}$, can be calculated as

$$P_S^{tot} = x_S \left( P_S^0 + \frac{1}{\eta_S} P_S^{tx,i} + w_{CA} \cdot TP_S \right) + (1 - x_S) P_S^{sleep}, \quad (5)$$

where $P_S^0$ is the static power consumption of an sBS, $P_S^{tx,i}$ is the instantaneous transmission power of the sBS, $\eta_S$ is the transmission efficiency that accounts for the power overhead for transmission such as channel encoding, $TP_S$ is the transmission rate that can be achieved from the transmission power, $w_{CA}$ is the energy efficiency (in J/bit or W/bps) of a cache read operation, $P_S^{sleep}$ is the power consumption of the sBS in the sleep mode, and $x_S$ is binary variable that indicates whether the sBS is active or not.

If a request is redirected to the MBS either due to a cache miss on the sBS or because the sBS is in the sleep mode, the MBS is responsible for issuing the transmission whether the requested file is in its cache or not. Therefore, the total power consumption of an MBS, denoted by $P_M^{tot}$, can be calculated as

$$P_M^{tot} = P_M^0 + \frac{1}{\eta_M} \cdot P_M^{tx,i} + w_{CA} \cdot TP_M^{hit} + w_{BH} \cdot TP_M^{miss}, \quad (6)$$

where $P_M^0$, $P_M^{tx,i}$, and $\eta_M$ are the static power, transmission power, and transmission efficiency for an MBS, respectively. $TP_M^{hit}$ is the data throughput of files available in the MBS’s cache, $TP_M^{miss}$ is the data throughput of files pulled via the backhaul link, and $w_{BH}$ is the energy efficiency for accessing the backhaul link.

Furthermore, if a file has to be downloaded from the core network to the MBS, there is an increase in the power consumption of the core network, denoted by $\Delta P_C$, which can be estimated as

$$\Delta P_C = w_{CN} \cdot TP_M^{miss}, \quad (7)$$

where $w_{CN}$ is the energy efficiency for data transmission in the core network.

III. PROBLEM FORMULATION AND SOLUTION METHOD

In this paper, a slotted time model is adopted for decision making in which the operation mode for an sBS is provided for time intervals of equal and constant duration $\tau$. Since the file requested from a cellular network is relatively small, we assume that all file transmission can be finished within the same time slot in which the request is initiated.

A. Problem Description

The objective of the proposed problem is to minimize the total energy consumption of the network while serving all user requests by controlling the operation modes of the sBSs. In a given time slot $t$, if the amount of requested data (in bits) from UEs connected to the sBS under study, denoted by $C_i^{tot}$, can be expressed as

$$C_i^{tot}[t] = C_S[t] + C_M[t] + C_{BH}[t], \quad (8)$$

where $C_S[t]$ is the amount of requested data that is cached by the sBS, $C_M[t]$ is the amount of requested data that is not cached by the sBS but can be found in MBS’s cache, and
$C_{BH}[t]$ is the amount of requested data that is not present in either sBS or MBS and has to be downloaded from the backhaul link.

When the sBS under study is in the active mode during time slot $t$, the total energy consumed to serve requests from its connected users, denoted by $E^{ON}[t]$, consists of three components, i.e. the energy consumption of the sBS under study itself, the energy consumed by the MBS to aid the sBS, and the increased amount of energy consumption in the core network to transfer data needed by the sBS. As mentioned above, the spatial densities of sBSs and MBSs are denoted by $\lambda_S$ and $\lambda_M$, respectively.

\[
E^{ON}[t] = \left( p^{S}_0 + \frac{\lambda_M}{\lambda_S} p^{M}_0 \right) \tau + \frac{1}{\eta_S} P^{S\tau}_S T_S[t] + \frac{1}{\eta_M} P^{M\tau}_M T_M[t] + w_{CA} (C_S[t] + C_M[t] + (w_{BH} + w_{CN}) C_{BH}[t]),
\]

(9)

where $P^{S\tau}_S$ and $P^{M\tau}_M$ are the maximum transmission power of an sBS and an MBS, respectively, whereas $T_S[t]$ and $T_M[t]$ are the amount of time needed to transmit files from the sBS and the MBS to the UEs in the coverage of the sBS under study, respectively, which can be calculated using the throughputs derived from Eqns. (1), (3), and (4). Please note that the ratio $\frac{\lambda_M}{\lambda_S}$ scales the effect of the MB and other sBSs excepting the studied one. The amount of requested files is also scaled explicitly by considering requests are directed to all sBSs and one MB.

On the other hand, if the sBS is in the sleep mode during time slot $t$, the total energy consumed to serve requests from its connected UEs, denoted by $E^{OFF}[t]$, can be estimated as

\[
E^{OFF}[t] = \left( P^{S\text{sleep}} + \frac{\lambda_M}{\lambda_S} p^{M}_0 \right) \tau + \frac{1}{\eta_M} P^{M\tau}_M T'_M[t] + w_{CA} C_M[t] + (w_{BH} + w_{CN}) (C_{BH}[t] + C_{Q}[t]),
\]

(10)

where $T'_M[t]$ is the amount of time needed to transmit files from the MBS to the users in the coverage of the sBS under study which can be calculated in the same way as $T_M[t]$ in Eqn. (9). Please note that we assume the cached contents in the sBS and the MBS do not overlap for the convenience of expression in Eqn. (10), but it can be trivially extended to the case in which there is overlapping between caches.

If we consider a total of $T$ consecutive time slots with $t = 0, 1, \ldots, T - 1$, the objective function, denoted by $E^{obj}$, can be calculated as

\[
E^{obj} = \sum_{t=0}^{T-1} (x[t] \cdot E^{ON}[t] + (1 - x[t]) \cdot E^{OFF}[t]),
\]

(11)

where $E^{ON}[t]$ and $E^{OFF}[t]$ are calculated as in Eqns. (9) and (10), respectively, and $x[t]$ is a binary decision variable which is set to 1 when the sBS under study is in the active mode in time slot $t$ and set to 0 when the sBS is in the sleep mode.

B. Control Problem Formulation

We first start with an offline control problem formulation in which the operation mode of the sBS under study is determined for all time slots within a control horizon, provided all information of user connections and download requests, e.g. URL and size of the requested files. In such an offline setting, the optimal value of decision variables $x[t]$’s can be determined using a simple solution method. Since the values of $C_S[t]$’s, $C_M[t]$’s, and $C_{BH}[t]$’s in Eqn. (8) can be derived from the size of each requested file, one can know the exact values of $E^{ON}[t]$’s and $E^{OFF}[t]$’s. Therefore, for time slot $t$, the value of $x[t]$ can be set to 1 if $E^{ON}[t] < E^{OFF}[t]$ and set to 0 otherwise. Although the offline solution yields optimal results, it requires the request pattern to be known before control decisions are made, which is not realistic.

In a more realistic setting for online control, the number and type of files that will be requested in future time slots cannot be known a priori, which means that values of $E^{ON}[t]$’s, $E^{OFF}[t]$’s, and $E^{ON}[t]$’s are calculated at the time of finding the optimal $x[t]$. However, as discussed in Section II-B, user profiles and information contexts including the equipment type, gender, age and occupation of each connected user can be obtained and updated by sBSs. In the online control problem, we assume that an sBS will collect the user profile/context, specified by a vector $v[t]$, at the beginning of time slot $t$, after which the operation mode of the sBS will be determined as a function of $v[t]$, while the actual energy consumption over a time slot can only be obtained at the end of the current time slot.

The aforementioned problem description fits in the framework of a contextual multi-armed bandit problem [18], [21] in which an agent needs to sequentially select from a set of actions to take in each step based on some observations in order to maximize his/her reward which is non-deterministic and unknown a-priori. For our control problem, the agent is the sBS under study, the action set consists of two different operation modes, and the reward in time slot $t$, denoted by $r[t]$, can be defined as:

\[
r[t] = \begin{cases} 
E^{OFF}[t] - E^{ON}[t], & x[t] = 1 \\
0, & x[t] = 0 
\end{cases}
\]

(12)

Notice that $r[t]$ is set to 0 when the sBS is in the sleep mode because it is not monitoring the user traffic and cannot estimate the energy saving. The goal of the control problem is then translated into finding the optimal action in time slot $t$, denoted by $x^*[t]$, such that

\[
x^*[t] = \arg \max_{x[t]} E[r[t]|\xi[t], x[t]]
\]

(13)

where \(\xi[t] = (v[0], \ldots, v[t], x[0] \ldots x[t-1], r[0] \ldots r[t-1])\) is the history of the system at the beginning of time slot $t$.

Algorithm 1: Thompson Sampling

1. Posit a-priori distributions $p(z)$, $z$ are parameters of the latent model.
2. for each time slot $t$ do
3. Sample new sets of $z$ from the posterior distribution $p(z|x(t))$
4. Receive the context $v[t]$
5. Perform the action with highest expected reward, namely $x^*[t] = \arg \max_{x[t]} E(r[t]|v[t], x[t], z)$
6. Update the model.
7. end

In general, a contextual multi-armed bandit problem is hard to solve [22] due to the partial observability of the system and the non-deterministic relationship between the
action taken and the reward. Thompson sampling [23] is a popular approach for solving the contextual multi-armed bandit problem by picking actions to balance exploration and exploitation. Generally, given the input context $v(t)$, a set of actions $x(t) \in X$, rewards $r(t)$ in $\mathbb{R}$ and the history of system $\xi[t]$, Thompson sampling is performed following the steps shown in Algorithm 1. Bayesian treatment is necessary in Thompson sampling. According to Bayes’ rule, the posterior distribution can be calculated as

$$p(z|\xi(t)) = \frac{p(\xi(t)|z)p(z)}{\int p(\xi(t)|z)p(z)dz},$$

where the $p(\xi(t)|z)$ is the likelihood distribution, $p(z)$ is a-prior distribution and $p(\xi(t))$ is the evidence. Please note that the latent model is not trivial to construct and requires the knowledge of system.

In our work, we adapt Thompson sampling to a learning agent by using a model-free Bayesian neural network (BNN) to approximate the latent model while preserving the Bayesian treatment. In this way, it relaxes the requirement of constructing a precise model and implicitly handles uncertainties in the observation properly.

### C. Solution Method to the Online Control Problem

1) Bayesian Neural Network: Bayesian neural network is a neural network with a-priori distributions instead of single fixed point estimate on its latent variables [24]. The BNN can be seen as an ensemble of a large number of neural networks with introduced variabilities on weights and biases. As a result, BNNs are robust to disturbances in the learning especially when the training set is noisy or incomplete. In addition, the over-fitting can be potentially alleviated because the implicitly introduced regularizations, which prevents BNN from over-exploiting the currently obtained observations (training sets) in the contextual bandit problem setting [25].

In BNNs, latent variables $z$ consist of weights $W$ and biases $b$. In analogy with Thompson sampling method, we posit a-priori distributions $p(z)$ on latent variables $z$, which denotes the assumption before observing the system. Here, observations are the history of system $\xi(t)$. The likelihood distribution $p(\xi(t)|z)$ in BNN can be computed as follows:

$$p(\xi(t)|z) = r[t] \log(o(x^*[t], v[t], z)), \quad (14)$$

because the BNN has been trained using historical observations. $o(\cdot)$ is the value at the output node of BNN, which corresponds to the chosen action $x^*[t]$, input context $v[t]$ and latent variables $z$. The action $x^*[t]$ is chosen to be the one producing the largest value at the output node. Please note that the likelihood here is not necessarily a probability within the region $[0, 1]$.

As we know from Bayes’ rule, in order to calculate the posterior distribution $p(z|\xi(t))$, the denominator $p(\xi(t))$ has to be computed first. However, calculating the denominator $p(\xi(t))$ requires the integral of sum over all possible latent variables, i.e.,

$$p(\xi(t)) = \int p(\xi(t)|z)p(z)dz.$$ For most cases of interest, this integral is intractable, therefore approaches are needed to approximately evaluate the posterior probability [26]. One of the most commonly used method is Markov chain Monte Carlo (MCMC), which keeps sampling a Markov chain by traversing the high probability area. However, MCMC is computationally very expensive and lack of a clear stopping criterion [27].

2) Variational Inference: In order to solve the intractable issue of Bayesian learning in neural networks by an efficient way, variational inference [26], [27] has been proposed as an alternative to MCMC. Different from MCMC, variational inference solves the optimization problem, has a better converge rate and is scalable to large problems.

Variational inference assumes that $p(z|\xi(t))$ can be approximated using a variational posterior $q(\theta|\xi)$ where $\theta$ is an unknown parameter to be found and the variational posterior distribution of latent variables can be adaptively updated by adjusting the value of $\theta$. The core idea of variational inference involves two steps. 1). positing variational posterior distributions $q(z|\theta)$ over the latent variables $z$. 2). use $q(z|\theta)$ to approximate the posterior distribution $p(z|\xi(t))$ by optimizing over its parameters $\theta$ to minimize the divergence metric (Dvgn), $\theta^* = \arg \min_{\theta} D_{\text{vgn}}(q(z; \theta), p(z|\xi(t)))$. For the divergence measurement, when the Kullback-Leibler (KL) divergence is used, the optimization problem becomes as follows:

$$\theta^* = \arg \min_{\theta} KL(q(z; \theta) \mid \mid p(z|\xi(t)))$$

$$= \arg \min_{\theta} \int_{-\infty}^{+\infty} q(z; \theta) \log \left(\frac{q(z; \theta)}{p(z|\xi(t))}\right)dz$$

$$= \arg \min_{\theta} E_{q(z; \theta)}[\log q(z; \theta) - \log p(z|\xi(t))]. \quad (15)$$

Please note that Eqn. (15) is still intractable due to the dependence on $p(z|\xi(t))$. Considering the property that can be derived from Bayes’ rule as follows:

$$\log p(\xi(t)) = KL(q(z; \theta) \mid \mid p(z|\xi(t))) + E_{q(z; \theta)}[\log p(\xi(t), z) - \log q(z; \theta)],$$

where the $\log p(\xi(t))$ is a constant with respect to the variational parameters $\theta$. Therefore, we can minimize $KL(q(z; \theta) \mid \mid p(z|\xi(t)))$ by instead of minimizing the cost function $F(\theta)$ below:

$$F(\theta) = E_{q(z; \theta)}[\log q(z; \theta) - \log p(\xi(t), z)]$$

$$= E_{q(z; \theta)}[\log q(z; \theta) - \log(p(\xi(t)|z)p(z))]. \quad (16)$$

The expectation value can be estimated by Monte Carlo integration which samples latent variables $z$ from the distribution $q(z; \theta)$. Then the estimated cost function $F(\theta)'$ can be calculated as follows:

$$F(\theta)' = \frac{1}{M_s} \sum_{s=1}^{M_s} \left[\log q(z_s; \theta) - \log(p(\xi(t)|z_s)p(z_s))\right]. \quad (17)$$

Here, $M_s$ is the cardinality of Monte Carlo integration sampling steps. $z_s$ is the sampled values of latent variables $z$ from the distribution $q(z; \theta)$ at the sampling step $s$. The $M_s$ is usually a small number compared to the sampling count in MCMC. Eqn. (17) is tractable because each term can be calculated.

3) Proposed Context-Driven Online Power Mangement Framework: To solve the aforementioned contextual multi-armed bandit problem, we propose to train a BNN using the variational inference method. As shown in Fig. 3, a BNN has a set of latent variables, $z = \{z_1, z_2, \ldots \}$, which includes edge weights $W_{ij}$ from the $i$-th neuron at the layer $l$ to the $j$-th neuron at the layer $l + 1$ and biases $b^l_i$ at layer $l$. The input nodes of the network correspond to the elements of $v[t]$, which are the input context of all connected users at the studied
sBS. More concretely, each kind of user profiles is categorized and encoded into one-hot encoding. Each bit in the one-hot encoding corresponds to one neuron at the input layer. Then, each connected user is binned into input nodes according to their categories. Whereas the two output nodes produce the probabilities that either of the two operation modes yields lower energy consumption, i.e. \( p(x[t] = 0|v[t], z) \) and \( p(x[t] = 1|v[t], z) \), respectively.

As discussed before, different from other well-known neural networks, such as artificial neural networks (ANNs) in which \( z \) take deterministic values, a BNN assumes that latent variables \( z \) have a-priori distributions \( p(z) \) and posterior distributions \( p(z|\xi_t) \). While the a-prior distribution is independent of the history of system and remains unchanged, the variational posterior distribution is updated at the end of each time slot after the action is taken and the reward is evaluated. In our implementation, both a-prior distributions \( p(z) \) and variational posterior distributions \( q(z; \theta) \) are assumed to be Gaussian distributions. For \( q(z; \theta) \), \( \theta \) denotes the mean value and standard deviation of the Gaussian distribution.

The proposed solution framework is summarized in Algorithm 2. After initializing the prior and variational posterior distribution of \( z \), the BNN will iteratively select the operation mode of the sBS at the beginning of each time slot and update the variational posterior distributions at the end of each time slot using forward propagation and backward propagation operations [28]. In time slot \( t \), the proposed solution proceeds as follows. First, at the beginning of the time slot, given the value of \( v[t] \) as inputs to the BNN, the value of \( x[t] \) is determined by forward propagating through the network with respect to a sampling of \( z \) and choosing the value corresponding to the output node with a larger output probability. After that, the operation mode is selected according to the \( x[t] \) value found. At the end of time slot \( t \), the value of parameter \( \theta \) is updated, which in turn modify the distribution for sampling latent variables in \( z \). We update \( \theta \) using a stochastic gradient descent approach which tries to adjust the value of \( \theta \) in the opposite direction of the gradient of a loss function \( L_\theta[t] \) through back propagation. Base on the Eqn. (14) and Eqn. (17), \( L_\theta[t] \) can be computed as follows

\[
L_\theta[t] = \frac{1}{M_s} \sum_{i=1}^{M_s} (\log q(z_i; \theta) - r[t] \log \alpha_i - \log p(z_i)), \tag{18}
\]

where \( z_1, \ldots, z_{M_s} \) are \( M_s \) samples of \( z \) according to probability distribution function \( q(z; \theta) \), \( \alpha_1, \ldots, \alpha_{M_s} \) are the output values on the output node corresponding to the selected \( x[t] \) in the \( M_s \) sampled networks, and \( r[t] \) is the reward defined in Eqn. (12). The updated \( \theta \) will then be used in time slot \( (t+1) \).

**Algorithm 2: Variational Inference BNN Agent**

1. Posit prior distributions \( p(z) \).
2. Posit variational posterior distributions \( q(z; \theta) \).
3. For each time slot \( t \) do
   4. Sample \( z \) from \( q(z; \theta) \).
   5. Receive the context \( v[t] \).
   6. Perform forward propagation for input \( v[t] \). Pick action \( x^*[t] = \arg \max_{x[t]} p(x[t]|v[t], z) \).
   7. Set operation mode according to \( x^*[t] \) and evaluate reward \( r[t] \) as in Eqn. (12).
   8. Use Monte Carlo integration to calculate the cost function \( L_\theta[t] \) in Eqn. (18).
   9. Perform backward propagation of the gradient of \( L_\theta[t] \) with respect to \( \theta \).
   10. Update \( \theta \) according to \( \theta \leftarrow \theta - \alpha_L \nabla L_\theta[t] \) where \( \alpha_L \) is the learning rate.
4. end

**IV. EXPERIMENTAL RESULTS**

Similar to references [29]–[31], we use the MovieLens 1M dataset [32] to characterize the pattern of mobile users’ download requests. The MovieLens 1M dataset contains about 1 million ratings of 3952 movies from 6040 users within the year 2000-2003. Each rating consists the user ID, movie ID and a timestamp along with the user’s gender (2 categories), age (7 categories) and occupation (20 categories). Therefore, we treat each movie as a file entry stored in caches in MBSs and sBSs and each rating as a download request for a file from the UE with a profile described in the dataset. This approach is reasonable since each rating process is usually done after downloading and watching the movie. Our assumption is that users with similar profiles would request similar files. Time slots with 15 minutes each are considered to cover the first 365 days in the dataset. Specifications of the two-tier cellular network are given in Table I. \( BW \) is the nominal channel bandwidth in the system. In practice, the effective channel bandwidth of an sBS can be approximated as \( BW \) while the effective bandwidth of an MBS is calculated as \( (1 - O_M) \cdot BW \) where \( O_M \) is the portion of overhead [20]. To estimate the SINR from a user to a base station, the path-loss for an sBS and an MBS are 30.6 + 36.7 log\(_{10}\)\( (d) \) and 35.3 + 37.6 log\(_{10}\)\( (d) \) in dB, respectively [11], where \( d \) is the distance specified in kilometers from a UE to a base station. The users’ positions are uniformly generated within the service range of an sBS. The interference is estimated based on the signal strength from three nearest base stations. The noise power is set to \( \sigma^2 = -95 dBm \) [11]. We adopt a least frequently used replacement strategy in caches in sBSs and MBSs which replaces \( p_u^S \) and \( p_u^M \) percentage of entries every day. \( N_M \) and \( N_S \) are the maximum number of files an sBS and an MBS can cache, respectively. Each file is assumed to have a size of 6 MB.

The user context, \( v[t] \), is comprised of the count of connected users that fall in each category of gender, age, and occupation, resulting in a BNN with 29 input nodes. We
TABLE I
SIMULATION PARAMETERS.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Channel</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BW</td>
<td>10MHz</td>
<td>OM</td>
<td>28.5% [20]</td>
</tr>
<tr>
<td>λ_M</td>
<td>0.001/m²</td>
<td>λ_S</td>
<td>0.1/m²</td>
</tr>
<tr>
<td>Power</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P_M</td>
<td>46dBm [12]</td>
<td>P_S</td>
<td>724.6W [11]</td>
</tr>
<tr>
<td>P_sleep</td>
<td>3.87W [11]</td>
<td>w_CN</td>
<td>70nW/bps [33]</td>
</tr>
<tr>
<td>Cache</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P_c</td>
<td>2%</td>
<td>P_s</td>
<td>10%</td>
</tr>
<tr>
<td>T_c</td>
<td>24hr</td>
<td>T_s</td>
<td>24hr</td>
</tr>
<tr>
<td>N_M</td>
<td>1000</td>
<td>N_S</td>
<td>100</td>
</tr>
</tbody>
</table>

Fig. 4. Cumulative energy consumption to serve requests from connected users of the sBS under study.

implement two hidden layers in the BNN with 200 neurons in each of the hidden layer. The two output nodes of the BNN correspond to active mode and sleep mode, respectively. Rectified linear units are used as the activation function for the input layer and hidden layers while softmax activation is used in the output layer. Gaussian distributions with a mean value of 0 and a standard deviation of $10^{-3}$ are assumed to be the prior distributions of all edge weights. The variational posterior distributions on weights and biases are also assumed to be Gaussian distributions parameterized by the mean value and the standard deviation, which are updated in each time slot. $M_e$ in Eqn. (18) is set to 5 to balance between exploration and exploitation. Stochastic gradient descent is used with mini-batches of 10 data points each with a learning rate of 0.001.

The cumulative energy consumption as defined in Eqn. (11) and cumulative regret associate with one sBS in one year is shown in Fig. 4 and Fig. 5, respectively. The regret is defined as the difference in cumulative energy consumption between the offline optimal solution as discussed in Section III-B and the online solutions shown in the figure. Our proposed variational inference BNN agent is compared with four baseline algorithms, namely, (1) always-active, (2) 10%-greedy-neural, (3) 30%-greedy-neural, and (4) 10%-greedy-simple. In the “always-active” scheme, the sBS is never switched to the sleep mode. The three other baselines adopt the well-known $\epsilon$-greedy learning strategy [34] in which the agent takes a random action with a probability $\epsilon$ in each step and takes the best action generated by the algorithm with a probability of $(1 - \epsilon)$. In “10%-greedy-neural” and “30%-greedy-neural”, the same neural network structure as the proposed BNN is used without any Bayesian assumption. To avoid over-exploration at later stages, the $\epsilon$ is set to decrease by half every 100 days. “10%-greedy-simple” is a simple baseline that does not consider any user context but only choose the best action according to the average reward in the history. As can be seen from Fig. 4, the energy consumption by using the proposed agent is 606 MJ, achieving 18.7% total energy reduction compared to the “always active” baseline (745 MJ). Furthermore, from Fig. 5, it can be observed that the regret of the proposed agent is nearly constant in later time slots while all four baselines see a linear increase of regret over time. Fig. 6 shows the relationship between total energy consumption and sBS/MBS density ratio $\frac{\lambda_S}{\lambda_M}$. Total energy consumption reduction can be achieved at 37.9%, 45.3% and 46.9% when the density ratio increases from 100 to 400, 800 and 1000, respectively. It can be seen that the proposed BNN agent can reduce the total energy consumption significantly in a scenario with a high density deployment of sBSs.

The relationship between the actual decision of the sBS and
the number of cache hits in the sBS if it is active is shown in Fig. 7 for the first 5000 time slots. As can be seen from the figure, after an initial “warm-up” period, the sBS will only be active when there are a relatively large number of cache hits, which matches our expectation, suggesting that our BNN agent can accurately interpret the user context and predict whether an active sBS will be energy efficient.

V. CONCLUSIONS

In this paper, we investigate how to improve the energy efficiency in a cache-enabled two-tier HetNet. An sBS is controlled to switch between the “active mode” and the “sleep mode” to reduce energy consumption driven by connected users’ contexts. The online energy minimization problem is carefully formulated into a contextual multi-armed bandit problem, and a variational inference BNN agent is proposed. With the proposed framework, an sBS can achieve a sub-linear increase of regret in energy consumption over time and reduce energy consumption by 46.9% for high density deployment scenario in one year comparing with the “always-active” baseline.

REFERENCES