Cellular Network Configuration via Online Learning and Joint Optimization

Xueying Guo*, George Trimponias†, Xiaoxiao Wang*, Zhitang Chen†, Yanhui Geng† and Xin Liu*
* *Computer Science Department, University of California, Davis, USA*
Email: guoxueying@outlook.com, {xxwa, xinliu}@ucdavis.edu
†*Noah's Ark Lab, Huawei Technologies*
Email:{g.trimponias, chenzhitang2, geng.yanhui}@huawei.com

***Abstract*—Cellular network configuration is critical for network performance. Current practice is labor-intensive, error-prone, and far from optimal. To automate efficient cellular network configuration, in this work, we propose an online-learning-based joint-optimization approach that addresses a few specific challenges: limited data availability, convoluted sample data, highly complex optimization due to interactions among neighboring cells, and the need to adapt to network dynamics. In our approach, to learn an appropriate utility function for a cell, we develop a neural-network-based model that addresses the convoluted sample data issue and achieves good accuracy based on data aggregation. Based on the utility function learned, we formulate a global network configuration optimization problem. To solve this high-dimensional non-concave maximization problem, we design a Gibbs-sampling-based algorithm that converges to an optimal solution when a technical parameter is small enough. Furthermore, we design an online scheme that updates the learned utility function and solves the corresponding maximization problem efficiently to adapt to network dynamics. To illustrate the idea, we use the case study of pilot power configuration. Numerical results illustrate the effectiveness of the proposed approach.**

***Keywords*-cellular network configuration; learning-based configuration; reinforcement learning; joint optimization;**

I. Introduction

In this work, we study cellular network configuration. As illustrated in Fig. 1, a cellular network consists of a number of base stations (BSs), each covering a certain geographic area, called a cell. In each cell, the BS has a large number of parameters to configure, including those of spectrum band, power configuration, antenna setting, user handoff, etc. The configuration of these BS parameters has a significant impact on the overall performance, and thus network configuration is a critical issue. Current practice is mostly based on field experience and manual adjustment. The process is labor-intensive, error-prone, and far from optimal. In this work, we propose a learning-based joint-optimization approach to automate and optimize cellular network configuration.

The goal of network configuration is to optimize network utility that measures service quality, such as network throughput and user delay. Network utility is typically defined as the sum of the utility of all cells. Because of the geographic contiguity of cells, the utility of one cell is not only affected by the configuration of its own BS, but also by those of its neighboring BSs. Thus, to optimize the network utility, the network should jointly configure all BSs.

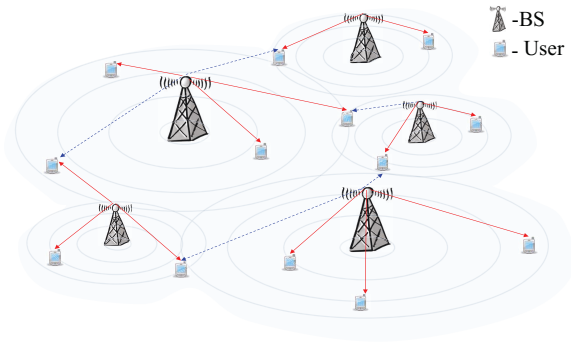

Figure 1. An illustration of cellular network configuration. A red solid line means that a mobile user is associated with the BS (i.e., the BS serves the user), and a blue dash line means that the mobile user can receive the BS's signal, but is not served by it.

The extensive literature on traditional network utility maximization (NUM) has focused on abstract closed-form utility functions, which often exhibit nice mathematical properties. However, in network configuration, the impact of a control parameter on network utility can be highly complex and thus the utility function is difficult to derive from analysis. Therefore, a natural step is to learn an appropriate utility function based on available network data, and then to optimize over the learned function. This approach faces a few specific challenges:

- Limited data availability: Because current network operation does not often change network configuration parameters unless performance issues occur, each cell contains only a limited amount of data.
- Convoluted sample data: The sample data collected in networks is in the format of network state, BS configuration, and the corresponding utility value. The caveat is that the network state depends not only on hidden environmental states that we do not directly observe, but also on the current configuration. Thus, we need to extract appropriate features to learn the utility model.
- Joint optimization among cells: The utility of a cell depends on the configuration of not only its BS, but also its neighboring BSs. Therefore, to maximize the network utility, we consider all cell configurations

jointly. This is difficult because of the complexity of the utility function and the high dimensionality of the control variables.
- Time-varying network dynamics: Because of inherent network dynamics (e.g., due to traffic variation and user mobility), the utility function is time-varying, as discussed in Sec. III-B. It is essential to design an online algorithm that adapts promptly to network dynamics.

To address these challenges, we propose an approach based on online learning and joint optimization for cellular network configuration. Our contributions are multi-fold:

- To learn the appropriate utility function, we develop a neural-network-based model that addresses the convoluted sample data issue and achieves good accuracy. The learned utility function is used to formulate a global network configuration optimization problem.
- To solve this high-dimensional non-concave maximization problem, we design a Gibbs-sampling-based algorithm that converges to the optimal solution when a temperature parameter is small enough. However, when the temperature parameter is small, the convergence time of the algorithm increases dramatically. To address this challenge, we further design an online algorithm that converges to a local optimal promptly.
- To illustrate the idea, we use the case study of pilot power configuration. Numerical results show the effectiveness of the proposed approach. The proposed approach can be applied to similar cellular network configuration problems.

The rest of the paper is organized as follows. The related work is in Sec. II. We present the system model and problem formulation in Sec. III. The learning-based utility prediction is conducted in Sec. IV. Based on the model learned, an online-learning-based joint-optimization method for cellular network configuration is proposed in Sec. V. We demonstrate the numerical results in Sec. VI, and conclude in Sec. VII.

II. Related Work

Most existing studies on pilot power configuration assume known models based on communication theories [1]–[5]. For instance, in [1], the authors study the problem of minimizing pilot power consumption while maintaining service coverage. The problem is formulated as integer programming and solved via a Dantzig-Wolfe decomposition method. The main limitation is that the modeling requires a significant amount of real-time information, such as user locations and wireless channel conditions, at any time. Further, even with a known analytical model, most existing studies such as [2], [3] focus on the design of heuristic policies.

We note that our problem falls into the general scope of reinforcement learning (RL). However, because of the challenges discussed earlier, directly applying off-the-shelf RL algorithms may not perform well. We next discuss related work on RL-based network configuration. In [6], the authors propose a tailored form of RL to configure the antenna parameters in a single femto-cell to minimize the transmission power. However, in a large-scale network, general RL algorithms can suffer from large state spaces and action sets. To address this challenge, a line of work exploits the paradigm of collaborative multiagent reinforcement learning [7], [8]. For example, in [9], the authors propose a distributed RL algorithm to configure power allocation of all cells with the objective of improving network performance. However, distributed RL algorithms usually rely on heuristics and their design is non-trivial. Along this line, we have designed a distributed Q-learning algorithm as a comparison baseline, as shown in Sec. VI.

III. System Model and Problem Formulation

In this work, we propose a cellular network configuration approach via online learning and joint optimization. Specifically, we demonstrate the idea in a pilot power configuration problem, as illustrated in Fig. 1. Pilot power of a base station allows mobile users to detect the base station, calibrate signals, and estimate channel quality between the BS and the mobile user. A mobile user can typically receive pilot powers from multiple BSs, and based on which it decides which BS to associate with (i.e., the red solid line).

A. Problem Formulation

We formulate the problem next. We use bold font to represent vectors. Consider a cellular network of N BSs. Each BS provides service for a corresponding cell. The time is discretized. For each time slot t, the network state vector is denoted by $\mathbf{S}^{(t)} = (S_1^{(t)}, S_2^{(t)}, \cdots, S_N^{(t)})$, where $S_n^{(t)}$ is the state for BS n in slot t. Examples of network state include the number of users, the number of active users, and uplink/downlink load, etc. The network control is denoted by $\mathbf{A}^{(t)} = (A_1^{(t)}, A_2^{(t)}, \cdots, A_N^{(t)})$ where $A_n^{(t)}$ represents the control variables for BS n in slot t, and its value is chosen from the BS action set $\mathcal{A}$. Pilot power is the control variable we consider in this work, which takes discrete values from 30dBm to 36dBm with a granularity of 0.5dBm.

Let $R_n^{(t)}$ denote the utility of cell n in slot t and define $r_n(\cdot)$ as the cell utility function of cell n that satisfies,

$$R_n^{(t)} = r_n(\mathbf{S}^{(t)}, \mathbf{A}^{(t)}). \tag{1}$$

That is, the utility of each cell is determined by the network states and control variables. In the pilot power configuration problem, the utility is the throughput, and the object is to maximize the network throughput by choosing appropriate control variables for all BSs at each time slot. That is,

$$\max_{\mathbf{A}^{(t)} \in \mathcal{A}^N} \sum_{t=1}^{T} \sum_{n=1}^{N} r_n(\mathbf{S}^{(t)}, \mathbf{A}^{(t)}). \tag{2}$$

In addition, note that in Eq. (1), the utility value depends on all states and control variables, and is thus very complex. To tackle this, we assume that the performance of a BS is mainly affected by the network states and control variables of itself and its neighbors. Thus, we make the following approximation:

$$r_n(\mathbf{S}^{(t)}, \mathbf{A}^{(t)}) \approx r_n(\mathbf{S}^{(t)}_{\mathcal{N}(n)}, \mathbf{A}^{(t)}_{\mathcal{N}(n)}), \tag{3}$$

where $\mathcal{N}(n) = \{n\} \cup \{i|$ BS i is BS n's neighbor$\}$ includes BS n and its neighbors; $\mathbf{S}^{(t)}_{\mathcal{N}(n)}$ is a vector which includes all $S^{(t)}_i$ such that $i \in \mathcal{N}(n)$; and similar for $\mathbf{A}^{(t)}_{\mathcal{N}(n)}$.

For example, in the pilot power configuration problem, we choose the neighbors of BS n as the cells that have the most impact on cell n's utility. Note that the neighborhood relationship may not be symmetric. That is, $i \in \mathcal{N}(n)$ does not necessarily imply $n \in \mathcal{N}(i)$.

To further simplify the problem, we assume that the network state $\mathbf{S}^{(t)}$ does not depend on historical actions $\mathbf{A}^{(t-1)}, \cdots, \mathbf{A}^{(1)}$. To satisfy the assumption, we need to choose network states appropriately, that is, network states should only depend on latent environment features such as user density, user location, traffic type, and mobility pattern, which do not depend on actions. Because of this assumption, the problem defined in (2) can be simplified: at each time slot t, we can choose the action that maximizes the current network utility $\sum_n r_n(\mathbf{S}^{(t)}, \mathbf{A}^{(t)})$. Thus, (2) becomes:

$$\max_{\mathbf{A}^{(t)} \in \mathcal{A}^N} \sum_{n=1}^{N} r_n(\mathbf{S}^{(t)}_{\mathcal{N}(n)}, \mathbf{A}^{(t)}_{\mathcal{N}(n)}), \text{ given network state } \mathbf{S}^{(t)}. \tag{4}$$

The network operates as follows: At the beginning of each slot t, we observe network state $\mathbf{S}^{(t)}$, based on which, we decide in a centralized manner the configuration of all cells $\mathbf{A}^{(t)}$ to maximize the sum of the utilities of all cells at time t. Note that, in contrast, our implemented baseline algorithm, a distributed Q-learning algorithm, still uses the aggregated throughput as the reward as in Eq. (2).

B. Network Dynamics

We note that networks exhibit time-varying characteristics, which implies that we need online algorithms to capture such dynamics. Network utility is affected by many network factors, such as traffic type and volume, arrival and departure pattern, and user mobility. However, some factors are "hidden" because a network typically does not and cannot collect all information. Therefore, when such "hidden" factors change in the network, the expression of cell utility functions, i.e., $r_n(\cdot)$, can change over time. To capture such dynamics, we need an online learning algorithm to train the model as well as a fast converging algorithm to optimize the configuration.

IV. Learning Utility Function

To learn a good utility function, we need to address two specific challenges: limited data availability and convoluted sample data, discussed as follows. We have employed a neural network regression model to predict cell utility function.

A. Data Aggregation

To fully utilize the limited data, we notice that a cellular network typically consists of cells with similar structures. Thus, we assume that the cell utility functions $r_n(\cdot)$ of different cells have the same expression, denoted by $r(\cdot)$. Thus, the problem defined in (4) can be approximated by the following, at each time slot t,

$$\max_{\mathbf{A}^{(t)} \in \mathcal{A}^N} \sum_{n=1}^{N} r(\mathbf{S}^{(t)}_{\mathcal{N}(n)}, \mathbf{A}^{(t)}_{\mathcal{N}(n)}), \text{ given network state } \mathbf{S}^{(t)}. \tag{5}$$

With this unified cell utility function, we can aggregate data from all cells to learn the utility function. This approach alleviates the data scarcity issue, especially when the algorithm needs to adapt to network dynamics.

B. Feature Selection with Convoluted Sample Data

The sample data is collected from an industrial-grade cellular network simulator that is described in Sec. VI, based on a random policy for pilot power configuration. Sample data is illustrated in Table I, where 'Cell' is the cell ID; 'Pilot' is the current pilot power of the BS in this cell; 'Utility' is the throughput of the corresponding cell with the unit of Kbit; 'Users' is the number of users in the cell, and 'Cluster' includes the cell IDs of the 5 neighboring cells that have the most impact on this cell's throughput.

Table I
Features of Sample Data.

Cell	Pilot	Load	Utility	Users	Cluster
101	30	0.5934	616.3	19	233;184;202;185;171;
102	30	0.1458	63.2	2	101;177;180;184;172;
107	33	0.8991	357.8	19	211;115;114;219;113;

An important problem in feature selection is that sample data is convoluted as shown in Table I. Specifically, the values of the load and the number of users in the sample data depend not only on latent environmental states that we do not directly observe (such as user density), but also on the current control variables, i.e. the pilot powers.

As a result, we need to carefully select state features so that they 1) extract latent environment information as much as possible, and 2) are as independent as possible from the control variables.

Considering these two factors, we have tested the prediction accuracy of several sets of features, as summarized in Table II, based on a neural network regression model. We measure prediction accuracy by the coefficient of determination R^2 as,

$$R^2 = 1 - \frac{\eta_{\text{ss}}}{\eta_{\text{var}} M_{\text{sp}}}, \tag{6}$$

Table II
COEFFICIENT OF DETERMINATION FOR DIFFERENT SELECTIONS OF FEATURES.

Features	Training data	Cross Validation
Pilot Powers, Load, Number of Users	0.729630	0.728747
Pilot Powers, Time-Average Load, Time-Average Number of Users	0.758423	0.717136
Pilot Powers, Cluster-Average Load, Cluster-Average Number of Users	0.127370	0.113907

where η_{ss} is the sum of squared prediction errors, η_{var} is the variance of the target, and M_{sp} is the total number of samples. The larger the value of R^2, the better the model can capture the observed outcomes. Note that $R^2 \leq 1$.

The first feature set includes the pilot powers of the target BS and its neighbors, the current load of the target BS, and the current number of users in the target BS, as shown in the first row of Table II. This directly uses the items from the sample data. The prediction accuracy of this feature set is about 0.73. However, as stated above, the load and the number of users depend on the current pilot powers, and thus this model is not useful in pilot power configuration. It only serves as a benchmark for prediction accuracy.

To address this convolution issue, we consider two other feature sets, as shown in the second and third rows of Table II. The second feature set includes time-average load and time-average number of users in the target cell, and the third one includes cluster-average load and cluster-average number of users, instead of the load and number of users as in the first one. Here, the cluster average means that we average the load or the number of users among the target cell and its neighboring cells. These two new feature sets alleviate the dependency of state features on the control variables by averaging over time slots or over mutually interfering cells. As shown in Table II, the prediction accuracy of the second feature set is similar to that of the first feature set while the prediction accuracy of the third one is low. Therefore, we choose the second feature set.

V. ONLINE-LEARNING-BASED JOINT OPTIMIZATION

After estimating the cell utility function $r(\cdot)$, we proceed to solve the optimization problem formulated in Eq. (4). This step is challenging because of the huge action space for network configuration. In this section, we design an online algorithm based on Gibbs sampling to solve this joint optimization problem.

A. Gibbs-Sampling-Based Network Configuration

In this subsection, we focus on a single slot t, and solve the problem in (4) for a given network state $\mathcal{S}^{(t)}$. We omit the superscript (t) in this section when no confusion is incurred.

We begin with some notations. We define $F(\cdot)$ as the network utility function:

$$F(\mathbf{a}) = \sum_{n=1}^{N} r_n(\mathbf{S}_{\mathcal{N}(n)}, \mathbf{a}_{\mathcal{N}(n)}), \forall \mathbf{a} \in \mathcal{A}^N. \tag{7}$$

Define $\hat{\mathbf{A}}$ as a random vector on the action space satisfying the probability distribution $\Pr[\hat{\mathbf{A}} = \mathbf{a}] = \pi_\nu(\mathbf{a})$ with,

$$\pi_\nu(\mathbf{a}) = \frac{\exp\left[\frac{F(\mathbf{a})}{\nu}\right]}{\sum_{\mathbf{z}\in\mathcal{A}^N} \exp\left[\frac{F(\mathbf{z})}{\nu}\right]}, \forall \mathbf{a} \in \mathcal{A}^N, \tag{8}$$

where $\nu > 0$ in (8) is called the temperature parameter.

We first introduce the following neighborhood concepts. Note that the neighborhood $\mathcal{N}$ in our problem is not necessarily a mutual concept, as discussed in Sec. III. Thus, we further define the interaction neighborhood $\mathcal{N}^{(1)}$ as,

$$\mathcal{N}^{(1)}(n) = \{i | n \in \mathcal{N}(i)\} \cup \mathcal{N}(n). \tag{9}$$

That is, $\mathcal{N}^{(1)}(n)$ includes both the BSs that affect cell n's utility, i.e. $\mathcal{N}(n)$, and the cells that are affected by BS n. We further define two-hop neighborhood $\mathcal{N}^{(2)}$ as in [10],

$$\mathcal{N}^{(2)}(n) = \cup_{i\in\mathcal{N}^{(1)}(n)} \mathcal{N}^{(1)}(i).$$

That is, $\mathcal{N}^{(2)}(n)$ is the union of the interaction neighborhoods of all interaction neighbors of BS n.

Then, by a similar argument as in [10], we can define an undirected graph based on the neighborhood concept of $\mathcal{N}^{(2)}$ so that (8) is the distribution of a Markov random field on this undirected graph of neighborhood $\mathcal{N}^{(2)}$, and the local specification of the Markov random field is,

$$\begin{aligned}&\Pr\{\hat{A}_n = \mathrm{a}_n | \hat{\mathbf{A}}_{\mathcal{N}^{(2)}(n)\setminus n} = \mathbf{a}_{\mathcal{N}^{(2)}(n)\setminus n}\}\\ =&\frac{\exp\left[\frac{1}{\nu}\sum_{i\in\mathcal{N}^{(1)}(n)} r_i(\mathbf{S}_{\mathcal{N}(i)}, \mathbf{a}_{\mathcal{N}(i)})\right]}{\sum_{\lambda\in\mathcal{A}} \exp\left[\frac{1}{\nu}\sum_{i\in\mathcal{N}^{(1)}(n)} r_i(\mathbf{S}_{\mathcal{N}(i)}, \lambda, \mathbf{a}_{\mathcal{N}(i)\setminus n})\right]},\end{aligned}$$

where $\mathcal{N}^{(2)}(n) \setminus n$ is a set including all the BSs in $\mathcal{N}^{(2)}(n)$ but excluding n.

Inspired by the Gibbs sampling algorithm, we have Module 1. Module 1 aims at solving the problem for one time slot t with a given network state $\mathbf{S}^{(t)}$ using an iterative algorithm. In this iterative algorithm, in each iteration τ, a vector $\tilde{\mathbf{A}}^{(\tau)} = (\tilde{A}_1^{(\tau)}, \tilde{A}_2^{(\tau)}, \cdots, \tilde{A}_N^{(\tau)})$ is generated, leading to a sequence of output vectors $\tilde{\mathbf{A}}^{(\tau)}$, $\tau = 1, 2, \cdots$. In Module 1, first in Line 1, based on $\mathcal{N}(\cdot)$, the undirected interfering neighborhood $\mathcal{N}^{(1)}$ is constructed. Then, in each iteration τ, $\tilde{\mathbf{A}}^{(\tau)}$ is generated by updating $\tilde{\mathbf{A}}^{(\tau-1)}$. Note that $\nu > 0$ is the temperature parameter as in (8).

Due to the property of the Gibbs distribution, it is possible to show that $\tilde{\mathbf{A}}^{(\tau)}$ in Module 1 has a steady-state distribution that chooses the optimal solution of the problem defined in (4) with a probability arbitrarily close to one, if the temperature parameter ν is sufficiently small. This ensures optimality.

Module 1 Gibbs-Sampling-Based Network Configuration

Input: ν, $\tilde{\mathbf{A}}^{(0)}$; N, $\mathcal{N}(\cdot)$, $\mathbf{S}^{(t)}$, $\mathcal{A}$ and $r_n(\cdot)$ for each n.
Output: $\tilde{\mathbf{A}}^{(\tau)}$ for each iteration τ.

1: **Init:** Calculate $\mathcal{N}^{(1)}(n)$ for each BS n by (9);
2: **for** each iteration τ **do**
3: $\tilde{\mathbf{A}}^{(\tau)} = \tilde{\mathbf{A}}^{(\tau-1)}$;
4: Pick i uniformly at random from $\{1, 2, \cdots, N\}$;
5: Update $\tilde{A}_i^{(\tau)}$ by picking its value from the action set $\mathcal{A}$ according to the following probability,

$$\forall \mu \in \mathcal{A}, \quad \Pr\{\tilde{A}_i^{(\tau)} = \mu\} = \frac{\exp\left[\frac{1}{\nu}\sum_{j\in\mathcal{N}^{(1)}(i)} r_j(\mathbf{S}^{(t)}_{\mathcal{N}(j)}, \mu, \tilde{\mathbf{A}}^{(\tau-1)}_{\mathcal{N}(j)\setminus i})\right]}{\sum_{\lambda\in\mathcal{A}}\exp\left[\frac{1}{\nu}\sum_{j\in\mathcal{N}^{(1)}(i)} r_j(\mathbf{S}^{(t)}_{\mathcal{N}(j)}, \lambda, \tilde{\mathbf{A}}^{(\tau-1)}_{\mathcal{N}(j)\setminus i})\right]}; \tag{10}$$

6: **end for**

B. Accelerate Convergence

While it has nice theoretical properties, Module 1 has two practical limitations. First, if ν is too small, computational challenges exist due to numerical overflow because the exponential factors in (10) can be extremely large. Second, when the temperature parameter ν is small enough, the convergence time for $\tilde{\mathbf{A}}^{(\tau)}$ in Module 1 can be extremely long. Thus, in this subsection, we focus on a single slot t, and design a fast converging algorithm.

The algorithmic design is presented in Module 2. It is inspired by the insight from Gibbs sampling: In Module 1, a smaller temperature parameter ν leads to a larger probability of selecting a local optimum solution of the problem in (4), while a larger parameter ν leads to a larger probability of escaping from the trap of a local optimum. Specifically, instead of randomly choosing one BS to update in each iteration as in Line 4 of Module 1, Module 2 updates the corresponding control variable for each BS in turn in Lines 4-6. In addition, after selecting a BS, Module 2 chooses the value of the control variable of the selected BS in a deterministic way in Line 5. This is equivalent to setting the temperature ν to 0 in Module 1[1]. Note that Module 2 speeds up the procedure by trading off the performance from a global optimum to a possible local optimum.

C. Online-Learning-Based Joint Network Configuration

Now, we combine Sec. IV and Sec. V-A to design online-learning-based joint network configuration in Algo. 1. In Algo. 1, let $\mathcal{N}(t, n)$ be the neighborhood of node n in slot t. Then, given a network state $\mathbf{S}^{(t)}$, we obtain control variables $\mathbf{A}^{(t)}$ by Module 2 for each time slot t.

In Lines 6-7, we use mini-batch to train the cell utility model $r(\cdot)$. For the model, we choose a neural network with a 25-width hidden layer which has a good performance.

[1]There is a slight difference when there are two or more values of the control variable $\tilde{A}_i^{(\tau)}$ that achieve the maximum probability in (10), but still one of these optimal values will be chosen when ν is set to 0.

Module 2 Fast Converging Network Configuration

Input: $\mathbf{a}_{\text{init}}^{(\text{old})}$; N, $\mathcal{N}(\cdot)$, $\mathbf{S}^{(t)}$, $\mathcal{A}$ and $r_n(\cdot)$ for each n.
Output: $\mathbf{A}^{(t)}$.

1: **Init:** Let $\mathbf{a}^{(\text{old})} = \mathbf{a}_{\text{init}}^{(\text{old})}$, and calculate $\mathcal{N}^{(1)}(\cdot)$ by (9);
2: **repeat**
3: $\mathbf{a}^{(\text{new})} = \mathbf{a}^{(\text{old})}$;
4: **for** $n = 1$ **to** N **do**
5: $\mathrm{a}_n^{(\text{new})} = \arg\max_{\mathcal{A}} \sum_{i\in\mathcal{N}^{(1)}(n)} r(\mathbf{S}^{(t)}_{\mathcal{N}(i)}, \mathbf{a}^{(\text{new})}_{\mathcal{N}(i)})$;
6: **end for**
7: **until** $\mathbf{a}^{(\text{old})} == \mathbf{a}^{(\text{new})}$;
8: $\mathbf{A}^{(t)} = \mathbf{a}^{(\text{new})}$;

Algorithm 1 Online-Learning-Based Joint Network Configuration

Input: $\mathbf{A}^{(0)}$, N, $B_{\max}$, initial cell utility model $r(\cdot)$, and observe $\mathcal{N}(t, n)$, $S_n^{(t)}$, $R_n^{(t)}$ for each n in each slot t.
Output: $A_n^{(t)}$ for each BS n in each slot t.

1: **Init:** Mini-batch $B = \emptyset$;
2: **for** each time slot t **do**
3: Observe neighborhood structure, let $\mathcal{N}(\cdot) = \mathcal{N}(t, \cdot)$, and observe state $\mathbf{S}^{(t)}$;
4: By Module 2, obtain $\mathbf{A}^{(t)}$, with setting $\mathbf{a}_{\text{init}}^{(\text{old})} = \mathbf{A}^{(t-1)}$ and $r_n(\cdot) = r(\cdot)$ for each n;
5: Observe cell utility $R_n^{(t)}$ for each BS n;
6: Add the tuples $(\mathbf{S}^{(t)}_{\mathcal{N}(n)}, \mathbf{A}^{(t)}_{\mathcal{N}(n)}, R_n^{(t)})$ for $n = 1, 2, \cdots, N$ to the mini-batch B; if the size of B exceeds $B_{\max}$, drop the oldest tuples in it;
7: Update the cell prediction model $r(\cdot)$ by stochastic gradient descent to minimize the loss function on the mini-batch;
8: **end for**

VI. Numerical Results

We evaluate the performance of the proposed approach on an industrial-grade cellular network simulator. The simulation scenario includes 87 cells, reproducing their geographical relationship of a metropolitan area. The simulation also includes thousands of users, randomly located in the area. Various aspects are simulated, such as wireless channel model, network equipment performance, BS antenna position, MIMO, etc.

We compare network utility performance for different configuration algorithms. Two baselines are considered. One is a random policy that selects pilot power uniformly at random; the other is a distributed Q-learning algorithm.

The results of different algorithms are shown in Fig. 2. Our Algo. 1 outperforms distributed Q-learning, in terms of both the overall performance and the ramp-up time. It is better in overall performance since it jointly optimizes network control, while in distributed Q-learning each cell makes decision independently. Furthermore, Algo. 1 has a much shorter ramp-up time compared to the distributed Q-

learning algorithm. We have also tested the performance of Algo. 1 with Module 1 and found that, our Algo. 1 still approaches this Module 1-based slow algorithm.

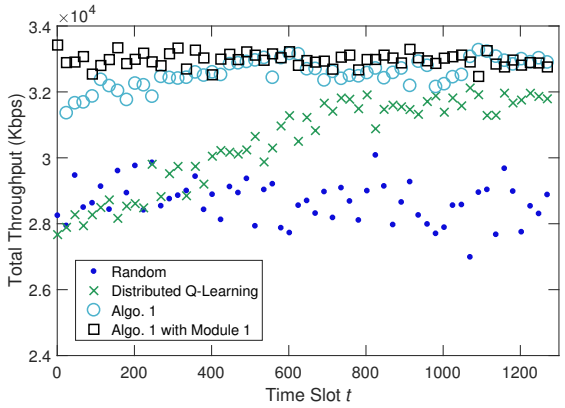

Figure 2. Total network throughput for different configuration algorithms.

VII. Conclusion

In the context of cellular network configuration, a learning-based approach needs to address a few specific challenges, namely, limited data availability, convoluted sample data, highly complex optimization due to interactions among neighboring cells, and the need to adapt to network dynamics. In this work, we develop an online-learning-based joint-optimization approach. In our approach, to learn an appropriate utility function, we develop a neural-network-based model that addresses the convoluted sample data issue and achieves good accuracy based on data aggregation. Based on the utility function learned, we formulate a global network configuration optimization problem. To solve this high-dimensional non-concave maximization problem, we design a Gibbs-sampling-based algorithm that converges to an optimal solution when a temperature parameter is small enough. To speed up its convergence, we further design an efficient algorithm that converges to a local optimum promptly. To adapt to network dynamics, we develop an online scheme that updates the learned utility function and solves the corresponding optimization problem as the network changes. To illustrate the idea, we use the case study of pilot power configuration. The simulation results show that our online utility model achieves a good prediction accuracy, and our online scheme outperforms a benchmark based on a distributed Q-learning algorithm.

The proposed approach has the potential to be applied to other similar network configuration problems, such as handoff threshold configuration, antenna adjustment, and transmission power allocation. Future research includes incorporating network change detection mechanisms, and generalizing the framework to self-organizing networks (SON).

Acknowledgment

The work was partially supported by NSF through grants CNS-1547461, CNS-1718901, and CCF-1423542.

References

[1] W. Ding and D. Yuan, "A decomposition method for pilot power planning in umts systems," in *Digital Information and Communication Technology and it's Applications (DICTAP), Second International Conference,*. IEEE, 2012, pp. 42–47.

[2] K. Valkealahti, A. Hõglund, J. Parkkinen, and A. Hamalainen, "Wcdma common pilot power control for load and coverage balancing," in *Personal, Indoor and Mobile Radio Communications. The 13th IEEE International Symposium,*, vol. 3. IEEE, 2002, pp. 1412–1416.

[3] A. Agustin, S. Lagen, and J. Vidal, "Energy efficient cell load-aware coverage optimization for small-cell networks," in *Communications (ICC), IEEE International Conference,*. IEEE, 2015, pp. 2036–2041.

[4] I. Ashraf, H. Claussen, and L. T. Ho, "Distributed radio coverage optimization in enterprise femtocell networks," in *IEEE International Conference Communications(ICC)*, 2010, pp. 1–6.

[5] I. Siomina and D. Yuan, "Soft handover overhead control in pilot power management in wcdma networks," in *Vehicular Technology Conference. VTC 2005-Spring*, vol. 3. IEEE, 2005, pp. 1875–1879.

[6] R. Razavi and H. Claussen, "Self-configuring switched multi-element antenna system for interference mitigation in femto-cell networks," in *Personal Indoor and Mobile Radio Communications (PIMRC), IEEE 22nd International Symposium on*. IEEE, 2011, pp. 237–242.

[7] C. Guestrin, M. G. Lagoudakis, and R. Parr, "Coordinated reinforcement learning," in *Proceedings of the Nineteenth International Conference on Machine Learning*, ser. ICML '02, 2002, pp. 227–234.

[8] J. R. Kok and N. Vlassis, "Collaborative multiagent reinforcement learning by payoff propagation," *J. Mach. Learn. Res.*, vol. 7, pp. 1789–1828, Dec. 2006.

[9] C.-Y. Liao, F. Yu, V. C. Leung, and C.-J. Chang, "A novel dynamic cell configuration scheme in next-generation situation-aware cdma networks," *IEEE Journal on Selected Areas in Communications,*, vol. 24, no. 1, pp. 16–25, 2006.

[10] S. C. Borst, M. G. Markakis, and I. Saniee, "Nonconcave utility maximization in locally coupled systems, with applications to wireless and wireline networks," *IEEE/ACM Transactions on Networking*, vol. 22, no. 2, pp. 674–687, April 2014.