A Collaborative Learning Based Approach for Parameter Configuration of Cellular Networks

Jie Chuai*, Zhitang Chen*, Guochen Liu*, Xueying Guo†, Xiaoxiao Wang†, Xin Liu†,
Chongming Zhu‡, and Feiyi Shen‡
*Noah's Ark Lab, Huawei Technologies, China
†Department of Computer Science, University of California, Davis, CA, USA
‡Huawei Technologies, China
Email: *chuaijie@huawei.com, *chenzhitang2@huawei.com, *liuguochen1@huawei.com, †guoxueying@outlook.com,
†xxwa@ucdavis.edu, †xinliu@ucdavis.edu, ‡zhuchongming@huawei.com, ‡shenfeiyi@huawei.com

***Abstract*—Cellular network performance depends heavily on the configuration of its network parameters. Current practice of parameter configuration relies largely on expert experience, which is often suboptimal, time-consuming, and error-prone. Therefore, it is desirable to automate this process to improve the accuracy and efficiency via learning-based approaches. However, such approaches need to address several challenges in real operational networks: the lack of diverse historical data, a limited amount of experiment budget set by network operators, and highly complex and unknown network performance functions. To address those challenges, we propose a collaborative learning approach to leverage data from different cells to boost the learning efficiency and to improve network performance. Specifically, we formulate the problem as a transferable contextual bandit problem, and prove that by transfer learning, one could significantly reduce the regret bound. Based on the theoretical result, we further develop a practical algorithm that decomposes a cell's policy into a common homogeneous policy learned using all cells' data and a cell-specific policy that captures each individual cell's heterogeneous behavior. We evaluate our proposed algorithm via a simulator constructed using real network data and demonstrates faster convergence compared to baselines. More importantly, a live field test is also conducted on a real metropolitan cellular network consisting 1700+ cells to optimize five parameters for two weeks. Our proposed algorithm shows a significant performance improvement of 20%.**

I. Introduction

In this paper, we study the parameter optimization problem in cellular networks. In a cellular network, a cell has hundreds of parameters to configure, including parameters for access control, handover, radio resource management, etc., [1]. Furthermore, the number of such parameters increases as more advanced features are deployed in the networks. These parameters have a large impact on network performance and should be configured appropriately based on the time varying environment [2]. However, this is by no means an easy task as the best parameter setting for a specific cell depends on many factors including user number distribution, user location distribution, user demand pattern, channel quality, neighboring cell configuration, etc., which could vary significantly over time and across cells. Furthermore, the relation between the network performance and the parameter configuration is also highly complex and beyond the capability of analytical models available.

In the current practice, the parameters are generally configured by field engineers based on human experience. To configure one parameter, an engineer first observes a few dimensions of states of a cell (e.g., the user density, signal quality), and sets an initial value of the parameter. After that, he/she observes the performance by collecting and analyzing the logs of the cell and decides according to his/her experience if the parameter value should be further tuned. This traditional approach is time-consuming and suboptimal. Firstly, human experts can only decide the parameter value based on a few dimensions of state information, which might not be sufficient to fully describe the exact status of the cell. Secondly, human experts only manage to tune a few parameters each time (generally one or two, otherwise it would be too complicated for human analysis). However, there are a large number of parameters to be configured in each cell [3], [4], and thus it is almost impossible for human experts to adjust these parameters within a reasonable time. Furthermore, parameters are sometimes coupled, which means they should be adjusted jointly. Due to these difficulties, in the current cellular networks, cell parameters are often set to their default values, and are not optimized at all.

A machine learning or data-driven approach is highly desirable for this problem to automate the parameter optimization process and improve the network performance. To realize that, we need to address the following challenges: firstly, quite often we face a cold start problem, which means that there is no diverse historical parameter data of each cell to learn the corresponding model/policy because most cells use the default configuration. Therefore, we have to learn the model/policy in a trial-and-error fashion that balances the exploration and exploitation trade-off; secondly, exploring the parameter space is deemed costly and risky from network operators' perspective and thus they tightly control the budget of exploration; finally, the network performance is a highly complex function of a cell's state and the configured parameters, which requires a large amount of data to learn.

To address the above challenges, we propose a collaborative learning based approach. The key idea is to leverage the data of different cells to boost the learning and decision-making of each cell. More specifically, our contributions are as follows:

(1) We formulate the parameter optimization problem as a transferable contextual multi-armed bandit problem. We prove the transfer efficiency of the proposed approach where we can significantly reduce the regret bound especially when the number of cells is large, as in our system. (2) To bridge the gap between theory and practice, we develop a practical algorithm to decompose a cell's model/policy for parameter optimization into a homogeneous and a heterogeneous model/policy, where the homogeneous model/policy is learned using all cells' data and the heterogeneous one is learned using each cell's own data. Learning the common model/policy is the key step towards transfer learning and faster policy convergence. Compensating the common policy with the cell's individual policy aims to respect dissimilarities amongst different cells. (3) We conduct a live field test in a metropolitan cellular network with 1700+ cells and observe a significant network performance improvement of over 20%.

The rest of this paper is organized as follows: In Sec. II, we present a motivating example of cellular network handover parameter optimization. In Sec. III, we discuss related work on data-driven approaches on cellular network optimization. We formulate the cellular network parameter optimization problem in Sec. IV and present the transferable contextual bandit formulation in Sec. V. In Sec. VI, we present our collaborative learning approach. We report experiments based on a simulator and a live test in a metropolitan cellular network in Sec. VII and conclude in Sec. VIII.

II. Example of Parameter Optimization

In this section, we use the parameter A2-threshold-RSRP in the Long-Term Evolution (LTE) standard [5] as an example of network parameter configuration. A2-threshold-RSRP is a parameter used during LTE handover process. When camping on a cell, a User Equipment (UE) keeps monitoring the signal quality (the Reference Signal Received Power, or RSRP value) of the serving cell. Once the RSRP value of the serving cell is worse than the A2-threshold-RSRP, UE starts to send measurement reports such that the serving cell can identify a target cell and prepare for handover.

The configuration of A2-threshold-RSRP impacts the throughput of the edge UEs (i.e., UEs at the boundaries of the cells). If this value is too small, measurement report and handover process cannot be triggered even when the signal quality of serving cell becomes poor, and the edge UE suffers from poor signal quality or even call drop. If this value is too large, then the UE will be triggered to send measurement reports too frequently, which occupies valuable bandwidth resources, and decreases the UE's data throughput. Therefore, to optimize the throughput of edge UEs, this A2-threshold-RSRP parameter should be carefully tuned.

Setting A2-threshold-RSRP to the best value requires taking into consideration of many factors including the traffic load of a cell, the number of (active) users, channel quality, etc., and thus is not easy. Machine learning or data-driven approach comes to the rescue and we propose our collaborative learning based framework to tackle general tasks of parameter optimization for cellular networks.

III. Related Work

In wireless communications, data-driven based approach is drawing increasing attentions recently [6], [7]. We note that the network parameter optimization can be formulated as a general reinforcement learning problem. Some recent work thus employs reinforcement learning algorithms. For example, in [8], authors design a Q-learning-based power allocation mechanism in a MIMO-NOMA cellular system. However, the extremely large state space and action set in large scale network can lead to the prohibitive computational complexity and convergence time for general RL algorithms. Some literature addresses this issue by multi-agent solutions. In [9], authors study into the traffic offloading in hyper-cellular networks (HCN), and propose a distributed Q-learning solution for network control. Yet, when the network scale is large, the heuristic multi-agent solutions can still suffer from relatively long convergence time and a lack of theoretical performance guarantees. Some recent work addresses this by degenerating the problem to bandit formulation [10], [11].

Multi-armed bandits (MABs) [12] is a sequential decision problem, which is a special case of reinforcement learning, where the reward of a decision is immediately observed. Well-known algorithms include UCB [13] and Thompson Sampling [14]. Further, with side information provided before decision making, the classic MAB is generalized to contextual bandit problems [15], [16].

In wireless networks, bandit algorithms have been applied in several application scenarios. In downlink scheduling problems, the bandit algorithms, especially Whittle Index policy, have been employed to deal with the curse of dimension of Markov Decision Process [17], [18], [19]. In vehicular networks, bandits have been applied to design learning-based task offloading [20], [21]. Further, emerging literature demonstrates the potential of contextual bandit in network configuration problems [10].

Transfer learning and multi-task learning are powerful methods that improve the learning efficiency by using the data samples from multiple sources [22], [23]. They have been successfully applied to many fields, such as text sentiment classification [24] and image classification [25]. In [26], the authors re-weight the instances in multi-source to address both marginal and conditional distribution differences between the source and target domains. In [27], the authors consider transfer learning via dimensionality reduction. They learn a low-dimensional latent feature space where the distributions between the source domain data and the target domain data are the same or close to each other. Similarly, [28] proposes a feature transformation approach for domain adaptation called Transfer Component Analysis (TCA) to discover common latent features that have the same marginal distribution across the source and target domains while maintaining the intrinsic structure of the original domain data.

IV. Problem Formulation

In this section, we present the problem formulation for the cellular network parameter optimization task. We consider a

cellular network of N cells. Consider a time-slotted system over a period of T time slots. At time step t, where $t \in \{1, 2, \ldots, T\}$, state information of all the cells are revealed. The state of Cell i is denoted by a vector $\mathbf{s}_t^i \in \mathcal{S}$, where $\mathcal{S}$ is the state space. The cell state may include the number of users in the cell, the average channel quality indicator (CQI) and the traffic load, etc., at time t. After observing the state, a parameter value $\mathbf{a}_t^i \in \mathcal{A}$ is chosen as the configuration for Cell i at time step t. The parameters could be related to handover (as described in Sec. II), spectrum allocation, power control, user scheduling, etc. Note that $\mathbf{a}_t^i$ is a vector, because we may need to optimize multiple parameters simultaneously. The network performance of Cell i at time t, denoted by $y_t^i \in \mathbb{R}$, is observed after $\mathbf{a}_t^i$ is configured. The performance metric could be the cell data throughput, edge user throughput, packet delay, etc. In this work, we assume y_t^i depends on the configured parameter value and the cell state, and is formulated as $y_t^i = f_i(\mathbf{s}_t^i, \mathbf{a}_t^i) + \xi_t^i$, where ξ_t^i is a random noise with zero mean. Note that the function f_i is a cell-specific function for each cell i. In this problem setting, the state space $\mathcal{S}$, the parameter space $\mathcal{A}$ and the performance target are the same for all the cells.

In practice, the network operator controls the frequency of parameter adjustment which may range from once every few hours to once every few days, depending on the network management infrastructures and policies of the operator.

The expected cumulative performance of the whole network over the period T is defined as

$$Y_T = \mathbb{E}\left[\sum_{t=1}^{T}\sum_{i=1}^{N} y_t^i\right]. \tag{1}$$

The goal of network parameter optimization is to maximize (or minimize, depending on the performance metric chosen) the expected cumulative performance Y_T.

Without loss of generality, we consider a performance metric that needs to be maximized (e.g., the cell data throughput). The regret of Cell i at time t is defined as

$$r_t^i = \left\{\max_{\mathbf{a}' \in \mathcal{A}} f_i(\mathbf{s}_t^i, \mathbf{a}')\right\} - f_i(\mathbf{s}_t^i, \mathbf{a}_t^i).$$

That is, it is the gap between $\mathbb{E}[y_t^i]$ and the optimal expected performance for Cell i at time t. The regret can also be defined similarly for performance metric that needs to be minimized (e.g., ratio of edge users). Therefore, the goal of network parameter optimization is now transformed to minimizing the cumulative regret over period T, i.e.,

$$\min_{\mathbf{a}_t^i : \forall i,t} \sum_{t=1}^{T}\sum_{i=1}^{N} r_t^i. \tag{2}$$

The problem is a contextual multi-armed bandit problem where multiple agents (i.e., cells) make decisions concurrently. In the next section, we will first introduce the background on contextual bandit problem and then discuss a new setting called transferable contextual bandit problem, followed by a theoretical analysis on how transferring data among cells can improve the bandit convergence performance.

V. Transferable Contextual Bandit

A. Contextual Bandit

Contextual Bandit is a powerful tool to solve the optimization of a complex system subject to a varying environmental condition [29]. For example, which movie we should recommend given a user's gender, age, list of movies watched, etc. Usually the contextual bandit problem can be formulated as a sequential optimization procedure where in each step, we receive a context $\mathbf{s}$ from the environment or the system and we are required to perform an action $\mathbf{a}$. Once the action is executed, the environment feeds back a reward y, which could be a random sample drawn from certain unknown distribution. The loop is repeated until we reach the limit of budget for actions. The key challenge here is to trade off between exploration and exploitation, given that we only have a limited budget to take actions. Exploitation is to choose the one from all actions made so far that yields the highest reward and exploration means getting more information of other actions that have not been tried. Too much exploitation might lead to sub-optima whereas too much exploration could also be risky because we might eventually be far from the optima. There are many works proposed to address this trade-off such as Upper Confidence Bound (UCB) and its variants [15], [30], Thompson Sampling [31], [32], ϵ-greedy, etc., for the context-free problems.

For the scenarios where the context is available, this extra context information is possible to make faster convergence to the optimal policy and CGP-UCB algorithm proposed in [29] considers playing a game for a sequence of T steps and at each step, the reward/payoff is formulated as $y_t = f(\mathbf{s}_t, \mathbf{a}_t) + \xi_t$, where $f : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is an unknown function and ξ_t is a random noise with zero mean. Since f is not known, there is no guarantee that at each step, we choose the optimal action and thus we have the regret $r_t = \{\max_{\mathbf{a}' \in \mathcal{A}} f(\mathbf{s}_t, \mathbf{a}')\} - f(\mathbf{s}_t, \mathbf{a}_t)$ and the cumulative regret $R_T = \sum_{t=1}^{T} r_t$. The goal here is to learn an algorithm to achieve sub-linear contextual regret, i.e., $R_T/T \to 0$ for $T \to \infty$. A nonparametric regression method called Gaussian Process (GP) [33] is proposed to learn the function f. A Gaussian process $\mathbf{f}_T = [f_1, \cdots, f_T]$ can be regarded as a vector of random variables, each of which is regarded as a function $f(\mathbf{x}_t)$ for $\mathbf{x}_t \in \mathcal{X}$ where in the contextual bandit setting, $\mathcal{X} = \mathcal{S} \times \mathcal{A}$. This vector is assumed to follow a multivariate Gaussian distribution $\mathcal{N}(\mathbf{0}, \mathbf{K}_T)$ with zero mean vector $\mathbf{0}$ and a covariance matrix $\mathbf{K}_T$ where $[\mathbf{K}_T]_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$ and k is a positive definite kernel function associated with a Reproducing Kernel Hilbert Space (RKHS) [34]. Note that usually we assume that f_t is not directly observed but its noisy version $y_t = f(\mathbf{x}_t) + \xi_t$ is available. Suppose we get another $\mathbf{x}^*$ and would like to infer its corresponding f^*. It is easy to show that the posterior distribution of f^* is also a Gaussian distribution with the following mean and variance:

$$\mu_T(\mathbf{x}^*) = \mathbf{k}_T(\mathbf{x}^*)^T \left(\mathbf{K}_T + \sigma^2\mathbf{I}\right)^{-1} \mathbf{y}_T, \tag{3}$$

$$\sigma_T^2(\mathbf{x}^*) = k(\mathbf{x}^*, \mathbf{x}^*) - \mathbf{k}_T(\mathbf{x}^*)^T \left(\mathbf{K}_T + \sigma^2\mathbf{I}\right)^{-1} \mathbf{k}_T(\mathbf{x}^*), \tag{4}$$

where $\mathbf{k}_T(\mathbf{x}^*) = [k(\mathbf{x}^*, \mathbf{x}_1), \cdots, k(\mathbf{x}^*, \mathbf{x}_T)]^T$ and σ is the standard deviation of ξ_t.

The CGP-UCB algorithm [29] is proposed as follows to choose an action at step t:

$$\mathbf{a}_t = \arg\max_{\mathbf{a}\in\mathcal{A}} \mu_{t-1}(\mathbf{s}_t, \mathbf{a}) + \beta_t^{1/2}\sigma_{t-1}(\mathbf{s}_t, \mathbf{a}), \tag{5}$$

where β_t is an appropriate constant and μ_{t-1} and σ_{t-1} are the posterior mean and standard deviation learned from the samples up to step t.

It has been shown in [29] that by flexibly combining kernels over the context and the action, CGP-UCB algorithm can be used to address various problems and it is guaranteed that with high probability, $R_T/T \to 0$ if $T \to \infty$.

B. Transferable Contextual Bandit

CGP-UCB is a powerful algorithm with nice sub-linear convergence. However, it solves a single agent/domain problem. In many real cases, we are to optimize multiple complex systems each of which has limited budget to explore different actions, e.g., we are to optimize multiple parameters for a cellular network and usually the operator requires convergence within two weeks. That means even with one parameter configuration per day (which is common in the current industrial practice), we only have up to 14 rounds to explore actions which is far from sufficient compared to the huge action space.

A simple application of CGP-UCB to each of the system or agent is not expected to converge within the limited budget and thus in this section, we propose a transferable and concurrent contextual bandit algorithm that leverages all experiences obtained in each system to speed up the convergence of each system. Furthermore, as a theoretical contribution, we also derive the speed up factor for the transferable CGP-UCB algorithm.

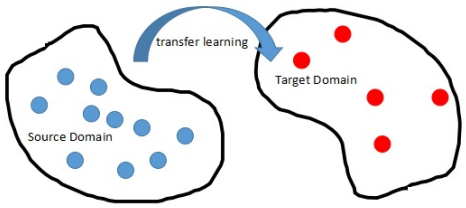


Fig. 1. Transfer Learning

As illustrated in Fig. 1, transfer learning is a powerful tool to generalize knowledge learned from one domain (the source domain) to another (the target domain). The advantage of transfer learning is that it can improve the model accuracy in the target domain, especially for those cases where in the target domain, there is little or even no labeled data.

There are several different settings well-studied in transfer learning literature, depending on whether $P_{\mathcal{S}}(Y|X) = P_{\mathcal{T}}(Y|X)$, but $P_{\mathcal{S}}(X) \neq P_{\mathcal{T}}(X)$, or $P_{\mathcal{S}}(Y|X) \neq P_{\mathcal{T}}(Y|X)$, but $P_{\mathcal{S}}(X) = P_{\mathcal{T}}(X)$, where $\mathcal{S}$ denotes the source domain and $\mathcal{T}$ denotes the target domain.

In this paper, we focus on the second setting where the tasks in the source and the target domains are different but there exists such a transformation $\tau(\cdot)$ that

$$P_{\mathcal{S}}(Y|\tau(X)) = P_{\mathcal{T}}(Y|\tau(X)) \tag{6}$$

Once we are able to find such a transformation function $\tau(\cdot)$, then we can share data from the source domain to the target domain to speed up the learning process in the target domain.

Let us get back to our bandit problem. We assume that there exists such a transformation and is known. In the rest of this section, $\mathbf{x}$ denotes the transformed feature that satisfies Eq. 6. Suppose we concurrently optimize k complex systems. Without loss of generality, let us denote the task for the complex system i as the target task $\mathcal{T}$ and all other complex systems together as the source task $\mathcal{S}$. The basic idea of the transferable contextual bandit is to transfer the experience/data from the source task $\mathcal{S}$ to the target task $\mathcal{T}$. We aim to analyze how much speed-up we can achieve through transfer learning.

First of all, we define the following "collectively exciting" property of samples in the RKHS which is a relaxed definition of "exciting" in [35]. We use $\phi(\mathbf{x})$ to denote the kernel mapping of $\mathbf{x}$ for some positive definite kernel function $k(\mathbf{x}, \mathbf{x}')$.

Defintion 1. *We say that the dataset $X = \{\phi(\mathbf{x}_i)\}_{i=1}^N$ is collectively exciting if $\frac{1}{N}\sum_i \phi(\mathbf{x}_i)\phi(\mathbf{x}_i)^T$ is positive definite, i.e., there exists an α such that*

$$\mathbf{0} \prec \alpha\mathbf{I} \preceq \frac{1}{N}\sum_i \phi(\mathbf{x}_i)\phi(\mathbf{x}_i)^T \preceq \beta\mathbf{I} \prec \infty.$$

Based on this assumption, we can prove the transfer efficiency as follows:

Theorem 1. *Denote by $D_{\mathcal{T}} = \{\phi_{\mathcal{T}}(\mathbf{x}_i)\}_{i=1}^{N_{\mathcal{T}}} \cup \{y_{\mathcal{T},i}\}_{i=1}^{N_{\mathcal{T}}}$ the dataset with $N_{\mathcal{T}}$ samples for a target domain and $D_{\mathcal{S}} = \{\phi_{\mathcal{S}}(\mathbf{x}_j)\}_{j=1}^{N_{\mathcal{S}}} \cup \{y_{\mathcal{S},j}\}_{j=1}^{N_{\mathcal{S}}}$ the dataset with $N_{\mathcal{S}}$ samples from the source domain. Assume that $\{\phi_{\mathcal{T}}(\mathbf{x}_i)\}_{i=1}^{N_{\mathcal{T}}}$ and $\{\phi_{\mathcal{S}}(\mathbf{x}_j)\}_{j=1}^{N_{\mathcal{S}}}$ are both collectively exciting. Pick $\delta \in (0,1)$ and set $\beta_t = 2\log(|D_{\mathcal{T}}|\pi_t\delta)$, where $\Sigma_{t>1}\pi_t^{-1} = 1, \pi_t > 0$. Then, the regret r_t with the model learned from $D_{\mathcal{T}} \cup D_{\mathcal{S}}$ for the task $\mathcal{T}$ at round t is bounded by $2\beta_t^{1/2}\sigma_{\mathcal{T}+\mathcal{S},t-1}(\mathbf{x}_t)$, where*

$$\sigma_{\mathcal{T}+\mathcal{S},t-1}(\mathbf{x}_t) \leq \underbrace{\sqrt{\frac{1+\sigma^{-2}N_{\mathcal{T}}\beta}{1+\sigma^{-2}N_{\mathcal{S}}\alpha+\sigma^{-2}N_{\mathcal{T}}\beta}}}_{\gamma}\sigma_{\mathcal{T},t-1}(\mathbf{x}_t),$$

where $\sigma_{\mathcal{T}+\mathcal{S},t-1}$ is the conditional standard deviation given by a model learned from the source and target domain data and $\sigma_{\mathcal{T},t-1}$ is the conditional standard deviation given by a model learned from only the target domain data.

Sketch of Proof Suppose we use a degenerated kernel [36]. According to the Woodbury-Sherman-Morrison formula, it can be easily shown that Eq. 4 can be reformulated as

$$\sigma_T^2(\mathbf{x}_t) = \phi(\mathbf{x}_t)^T\left(\sigma^{-2}\mathbf{\Phi}\mathbf{\Phi}^T + \mathbf{I}\right)^{-1}\phi(\mathbf{x}_t),$$

where $\mathbf{\Phi} = [\phi(\mathbf{x}_t), \cdots, \phi(\mathbf{x}_T)]$. Denote by $\sigma^2_{\mathcal{T},t-1}(\mathbf{x}_t)$ the estimation error of $f(\mathbf{x}_t)$ with a model trained from only the

target task data up to time $t-1$ and $\sigma^2_{\mathcal{T}+\mathcal{S},t-1}(\mathbf{x}_t)$ the estimation error with a model trained from both the target and source tasks data. Denote $\Sigma_{\mathcal{T}} := \left(\sigma^{-2}\mathbf{\Phi}_{\mathcal{T}}\mathbf{\Phi}_{\mathcal{T}}^T + \mathbf{I}\right)$ and $\Sigma_{\mathcal{T}+\mathcal{S}} := \left(\sigma^{-2}\mathbf{\Phi}_{\mathcal{T}}\mathbf{\Phi}_{\mathcal{T}}^T + \sigma^{-2}\mathbf{\Phi}_{\mathcal{S}}\mathbf{\Phi}_{\mathcal{S}}^T + \mathbf{I}\right)$. We have $\sigma^2_{\mathcal{T},t-1}(\mathbf{x}_t) = \phi(\mathbf{x}_t)^T\Sigma_{\mathcal{T}}^{-1}\phi(\mathbf{x}_t)$ and $\sigma^2_{\mathcal{T}+\mathcal{S},t-1}(\mathbf{x}_t) = \phi(\mathbf{x}_t)^T\Sigma_{\mathcal{T}+\mathcal{S}}^{-1}\phi(\mathbf{x}_t)$. Using the assumption of *Collectively Exciting*, there exist α and β such that $\sigma^{-2}\Sigma_{\mathcal{T}} \preceq \sigma^{-2}N_{\mathcal{T}}\beta\mathbf{I} \prec \infty$ and $\mathbf{0} \prec \sigma^{-2}N_{\mathcal{S}}\alpha\mathbf{I} \preceq \sigma^{-2}\Sigma_{\mathcal{S}}$. Consequently, we obtain

$$\begin{aligned}\Sigma_{\mathcal{T}+\mathcal{S}} &= \Sigma_{\mathcal{T}} + \sigma^{-2}\sum_j \phi_{\mathcal{S}}(\mathbf{x}_j)\phi_{\mathcal{S}}(\mathbf{x}_j)^T \succeq \Sigma_{\mathcal{T}} + \sigma^{-2}N_{\mathcal{S}}\alpha\mathbf{I} \\ &\succeq \Sigma_{\mathcal{T}} + \frac{\sigma^{-2}N_{\mathcal{S}}\alpha}{1+\sigma^{-2}N_{\mathcal{T}}\beta}\Sigma_{\mathcal{T}} \\ &= \left(1 + \frac{\sigma^{-2}N_{\mathcal{S}}\alpha}{1+\sigma^{-2}N_{\mathcal{T}}\beta}\right)\Sigma_{\mathcal{T}}.\end{aligned}$$

Obviously, we have

$$\Sigma^{-1}_{\mathcal{T}+\mathcal{S}} \preceq \frac{1+\sigma^{-2}N_{\mathcal{T}}\beta}{1+\sigma^{-2}N_{\mathcal{S}}\alpha+\sigma^{-2}N_{\mathcal{T}}\beta}\Sigma^{-1}_{\mathcal{T}},$$

and thus

$$\sigma_{\mathcal{T}+\mathcal{S},t-1}(\mathbf{x}_t) \leq \sqrt{\frac{1+\sigma^{-2}N_{\mathcal{T}}\beta}{1+\sigma^{-2}N_{\mathcal{S}}\alpha+\sigma^{-2}N_{\mathcal{T}}\beta}}\sigma_{\mathcal{T},t-1}(\mathbf{x}_t).$$

According to Theorem 1, we can see that by leveraging data from other systems, we can reduce the regret bound at each round by a discounting factor $\gamma < 1$ which leads to faster convergence. The speed-up is more significant if $N_{\mathcal{T}} \ll N_{\mathcal{S}}$ which is possible in some cases. For example, in the cellular parameter optimization task, we can easily have thousands of cells. Note that the speed-up is also determined by α. If the dataset $D_{\mathcal{S}}$ lacks diversity, then α is small, resulting a smaller speed-up. That means when we transfer the experience from other systems, it is better to transfer data with sufficient diversity.

Note that, this theoretical result is obtained with the assumption that in the source and the target domain, the underlying functions f to learn are identical, i.e., there exists a $\tau(\cdot)$ such that Eq. 6 is valid. In practice, $\tau(\cdot)$ needs to be learned. In the next section, we will discuss how to bridge the gap between the theory and the practice.

VI. A Collaborative Contextual Multi-Armed Bandit Framework

In Sec. V, we have shown that transferring data between the cells can speed up the convergence of individual cells. In this section, we propose a Collaborative Contextual Multi-Armed Bandit framework to concurrently optimize parameters of all the cells, see Fig. 2.

More specifically, we propose a method to learn the function f_i for any Cell i by utilizing available data from all the cells. The main idea of our method is to divide the function f_i into a common function and a cell-specific function. The common function captures the common behavior of all cells, thus can be learned by leveraging different cells' exploration experience; the cell-specific function reflects the cell's individual behavior, and fine-tunes the common function to make the overall function f_i customized for Cell i. This function decomposition has two merits: 1) the common function is learned using all cells' data and thus it can converge quickly; 2) the cell-specific function acknowledges the differences among difference cells. We present details of the decomposition method in the following.

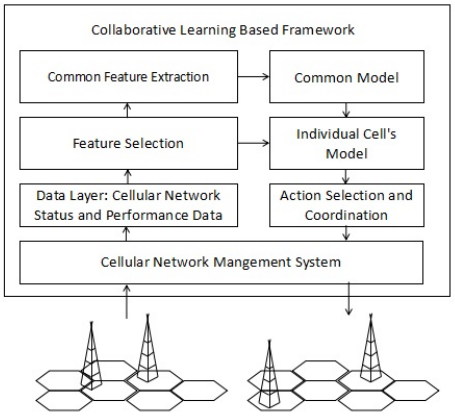


Fig. 2. A Collaborative Learning Based Framework

A. State Decomposition and Learning Latent State Space

First, we assume the state space $\mathcal{S}$ can be decomposed into two sub-spaces $\tilde{\mathcal{S}}$ and $\hat{\mathcal{S}}$ and $\mathcal{S} = \tilde{\mathcal{S}} \times \hat{\mathcal{S}}$. The cell state $\mathbf{s}_t^i$ thus can also be written as $\mathbf{s}_t^i = (\tilde{\mathbf{s}}_t^i, \hat{\mathbf{s}}_t^i)$, where $\tilde{\mathbf{s}}_t^i \in \tilde{\mathcal{S}}$ and $\hat{\mathbf{s}}_t^i \in \hat{\mathcal{S}}$.

The idea of state space decomposition is related to *common feature learning* in the literature of multi-task learning [37]. The sub-space $\tilde{\mathcal{S}}$ here represents the space of common features shared by different cells, where the common feature captures some latent structures in the cell state. The sub-space $\hat{\mathcal{S}}$ then can be seen as the space of features that are heterogeneous across cells.

More specifically, in this work, we consider the case where $\tilde{\mathbf{s}}_t^i$ is a linear transformation of the original state, i.e., $\tilde{\mathbf{s}}_t^i = W^T\mathbf{s}_t^i$. We wish to learn a transformation W such that the transformed states have *maximum cross covariance* with the performance, i.e.,

$$W^* = \arg\max_W \sum_i \sum_t \|cov(W^T\mathbf{s}_t^i, y_t)\|_F^2, \tag{7}$$

where $\|\cdot\|_F$ denotes the Frobenium norm. The interpretation of Eq. 7 is that we want to find such a common latent feature space for all cells that maximizes the correlation between the projected state and the target variable. Finding W^* according to Eq. 7 is essentially the same as using the Partial Least Squares (PLS) decomposition method [38]. Therefore, we could use the PLS method to find the transformation W.

B. Reward Decomposition

Besides the state decomposition, we further decompose the performance/reward into two parts. We divide the function f_i into a common function h and a cell-specific function g_i, i.e.,

$$y_t^i = h(\tilde{\mathbf{s}}_t^i, \mathbf{a}_t^i) + g_i(\mathbf{s}_t^i, \mathbf{a}_t^i) + \epsilon_t^i \tag{8}$$

where ϵ_t^i is a random noise with zero mean and $\tilde{\mathbf{s}}_t^i$ represents the extracted latent states. Note that in the above representation, the performance/reward contributed by the common function depends on the configured parameter $\mathbf{a}_t^i$ and the latent state $\tilde{\mathbf{s}}_t^i$; while we assume the part contributed by the cell-specific function depends on both the common feature $\tilde{\mathbf{s}}_t^i$ and the cell-specific feature $\hat{\mathbf{s}}_t^i$, and the configured parameter $\mathbf{a}_t^i$, and thus is a function of the original state $\mathbf{s}_t^i$ and parameter configuration $\mathbf{a}_t^i$.

C. Learning Common and Cell-specific Models

Now we illustrate how the common and cell-specific models could be learned. We use D_{t-1}^{all} to denote the dataset of the available data from all the cells before time t, i.e., $D_{t-1}^{all} = \left\{(\mathbf{s}_{t'}^i, \mathbf{a}_{t'}^i, y_{t'}^i) : i, t' \in \mathbb{N}, 1 \leq i \leq N, t' < t\right\}$.

1) Learn transformation W: at time t, we first use dataset D_{t-1}^{all} to learn W using the PLS method.
2) Learn common function h: we then fit a regression model h using dataset D_{t-1}^{all}, i.e.,

$$h_{t-1}^* = \arg\min_h \sum_{i=1}^{N} \sum_{t'=1}^{t-1} \|y_{t'}^i - h(W^T \mathbf{s}_{t'}^i, \mathbf{a}_{t'}^i)\|_2^2 + \lambda C_h \tag{9}$$

where C_h represents the complexity of model h and λC_h is used as a regularization term to avoid over-fitting.
3) Learn cell-specific function g_i: after we estimate h^*, we can obtain the regression residual of each cell for samples collected up to time t, i.e.,

$$\tilde{y}_{t'}^i = y_{t'}^i - h_{t-1}^*(W^T \mathbf{s}_{t'}^i, \mathbf{a}_{t'}^i) \tag{10}$$

We use D_{t-1}^i to denote the set of residual data of Cell i before time t, i.e., $D_{t-1}^i = \left\{(\mathbf{s}_{t'}^i, \mathbf{a}_{t'}^i, \tilde{y}_{t'}^i) : t' \in \mathbb{N}, t' < t\right\}$.
The cell-specific function g_i is then estimated using dataset D_{t-1}^i, i.e.,

$$g_{i,t-1}^* = \arg\min_{g_i} \sum_{t'=1}^{t-1} \|\tilde{y}_{t'}^i - g_i\left(\mathbf{s}_{t'}^i, \mathbf{a}_{t'}^i\right)\|_2^2 + \lambda C_{g_i} \tag{11}$$

where λC_{g_i} is a regularization term. That is, we find g_i by minimizing the regression error of residual of Cell i.

D. Action Selection

At each round t, as discussed in the previous section, we learn the two models based on the dataset D_{t-1}^{all} collected from all cells. Assume the performance is to be maximized, the policy to take actions once we receive the state information at time t is given as follows:

$$\begin{aligned} &\mathbf{a}_t^{i,*} = \arg\max_{\mathbf{a}^i} h_{t-1}^*(\tilde{\mathbf{s}}_t^i, \mathbf{a}^i) + g_{i,t-1}^*(\mathbf{s}_t^i, \mathbf{a}^i), \text{ w.p. } 1-\epsilon, \\ &\mathbf{a}_t^{i,*} \sim U(|\mathcal{A}|) \text{ w.p. } \epsilon, \end{aligned} \tag{12}$$

where we denote by $U(|\mathcal{A}|)$ a uniform distribution over the action domain.

More specifically, the value of ϵ varies over time t: at start, more exploration is needed and a larger value of ϵ is used; as time goes by, the value of ϵ is decreased to have more exploitation. When $t = 1$, ϵ is set to 1.

It should be noted that, although the proof in Sec. V uses the Gaussian Process Regression model, this does not limit our choice of other regression models for h and g_i. In the next section, we will show experiment results where h and g_i are learned using different regression models.

VII. Experiment Results

In this section, we present the experiment results of our framework. We first introduce the datasets we used in our experiments. Then we discuss the regression results, where we compare the prediction accuracy of regression models learned by using single cell's data, all cells' data directly, and by using our collaborative modeling method. After that, we present the convergence results of the contextual multi-armed bandit experiments. More importantly, we present results from a field test conducted in a real cellular network.

A. Datasets

We use the following two datasets in our experiments. Both datasets are collected from base stations of a metropolitan cellular network. Each dataset contains data from a few hundreds cells. For each cell, a data sample is collected each hour. Each data sample contains a number of columns, including the cell ID, sample time, configuration of different parameters, cell state measurements and performance measurement of the cell. Details of each dataset is provided in the following.

1) Single-Parameter Dataset: The dataset contains data from 297 cells over 17 days. During the collection period, the A2-threshold-RSRP parameter is adjusted for each cell once per day, and different cells may be configured different parameter values on the same day. Each sample contains a number of cell state measurements at the sample time instant, e.g., the number of total users within the cell, the number of active users, the average channel quality indicator (CQI) of the cell, etc. The performance metric for each cell in this dataset is defined as the ratio of users with experienced throughput less than 5Mbps (less-than-5M-ratio), which is a measurement of the ratio of low-throughput users (i.e., edge users). Note that the target here is to minimize the performance metric, since we wish to minimize the amount of low-throughput users in the network.

2) Multiple-Parameter Dataset: In the second dataset, data is collected from 185 cells over 14 days. During the collection period, two parameters are adjusted simultaneously for each cell once per day, and the configured parameter values for different cells might be different on the same day. The two parameters adjusted are related to uplink power control: one is the uplink path loss compensation coefficient and the other is the target uplink received power at the base station. Similar to the single-parameter dataset, cell state measurements are collected for each cell at each hour, and the performance metric for each cell is the ratio of users with experienced throughput less than 5Mbps (less-than-5M-ratio).

B. Prediction Accuracy of Regression Models

We first present the prediction accuracy of different regression models. For each dataset, data is divided into training and test sets. Regression models are learned using the training data and tested on the test set. The prediction accuracy is measured by the coefficient of determination, i.e., the R^2 score, on the test set. More specifically, for Cell i, assume the true performance of the test samples are $y_i = [y_{i,1}, y_{i,2}, \ldots, y_{i,n_i}]$, and the predicted performance are $\hat{y}_i = [\hat{y}_{i,1}, \hat{y}_{i,2}, \ldots, \hat{y}_{i,n_i}]$, where n_i is the number of test samples for Cell i. The R^2 score for Cell i is computed as

$$R_i^2 = 1 - \sum_j (y_{i,j} - \hat{y}_{i,j})^2 / \sum_j (y_{i,j} - \bar{y}_i)^2 \qquad (13)$$

where $\bar{y}_i = \frac{1}{n_i} \sum_j y_{i,j}$.

Intuitively, R_i^2 reflects how much variance in the test data could be correctly predicted by the trained model. If $R_i^2 = 1$, the model can predict Cell i's real performance perfectly; if $R_i^2 = 0$, it means the model is just a naive predictor which always predicts the mean performance; if the score is negative, the model is even worse than the naive predictor. We use the cell average R^2 score as a measure of the prediction accuracy of the regression models, i.e., $\bar{R}^2 = \frac{1}{N} \sum_i R_i^2$ where N is the number of cells in the dataset.

Single-Parameter Dataset Results We present results on the single-parameter dataset in the following. The train/test data split is conducted for each cell, and the split ratio is 1:1, which means 50% of the data in each cell is used as the training data, and the rest is used as test data. Cell state data are first pre-processed. The top 25 state features that are most related to the cell performance are selected. The 25-dim state vectors are normalized and regarded as the cell state vector $\mathbf{s}_t^i$. Prediction scores with the 25-dim features are shown in Table I. We present two cases: the *No Shuffle* and *Shuffle* case. For *No Shuffle*, each cell's data is sorted according to the sample time, and data collected earlier are used to train the model to make predictions for the future; for *Shuffle*, each cell's data is randomly shuffled and then splitted into the training and test data.

TABLE I
$\bar{R}^2$ SCORE FOR DIFFERENT MODELS

Model	No Shuffle		Shuffle	
	Train	Test	Train	Test
Single Cell SVR	0.580	-0.147	0.548	0.437
Single Cell MLP-Bagging	0.174	-0.627	0.105	0.013
Common SVR	0.304	0.117	0.456	0.350
Common MLP-Bagging	0.595	0.087	0.611	0.509
SVR + SVR	0.691	0.144	0.690	0.554
MLP-Bagging + SVR	0.758	0.189	0.747	0.627
SVR + MLP-Bagging	0.915	0.090	0.907	0.647
MLP-Bagging + MLP-Bagging	0.923	0.069	0.917	0.638

We find the Support Vector Regression (SVR) and Multi-layer Perceptron (MLP) models perform well on the dataset. In addition, bagging method is used to train the MLP model to reduce the model variance.

For the rows of Table I, "Single Cell XX" means models trained with single cell's data; "Common XX" means a common model is trained for all the cells with all the training data; for the rest rows, the model names are given as "common + cell-specific" model, e.g., "MLP-Bagging + SVR" means MLP-Bagging model is used to train the common model, and SVR is used to train the cell-specific model.

For the common and cell-specific modeling results in Table I, the transformation matrix W is set to I, i.e., the latent state has the same number of dimensions as the original state. We also tried different number of dimensions for the latent state. For example, for "SVR+SVR" model, in the *No Shuffle* case, the test scores are 0.144, 0.177 and 0.174 with 25, 23 and 21 latent dimensions, respectively, which shows that extracting the commons features improves the prediction accuracy. From the results, it is obvious that using the common and cell-specific models together performs best among all the methods. Results of the *Shuffle* case are generally better than the *No Shuffle* case, since the data distributions of the cells may change over time during the data collection period. Note that in practice, we can only use past data to learn models and make predictions, hence results for the *No Shuffle* case have more practical implications.

C. Multi-armed Bandit Experiments On Simulator

In this part, we present the convergence results of the multi-armed bandit experiment tested on simulator. We first describe the simulator construction, and then present the results.

1) Simulator Construction: To verify our bandit algorithm, we need a simulator that could return the network performance for any queried state and parameter value. However, the current datasets only contain the measured performance under a limited number of cell states and parameter values. Therefore, we need to estimate the network performance for a given state and parameter value based on the historical data. The method we use to do this estimation is to learn a regression model using the datasets. More specifically, we use the "common and cell-specific" modeling approach discussed earlier to learn the regression models.

2) Simulation Procedures: The detailed simulation procedures are illustrated in Algorithm 1.

In the simulation, we follow the real parameter adjustment practice of network operators, where parameter values could be changed only once each day for each cell. Therefore, one iteration corresponds to one day. For Cell i, the real measurements of cell states at Day t in the dataset are used as Cell i's states in Iteration t. Note that since we have one data sample each hour, in this simulation setting, we can obtain multiple data samples for each cell after each iteration. We use $\bar{y}_t^i$ to denote the average performance of Cell i in Iteration t, and $\bar{y}_t^{i,o}$ to denote the optimal average performance of Cell i in Iteration t. This optimum could be found by enumerating all the parameter configurations in $\mathcal{A}$ and choosing the one for which simulator returns the best performance. The regret of Cell i in Iteration t is then $r_t^i = \bar{y}_t^i - \bar{y}_t^{i,o}$ (since the performance metric less-than-5M-ratio should be minimized). We compute the mean regret of all the cells in each iteration.

Algorithm 1 Simulation Procedures

Initialize: dataset $D^{all} = \{\}$
for $t = 1$ to T **do**
  if $t = 1$ **then**
    For all i, generate a random $\mathbf{a}_1^i$ for Cell i
  else
    Use dataset D^{all} to learn h and g_i for all i
    Observe cell states for Day t
    Decide $\mathbf{a}_t^i$ of Day t for each Cell i based on cell states of Day t and the models h and g_i
  end if
  Get performance of all cells for Day t from simulator
  Add simulation data of Day t to D^{all}
  $t \leftarrow t + 1$
end for

Fig. 3. Cell Mean Regret For Single-Parameter Experiment

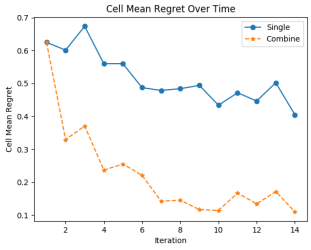


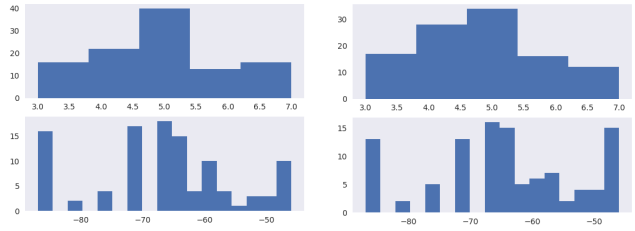
Fig. 4. Cell Mean Regret For Multiple-Parameter Experiment

3) Single-Parameter Experiment: The following are results on the single-parameter dataset. The dataset has data over 17 days, so the total number of iterations T is set to 17. The search range of the A2-threshold-RSRP parameter is $[-112, -80]$ dBm. The results are shown in Fig. 3. The line "Single" gives the results where each cell only uses its own exploration data to make decisions in each iteration; the line "Combine" represents the results where the collaborative learning framework is used, and exploration data of all the cells are used in each iteration to learn the common and cell-specific models and make decisions for all cells. It could be seen from the results that the mean regret drops much more quickly by using the collaborative learning method.

4) Multiple-Parameter Experiment: Results for the multiple-parameter experiment is shown in Fig. 4. The dataset has data over 14 days, so the total number of iterations T is set to 14. The first parameter is a coefficient and has range from 3 to 7; the second one ranges from -85 dBm to -46 dBm with stepsize of 5 dBm. We also compared distributions of the recommended parameter values with the optimum of the simulator in the last iteration. The results are given in Fig. 5 which shows that distributions of the recommended parameter values by the learned models are close to the ground truth.

D. Live Field Test Results

Next, we present the testing results of our algorithm on a real metropolitan network. The networks contains 1755 cells



(a) (b)

Fig. 5. (a) Distribution of optimal parameter values from simulator; (b) Distribution of recommended parameter values by our method.

TABLE II
PARAMETERS OPTIMIZED IN REAL NETWORK TEST

Parameter	Meaning
A	An upper bound on the uplink reception power; used for uplink power control
B	Target initial downlink BLER; used for deciding downlink modulation and coding scheme (MCS)
C	Controls how MCS is adjusted to utilize unoccupied resource blocks (RBs)
D	Controls the initial MCS of users
E	Controls the MCS adjustment speed

in total. Five parameters related to power control and user scheduling are adjusted simultaneously. Since these parameters are related to proprietary products, we omit the parameter names and only list their physical meanings in Table II. Each parameter can take around 10 possible values.

The test lasts for 14 days, starting from April 24 to May 7, 2018. Before testing, the configurations of the cells are set to default configurations. During testing, for each cell, the parameter configuration is adjusted in the morning of each day, and data of all the cells are collected at the end of the day. The performance target to be optimized is the less-than-5Mbps-ratio mentioned previously.

Note that in the real testing, we have to decide the parameter configuration for each cell at the beginning of each day before we observe the cell states. Therefore, we use the cell states of the previous week as an approximation to decide the parameter configuration.

Fig. 6 shows the cell average less-than-5Mbps-ratio of the test region over the testing period. It could be seen that the target less-than-5Mbps-ratio drops for around 20% over the optimization period, which shows a significant performance improvement.

VIII. CONCLUSION

In this paper, we propose a collaborative-learning-based approach to optimize parameter configurations in cellular networks. The proposed approach leverages the contextual bandit algorithm and facilitates it with transfer learning to improve the policy learning and decision making efficiency. Experimental results based on simulations and real network tests show that our framework is effective. Our practice also demonstrates that machine learning or data-driven approaches have good potential towards self-evolving wireless networks.

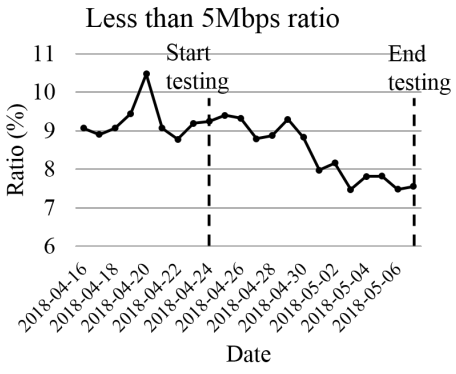


Fig. 6. Results of Real Network Testing

REFERENCES

[1] E. Dahlman, S. Parkvall, and J. Skold, *4G: LTE/LTE-Advanced for Mobile Broadband*. Academic Press, 2013.

[2] P. Bhat, S. Nagata, L. Campoy, I. Berberana, T. Derham, G. Liu, X. Shen, P. Zong, and J. Yang, "LTE-Advanced: an operator perspective," *IEEE Communications Magazine*, vol. 50, no. 2, 2012.

[3] Z. Guohua, P. Legg, and G. Hui, "A network controlled handover mechanism and its optimization in LTE heterogeneous networks," in *IEEE Wireless Communications and Networking Conference*. IEEE, 2013, pp. 1915–1919.

[4] A. S. Priyadharshini and P. Bhuvaneswari, "A study on handover parameter optimization in LTE-A networks," in *International Conference on Microelectronics, Computing and Communications (MicroCom)*. IEEE, 2016, pp. 1–5.

[5] 3GPP TS36.331, *Evolved Universal Terrestrial Radio Access (E-UTRA); Radio Resource Control (RRC); Protocol specification*, 2016.

[6] X. Cheng, L. Fang, X. Hong, and L. Yang, "Exploiting mobile big data: sources, features, and applications," *IEEE Network*, vol. 31, no. 1, pp. 72–79, January 2017.

[7] S. Bi, R. Zhang, Z. Ding, and S. Cui, "Big data aware wireless communication: challenges and opportunities," *Big Data over Networks*, pp. 180–216, 2016.

[8] L. Xiao, Y. Li, C. Dai, H. Dai, and H. V. Poor, "Reinforcement learning-based NOMA power allocation in the presence of smart jamming," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 4, pp. 3377–3389, April 2018.

[9] X. Chen, J. Wu, Y. Cai, H. Zhang, and T. Chen, "Energy-efficiency oriented traffic offloading in wireless networks: a brief survey and a learning approach for heterogeneous cellular networks," *IEEE Journal on Selected Areas in Communications*, vol. 33, no. 4, pp. 627–640, April 2015.

[10] X. Guo, G. Trimponias, X. Wang, Z. Chen, Y. Geng, and X. Liu, "Cellular network configuration via online learning and joint optimization," in *IEEE International Conference on Big Data*, December 2017, pp. 1295–1300.

[11] ——, "Learning-based joint configuration for cellular networks," *IEEE Internet of Things Journal*, 2018.

[12] S. Bubeck and N. Cesa-Bianchi, "Regret analysis of stochastic and nonstochastic multi-armed bandit problems," *Machine Learning*, vol. 5, no. 1, pp. 1–122, 2012.

[13] P. Auer, N. Cesa-Bianchi, and P. Fischer, "Finite-time analysis of the multiarmed bandit problem," *Machine Learning*, vol. 47, no. 2-3, pp. 235–256, 2002.

[14] O. Chapelle and L. Li, "An empirical evaluation of Thompson Sampling," in *Advances in Neural Information Processing Systems*, 2011, pp. 2249–2257.

[15] W. Chu, L. Li, L. Reyzin, and R. Schapire, "Contextual bandits with linear payoff functions," in *AISTATS*, 2011, pp. 208–214.

[16] L. Li, W. Chu, J. Langford, and R. E. Schapire, "A contextual-bandit approach to personalized news article recommendation," in *ACM International Conference on World Wide Web*, 2010, pp. 661–670.

[17] W. Ouyang, A. Eryilmaz, and N. B. Shroff, "Asymptotically optimal downlink scheduling over Markovian fading channels," in *Proceedings of IEEE INFOCOM*, March 2012, pp. 1224–1232.

[18] X. Guo, R. Singh, P. R. Kumar, and Z. Niu, "Optimal energy-efficient regular delivery of packets in cyber-physical systems," in *IEEE ICC*, 2015.

[19] R. Singh, X. Guo, and P. R. Kumar, "Index policies for optimal mean-variance trade-off of inter-delivery times in real-time sensor networks," in *Proceedings of IEEE INFOCOM*, 2015.

[20] Y. Sun, X. Guo, S. Zhou, Z. Jiang, X. Liu, and Z. Niu, "Learning-based task offloading for vehicular cloud computing systems," in *IEEE ICC*, 2018.

[21] Y. Sun, J. Song, S. Zhou, X. Guo, and Z. Niu, "Task replication for vehicular edge computing: a combinatorial multi-armed bandit based approach," in *IEEE GLOBECOM*, 2018.

[22] S. J. Pan, Q. Yang *et al.*, "A survey on transfer learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 10, pp. 1345–1359, 2010.

[23] K. Weiss, T. M. Khoshgoftaar, and D. Wang, "A survey of transfer learning," *Journal of Big Data*, vol. 3, no. 1, p. 9, 2016.

[24] C. Wang and S. Mahadevan, "Heterogeneous domain adaptation using manifold alignment," in *International Joint Conference on Artificial Intelligence*, vol. 22, no. 1, 2011, p. 1541.

[25] L. Duan, D. Xu, and I. Tsang, "Learning with augmented features for heterogeneous domain adaptation," *arXiv preprint arXiv:1206.4660*, 2012.

[26] R. Chattopadhyay, Q. Sun, W. Fan, I. Davidson, S. Panchanathan, and J. Ye, "Multisource domain adaptation and its application to early detection of fatigue," *ACM Transactions on Knowledge Discovery from Data*, vol. 6, no. 4, p. 18, 2012.

[27] S. J. Pan, J. T. Kwok, and Q. Yang, "Transfer learning via dimensionality reduction," in *AAAI*, vol. 8, 2008, pp. 677–682.

[28] S. J. Pan, I. W. Tsang, J. T. Kwok, and Q. Yang, "Domain adaptation via transfer component analysis," *IEEE Transactions on Neural Networks*, vol. 22, no. 2, pp. 199–210, 2011.

[29] N. Srinivas, A. Krause, S. M. Kakade, and M. Seeger, "Gaussian process optimization in the bandit setting: no regret and experimental design," *arXiv preprint arXiv:0912.3995*, 2009.

[30] M. Valko, N. Korda, R. Munos, I. Flaounas, and N. Cristianini, "Finite-time analysis of kernelised contextual bandits," *arXiv preprint arXiv:1309.6869*, 2013.

[31] S. Agrawal and N. Goyal, "Analysis of Thompson Sampling for the multi-armed bandit problem," in *Conference on Learning Theory*, 2012, pp. 39–1.

[32] H. Wu and X. Liu, "Double Thompson Sampling for dueling bandits," in *Advances in Neural Information Processing Systems*, 2016, pp. 649–657.

[33] C. E. Rasmussen, "Gaussian processes in machine learning," in *Advanced Lectures on Machine Learning*. Springer, 2004, pp. 63–71.

[34] B. Schölkopf, A. J. Smola *et al.*, *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT Press, 2002.

[35] R. M. Johnstone, C. R. Johnson, R. R. Bitmead, and B. D. Anderson, "Exponential convergence of recursive least squares with exponential forgetting factor," in *21st IEEE Conference on Decision and Control*. IEEE, 1982, pp. 994–997.

[36] L. Le Gratiet and J. Garnier, "Asymptotic analysis of the learning curve for Gaussian process regression," *Machine Learning*, vol. 98, no. 3, pp. 407–433, 2015.

[37] A. Argyriou, T. Evgeniou, and M. Pontil, "Multi-task feature learning," in *Advances in Neural Information Processing Systems*, 2007, pp. 41–48.

[38] J. A. Wegelin *et al.*, "A survey of Partial Least Squares (PLS) methods, with emphasis on the two-block case," 2000.