Implementation and Optimization of Real-Time Fine-Grained Air Quality Sensing Networks in Smart City

Zhiwen Hu, Zixuan Bai, Kaigui Bian, Tao Wang, and Lingyang Song
School of Electronics Engineering and Computer Science, Peking University, Beijing, China
{zhiwen.hu, zixuan.bai, bkg, wangtao, lingyang.song}@pku.edu.cn

***Abstract*—Driven by the increasingly serious air pollution problem, the monitoring of air quality has gained much attention in both theoretical studies and practical implementations. In this paper, we present the implementation and optimization of our own air quality sensing system, which provides real-time and fine-grained air quality map of the monitored area. The objective of our optimization problem is to minimize the average joint error of the established real-time air quality map, which involves data inference for the unmeasured data values. A deep Q-learning solution has been proposed for the power control problem to reasonably plan the sensing tasks of the power-limited sensing devices online. A genetic algorithm has been designed for the location selection problem to efficiently find the suitable locations to deploy a limited number of sensing devices. The performance of the proposed solutions are evaluated by simulations, showing a significant performance gain when adopting both strategies.**

***Index Terms*—Air quality, power efficiency, reinforcement learning, genetic algorithm**

I. Introduction

Based on a recent report of the World Health Organization [1], air pollution has become one of the greatest threat to human health. The degree of air pollution is usually defined according to the concentrations of some typical air pollutants, including the fine Particulate Matters (e.g., $PM_{2.5}$) and other basic chemical substances [2]. To measure the concentrations of typical air pollutants, the governments have deployed authoritative monitoring systems across the country with high costs. Despite the high precision they can achieve, these systems only have a few number of stations over a large area and provide measurements with significant latency [3].

However, recent studies show that the concentrations of air pollutants have the intrinsic characteristics to change from meters to meters, especially in the urban areas with complicated terrain [4], [5]. It is preferred that a large number of low-cost Internet-of-Things (IoT) sensing devices are densely deployed to provide more frequent sensing [6], [7], in which way the air quality data can have a higher spatial-temporal resolution [8], [9]. The citizens can benefit from the valuable information provided by the air quality sensing system [11], by following the suggestions like keeping away from the highly polluted area or deciding the best ventilation system for a building [10].

In this paper, we present our own air quality sensing system, which provides real-time and fine-grained air quality map of the monitored area. This system has been deployed in Peking University (PKU) for 7 months and we have collected over 100 thousand data values from 30 devices. To guarantee the accuracy of the real-time and fine-grained air quality map, we study the corresponding optimization problem, where the limited number of available sensing devices and the limited capacity of their batteries are the constraints. Since related works rarely address such issues, in this paper, the optimization problem is studied in detail.

Specifically, a battery-powered sensing device can only perform limited times of power-consuming actions, such as detecting air pollutants, or uploading data to the server. To recover a real-time and fine-grained air quality map from the sparse data, a procedure of inference and estimation is required [12], [13]. The accuracy of inferring the data at unmeasured locations and unmeasured times depends on the spatial-temporal structure of the collected data. For instance, inferring the current air quality based on a measured value from long ago would be questionable [14]. In addition, inferring the air quality at a certain location based on the data from a hardly correlated location is also inaccurate [15], [16]. Therefore, it is necessary to consider the problems of where to deploy the limited number of sensing devices (location selection) and when to perform sensing actions (power control).

In our work, we model the measurement error and the inference error based on the data collected by our own system. Our objective is to minimize the joint error of the real-time and fine-grained air quality map, by properly designing the power control and location selection strategies. The power control problem is solved based on deep Q-learning by considering the system as a Markov Decision Process (MDP). The location selection problem is solved by using k-means clustering for initialization and using the genetic algorithm for improvement. Both solutions achieve satisfactory suboptimal outcomes, and the combination of them shows a significant superiority to reduce the average joint error.

The main contributions of our work are listed as below:

1) We present our real-time and fine-grained IoT air quality sensing system that has been deployed for 7 months.
2) We provide a deep Q-learning solution for the power control problem to plan the sensing tasks online.
3) We design a genetic algorithm for the location selection problem to efficiently deploy the sensing devices.

The rest of our paper is organized as follows. Section II provides an overview of our system. Section III formulates the problem of minimizing the joint error. Section IV presents the deep Q-learning solution for power control. Section V presents the genetic solution for location selection. Section VI shows the simulation results of the proposed solutions. Finally, we conclude our paper in Section VII.

978-1-5386-8088-9/19/$31.00

II. System Overview

As shown in Fig. 1, our air quality sensing system consists of four layers [17]. The sensing layer collects the data of real-time air quality, which is carried out by the sensing devices installed near the ground. The transmission layer enables the bidirectional communications, which is supported by the infrastructure of the current wireless communication networks. The processing layer is implemented in the cloud server, which is responsible to receive, record and process the data from the sensing layer, and to control the behaviour of the sensing layer. The presentation layer can provide valuable information for the users, which includes our official website and our official WeChat subscription account.

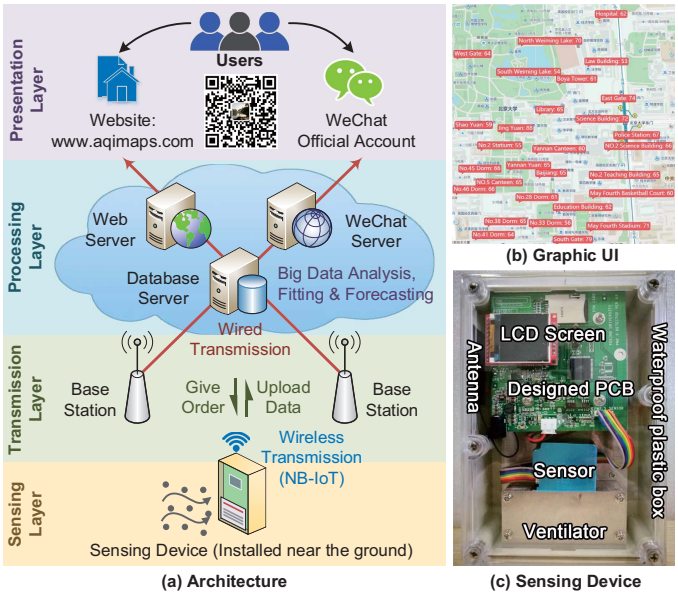

Fig. 1. An overview of our air quality sensing system deployed in PKU.

This system has been deployed in PKU since Feb. 2018. During the deployment, we have collected over 100 thousand effective values, mostly for the concentrations of PM2.5. Here we provide the data set collected by 30 on-ground sensing devices [19]. Specifically, it contains the PM2.5 values from two time periods, including the period from March 1st 2018 to May 15th 2018, and the period from June 5th 2018 to Augest 25th 2018. The provided data set is used to extract some important statistical properties of the monitored area, as given in Section III, in which way we are able to design the corresponding power control and location selection strategies.

III. Optimization Problem Formulation

A. Problem Overview

The set of all suitable sensing locations in the concerned area is given by $\mathcal{K}$, with $|\mathcal{K}| = K$. Only $L < K$ sensing devices are available to be deployed. We denote the set of locations with sensing devices as $\mathcal{K}_L$, where $\mathcal{K}_L \in \mathcal{K}$ and $|\mathcal{K}_L| = L$.

The system is divided into equal-length time slots. We provide the 0-1 matrix $\Phi_{K\times T}$ to represent the power control strategy, where T is the expected number of time slots that the whole system should sustain without recharging. As the element of the matrix $\Phi_{k\times t}$, $\phi_{k,t}=1$ indicates that the device in the k^{th} location is turned on to sense data at the t^{th} time slot, and $\phi_{k,t}=0$ indicates that this device still keeps asleep.

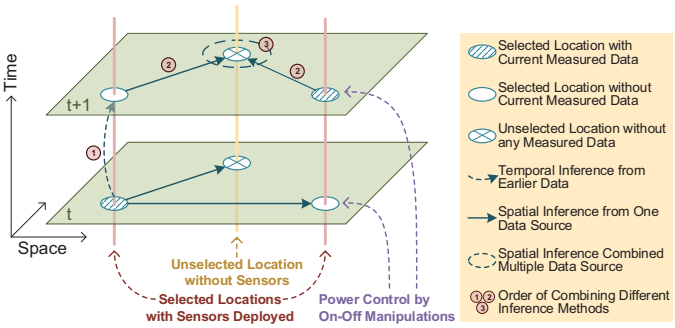

Fig. 2. The spatial-temporal model of the air quality sensing system which contains spatial and temporal inference to provide real-time air quality map.

B. Measurement and Inference Errors

We model the measurement error and inference error based on our statistical data [19]. The inference error here is modeled independently of any advanced inference algorithms (such as neural networks), in which way we can depict the most general situation. We denote the air quality at the k^{th} location at the t^{th} time slot as a random variable, given by $X_{k,t}$.

Measurement: The measurements of the sensing devices are not perfect. The distribution of the measured value (e.g., PM2.5) approximately complies to Gaussian distribution:

$$\begin{cases} X_{k,t} \sim \mathcal{N}(\mu_{k,t}, \sigma^2_{k,t}), \\ \mu_{k,t} \approx \mu_t, \\ \sigma^2_{k,t} \approx \mu_{k,t} \times \sigma_0^2, \end{cases} \quad \forall t \geq 0,\ \forall k \in \mathcal{K},\ \phi_{k,t}=1, \qquad (1)$$

where $\mu_{k,t}$ is the precise value[1], μ_t is the average value at time t, and σ_0 is a constant reflecting the error of the sensors. We can see that the precision of the measurement decreases as the air quality is getting bad ($\sigma^2_{k,t}$ is larger as μ_t is larger).

Temporal inference: With a measured value $X_{k,t}$, we can infer the value at time $t+\tau$ for the same location. The deviation of the new value from the original one can be seem as a additive random noise. For any two adjacent time slots, such a noise has a fixed distribution, given as $X_d^{t\to t+1} \sim \mathcal{N}(0, \sigma_d^2), \forall t \geq 0$, where σ_d^2 is the constant showing the average change rate of the air quality. The distribution of $X_{k,t+\tau} = X_{k,t} + X_d^{t\to t+1} + \cdots + X_d^{t+\tau-1 \to t+\tau}$ is given by

$$\begin{cases} X_{k,t+\tau} \sim \mathcal{N}(\mu_{k,t}, \sigma^2_{k,t+\tau}), \\ \sigma^2_{k,t+\tau} = \sigma^2_{k,t} + \tau\sigma_d^2, \end{cases} \quad \forall t \geq 0,\ \forall k \in \mathcal{K},\ \phi_{k,t}=1, \qquad (2)$$

implying a longer time span decreases the inference accuracy.

Spacial inference by single source: Based on $X_{k,t}$, we are able to infer the value at another location at the same time slot, $X_{k',t}$. To achieve this, we exploit the relevance among different locations from historical data and find that the deviations between two locations can also be modeled as additive. Specifically, the additive deviation from location k to location k' is denoted as $X_d^{k\to k'} \sim \mathcal{N}(\mu_{k,k',t}, \sigma^2_{k,k',t})$, where $\mu_{k,k',t} \approx \mu_t \times \mu_{k,k'}$ and $\sigma^2_{k,k',t} \approx \mu_t \times \sigma^2_{k,k'}$. Here, $\mu_{k,k'}$ is the

[1]The precise PM2.5 value can be detected by a high-precision instrument TSI8530, which is expensive and not economical to be massively deployed.

normalized average deviation from location k to location k', and $\sigma_{k,k'}$ is the normalized increased variance when using $X_{k,t}$ to infer $X_{k',t}$. Now we have the distribution of the inferred value $X_{k',t}=X_{k,t}+X_d^{k\to k'}$ as:

$$\begin{cases} X_{k',t} \sim \mathcal{N}(\mu_{k',t}, \sigma^2_{k',t}), \\ \mu_{k',t} = \mu_{k,t} + \mu_{k,k',t}, \\ \sigma^2_{k',t} = \sigma^2_{k,t} + \sigma^2_{k,k',t}, \end{cases} \quad \forall t \geq 0, \forall k, k' \in \mathcal{K}. \tag{3}$$

Spacial inference by multiple sources: We can further combine the results of single-source spatial inferences from M locations to jointly infer a value of a specific location. We denote the m^{th} single-source spatial inference for the target location k as $X_{k,t,m} \sim \mathcal{N}(\mu_{k,t,m}, \sigma^2_{k,t,m})$, $1 \leq m \leq M$. By multiplying the probability density functions (PDF) of these inference results together, we obtain the PDF of the target location k. For simplicity, we assume the distributions of different $X_{k,t,m}$ are independent. Therefore the final inference $X_{k,t}$ also has a Gaussian distribution, given as:

$$\begin{cases} X_{k,t} \sim \mathcal{N}(\mu_{k,t}, \sigma^2_{k,t}), \\ \mu_{k,t} = \dfrac{\sum_{m=1}^{M} \mu_{k,t,m}/\sigma^2_{k,t,m}}{\sum_{m=1}^{M} 1/\sigma^2_{k,t,m}}, \\ \sigma^2_{k,t} = 1/\big(\sum_{m=1}^{M} 1/\sigma^2_{k,t,m}\big), \end{cases} \quad \forall t \geq 0, \forall k' \in \mathcal{K}. \tag{4}$$

The result has a weighted mean and has a smaller variance.

Rule of inference: As shown in Fig. 2, for a measured value, no inference is performed. For an unmeasured location at current time, we consider a three-step inference. The first step is to execute temporal inference for each one of the selected locations based on their previous measured values by Eqn. (2), in which way we have L values for the current time. The second step is to utilize these values to perform L times of single-source spatial inference for the target location, based on Eqn. (3). And the final step is to obtain the multi-source spatial inference result, based on Eqn. (4).

C. Environment Model

The value of μ_t can be seen as the air quality for the whole area in a coarse-grained perspective. And we aim to establish a statistic model for the change of μ_t. From our collected data, we can obtain the the probability of air quality transition between adjacent time slots, denoted by

$$P_{y,y'} = P\big[\mu_t = y' \big| \mu_{t-1} = y\big], \quad t \in [1, T], \quad y, y' \in \mathcal{Y}, \tag{5}$$

where $\mathcal{Y}$ is the value space of the possible air quality. The values of air quality (such as PM2.5) are usually in the form of integer, thus we consider $\mathcal{Y}$ as a finite discrete value space.

It is assumed that μ_t can be roughly known when it comes to the t^{th} time slot. The corresponding approaches could be neural networks [13], or checking the official weather report (which is not our focus in this paper). We focus on how to increase the accuracy of the fine-grained air quality map by power control and location selection, given as follows.

D. Problem of Power Control and Location Selection

We assume that each sensing device can only perform E times of sensing tasks before its battery dies, where $E < T$. In addition, we expect that each device should not be silent for too long. The maximum number of consecutive time slots that a device can keep asleep is S. We guarantee that $S \cdot E > T$ to avoid contradiction.

Since the server needs to provide a *real-time* distribution of the air quality, the incomplete data at the unmeasured locations at the current time should be estimated according to the spatial and the temporal inference mentioned in Section III-B. For $X_{k,t}$, we define its *joint error*, $J_{k,t}$, as the indicator to quantitatively show reliability of the data, given as

$$J_{k,t} = \sqrt{\sigma^2_{k,t} + (\mu_{k,t} - \mu_t)^2}, \tag{6}$$

which jointly considers the variance of the value and the deviation from the average value. A larger variance or a larger deviation could reduce the reliability of the data.

Our objective is to minimize the average joint error of the real-time air quality map, given by

$$\min_{\mathcal{K}_L} \min_{\{\phi_{k,t}\}} \quad \bar{J} = T^{-1} K^{-1} \sum_{t=1}^{T} \sum_{k \in \mathcal{K}} J_{k,t}, \tag{7}$$

$$s.t. \sum_{t=0}^{T} \phi_{k,t} \leq E, \quad \forall k \in \mathcal{K}_L, \tag{8}$$

$$\sum_{t}^{t+S+1} \phi_{k,t} \geq 1, \quad \forall k \in \mathcal{K}_L, \forall t \in [0, T-S-1], \tag{9}$$

$$\phi_{k,t} = 0, 1 \quad \forall k \in \mathcal{K}_L, \forall t \in [0, T], \tag{10}$$

$$\phi_{k,t} = 0, \quad \forall k \notin \mathcal{K}_L, \forall t \in [0, T], \tag{11}$$

$$|\mathcal{K}_L| = L, \quad |\mathcal{K}_L| \in \mathcal{K}, \tag{12}$$

where we assume all the sensors should perform sensing at $t=0$ for a good initialization and the situation at $t=0$ is not counted. Eqn. (8) implies the power budget constraint, Eqn. (9) indicates the maximum silence time span, and Eqn. (12) shows the constraints of location selection.

The joint optimization of power control and location selection is highly intractable. Therefore, in the following part of this paper, we separate the problem into the power control problem and the location selection problem. Specifically, we first study the problem of power control in a stochastic environment based on a fixed location selection in Section IV. Next, in Section V, we study the problem of location selection based on a fixed power control strategy in a given environment. By combining the solutions of these problems together, it is expected that a satisfactory outcome can be acquired.

IV. Power Control Strategy

The power control problem can be transformed into a Markov Decision Process (MDP), which consists of the set of states $\mathcal{S}$, the set of available actions $\mathcal{A}$, the state transition probability matrix $\mathcal{P}$, and the set of reward functions $\mathcal{R}$.

Definition 1. *In the power control problem with L sensing devices, the i^{th} system state in the whole state transition history is defined in the following form:*

$$S_i = (S_i^t, \vec{S_i^p}, \vec{S_i^d}, \vec{S_i^r}, S_i^e, S_i^l), \tag{13}$$

which has six components. The integer $S_i^t \in [0, T+1]$ represents the time of the system. The L-length integer vector $\vec{S_i^p}$ indicates the remaining power of each sensing device, with $\vec{S_i^p}(l) \in$

$[0, E]$, $\forall 1 \le l \le L$. *The L-length integer vector* $\vec{S_i^d}$ *shows the number of time slots since the last time of measurement for each device, with* $\vec{S_i^d}(l) \in [0, S]$, $\forall 1 \le l \le L$. *The L-length integer vector* $\vec{S_i^r}$ *records the average air quality value during the last time of measurement for each device,* $\vec{S_i^r}(l) \in \mathcal{Y}$, $\forall 1 \le l \le L$. *The integer* $S_i^e \in \mathcal{Y}$ *shows the current average air quality* μ_t. *And the integer* $S_i^l \in [1, L]$ *implies who's turn it is to take the action at this state*[2].

Initial state: The initial state is $S_1 = (1, \vec{S_1^p}, \vec{S_1^d}, \vec{S_1^r}, \mu_1, 1)$, where $\vec{S_1^p}(l) = E$, $\vec{S_1^d}(l) = 1$, $\vec{S_1^r}(l) = \mu_0$, $\forall 1 \le l \le L$.

Action set: For each state $S_i = (S_i^t, \vec{S_i^p}, \vec{S_i^d}, \vec{S_i^r}, S_i^e, S_i^l)$, two actions can be performed, given by $\mathcal{A} = \{a_0, a_1\}$, where a_0 represents keeping the $(S_i^l)^{th}$ device asleep and a_1 represents turning the $(S_i^l)^{th}$ device on. If $S_i^p(S_i^l) > 0$ and $S_i^d(S_i^l) = S$, meaning that the $(S_i^l)^{th}$ device has been asleep long enough and still have power, then only action a_1 can be executed. If $S_i^p = 0$, meaning that the $(S_i^l)^{th}$ device has no power, then only a_0 can be executed. For other cases, both a_0 and a_1 can be chosen for the $(S_i^l)^{th}$ device.

State transition for $S_i^l < L$: For the current state $S_i = (S_i^t, \vec{S_i^p}, \vec{S_i^d}, \vec{S_i^r}, S_i^e, S_i^l)$, if a_0 is performed, then for the next state S_{i+1}, we have $S_{i+1}^l = S_i^l + 1$ and the other components the same as S_i. If a_1 is performed, then for the next state S_{i+1}, we have $S_{i+1}^l = S_i^l + 1$, $S_{i+1}^p(S_i^l) = S_i^p(S_i^l) - 1$, $S_{i+1}^d(S_i^l) = 0$, $S_{i+1}^r(S_i^l) = S_i^e$, and other components the same as S_i.

State transition for $S_i^l = L$: Fort the current state $S_i = (S_i^t, \vec{S_i^p}, \vec{S_i^d}, \vec{S_i^r}, S_i^e, S_i^l)$, if a_0 is performed, then $S_{i+1}^t = S_i^t + 1$, $S_{i+1}^l = 1$, $\vec{S_{i+1}^d} = \vec{S_i^d} + 1$, $S_{i+1}^e \sim P[\mu_t | \mu_{t-1} = S_i^e]$, and the other components the same as S_i. If a_1 is performed, then $S_{i+1}^t = S_i^t + 1$, $S_{i+1}^l = 1$, $S_{i+1}^p(L) = S_i^p(L) - 1$, $S_{i+1}^r(L) = S_i^e$, $S_{i+1}^d(l) = S_i^d(l) + 1$ for $1 \le l \le L-1$, $S_{i+1}^d(L) = 1$, $S_{t+1}^e \sim P[\mu_t | \mu_{t-1} = S_i^e]$, and the other components the same as S_i.

Termination condition: When it comes to $S_i^t = T + 1$, the state transition terminates and no more actions are needed.

Reward: For each state s, there is a reward when taking an action a, denoted as R_s^a. Specifically, for the state with $S_i^l = 1$, the reward is defined as $-\sum_{k \in \mathcal{K}} J_{k,t}$ after the 1^{th} device takes its action. For the state with $S_i^l > 1$, the reward is defined as the marginal decrease of $\sum_{k \in \mathcal{K}} J_{k,t}$ after the corresponding device takes its action. The sum of L rewards with a certain time slot t equals to the final value of $-\sum_{k \in \mathcal{K}} J_{k,t}$ after all the L devices take their actions.

State value function: For each state, there is a value function $V(S)$, representing the utility of this state. Specifically, the termination state has zero utility. For each intermediate state, if $s \to s'$ with action a and reward R_s^a, then we have $V(s) = R_s^a + V(s')$. For the initial state, S_1, the state value function is $V(S_1) = \sum_i R_{S_i}^a = -\sum_{t=1}^{T} \sum_{k \in \mathcal{K}} J_{k,t}$. We can see that, maximizing the value of $V(S_1)$ by properly selecting the action for each state is the same as minimizing the average joint error $\bar{J}$ as the objective function describes.

[2]We arrange the sensing devices to take actions in a predefined order at each time slot. This will not influence the potential optimal result as long as the reward is correctly defined.

State-action value function: The value of performing action a on state s is denoted as $Q(s, a)$. By further taking into account "take the best action for each state", we have

$$V(s) = \max_a \big[Q(s, a)\big], \tag{14}$$

$$Q(s, a) = R_s^a + \sum\nolimits_{s'} \mathcal{P}_{ss'}^a V(s'), \tag{15}$$

where the state transition probability is also considered.

Approximation of Q(s, a): Although there exists an optimal solution to calculate the best solution, the computation complexity increases exponentially with the number of devices L. We use a deep neural network (NN) to approximate $Q(s, a)$ and we denote the approximated $Q(s, a)$ as $\tilde{Q}(s, a)$. This neural network uses the feature vector $\vec{f}(s, a)$ of a state-action pair as the input and calculates the corresponding $\tilde{Q}(s, a)$. We denote the neural network model as $\tilde{Q}(s, a) = \mathcal{NN}\big(\vec{f}(s, a)\big)$.

Feature design: A feature vector has multiple real number components, showing some properties of a given state-action pair. In our implementation, the length of the feature vector is $5L + K + 3$, where K is the number of all the locations. Due to space limit, details are omitted in this paper and they can be found in our journal paper [18].

Training of the NN: We first use a random power control strategy to experience through the state transition procedure. In this way, a set of $\big\langle \vec{f}(s, a), Q(s, a) \big\rangle$ values can be collected. Then we perform a training process to minimize $\sum \big[\mathcal{NN}\big(\vec{f}(s, a)\big) - Q(s, a)\big]^2$ and get the initial training $\mathcal{NN}$ model. Next, we use the $\mathcal{NN}$ model to perform action decision in a new system, with probability ϵ to select a random action and with $0 < 1 - \epsilon < 1$ to select action $a = \arg\max \tilde{Q}(s, a)$. In this way, we can collect a new episode of the experience and record the tuples of $< \vec{f}(s, a),\ R_s^a,\ \tilde{Q}(s', a') >$ along the experienced states. The recorded tuples also form a new training set $\big\{\big\langle \vec{f}(s, a),\ R_s^a + \tilde{Q}(s', a') \big\rangle\big\}$. We can repeatedly use the $\mathcal{NN}$ to perform action decision in new systems and create multiple sets of training data, just like the way in [20]. As the training goes on, the accuracy of $\tilde{Q}(s', a')$ improves and the adopted action for each state, i.e., $\pi(s) = \arg\max_a \tilde{Q}(s, a)$, can also lead to a lower average joint error $\bar{J}$.

V. Location Selection Strategy

The problem of location selection is solved by first using a clustering algorithm to select initial locations and then using a genetic algorithm to iteratively improve the performance. Here, we utilize μ_{k_1,k_2} and σ_{k_1,k_2}^2 to quantitatively describe the statistic relation of the locations k_1 and k_2. A higher value of $|\mu_{k_1,k_2}|$ or a higher value of σ_{k_1,k_2}^2 indicates that the air quality at k_1 and the air quality at k_2 have lower similarity.

High dimensional feature space: For any two locations, $k_1, k_2 \in \mathcal{K}$, we defined θ_{k_1,k_2} as their distance, given by

$$\theta_{k_1,k_2} = \sqrt{\mu_{k_1,k_2}^2 + \sigma_{k_1,k_2}^2}, \quad k_1, k_2 \in \mathcal{K}. \tag{16}$$

A greater distance indicates less similarity[3]. We need to decide the coordinate of each location $k \in \mathcal{K}$, denoted by

[3]Note that to avoid the case like $\theta_{1,3} > \theta_{1,2} + \theta_{2,3}$ (which is not acceptable in Euclidean space), we can add a same small amount value $\delta = \theta_{1,3} - \theta_{1,2} - \theta_{2,3}$ to all the values of θ_{k_1,k_2}, $\forall k_1 \ne k_2$.

$\vec{x}_k = (x_k^{(1)}, x_k^{(2)}, \cdots, x_k^{(D)})$, where D is the dimension of the feature space, which is set to be $D = K-1$. The coordinate of the first location is set as $\vec{x}_1 = (0, 0, \cdots, 0)$. The coordinate of the second location is $\vec{x}_2 = (\theta_{1,2}, 0, \cdots, 0)$. For the k^{th} coordinate $\vec{x}_k$, we need to calculate the solution of a $(k-1)$-variable quadratic equation set, where the distances from $\vec{x}_k$ to all the previous coordinates are the conditions.

Clustering and initial sets: The L locations that we aim to select from $\mathcal{K}$ should be "as separated as possible". Therefore, we use the classical k-means algorithm to cluster the K locations in the feature space into L clusters. We then randomly choose one location from each cluster to create a location set set $\mathcal{L}$, with $|\mathcal{L}| = L$. Since there could be many possible $\mathcal{L}$, we denote the collection of the location sets as

$$\begin{cases} \mathcal{C} = \{\mathcal{L}_c, \}, & \forall c = 1, 2, \cdots C, \\ \mathcal{L}_c = \{k_l\}, & \forall k_l \in \mathcal{K}\ , \forall l = 1, 2, \cdots L. \end{cases} \tag{17}$$

Therefore, $\mathcal{C}$ becomes the initial sets of the location selection.

Genetic coding: For each location set $\mathcal{L}$, we use a K-length vector to indicate the corresponding coded gene, given by

$$\vec{G}(\mathcal{L}) = \big(G^{(1)}(\mathcal{L}), G^{(2)}(\mathcal{L}), \cdots, G^{(K)}(\mathcal{L})\big), \tag{18}$$

where $G^{(k)}(\mathcal{L})$ is a boolean function, indicating whether the k^{th} location is included in the location set $\mathcal{L}$. For each valid location set (i.e., up to L selected locations), the number of "1"s in its coded gene is no more than L. For the initial collection of location sets $\mathcal{C}$, we encode them and add their genes to a gene pool, denoted by $\mathcal{G}$.

Genetic mutation: Each gene creates M copies of itself, where each bit of the gene has a fixed possibility p_m to mutate during replication ($0 \to 1$, or $1 \to 0$). Therefore, different genes (solutions) are generated and added to the gene pool $\mathcal{G}$. Note that the number of 1 may not satisfy the constraint, which will be considered in the following genetic selection part.

Genetic recombination: For any two of the existing genes $\vec{G}_1$ and $\vec{G}_2$, we randomly choose a position $g = 1, 2, \cdots K-1$ and exchange the bits of these two genes behind the position g. The new genes are $\vec{G}_1' = \big(G_1^{(1)}, \cdots, G_1^{(g)}, G_2^{(g+1)}, \cdots, G_2^{(K)}\big)$ and $\vec{G}_2' = \big(G_2^{(1)}, \cdots, G_2^{(g)}, G_1^{(g+1)}, \cdots, G_1^{(K)}\big)$. These new genes are also added into the gene pool $\mathcal{G}$.

Genetic selection: First, the duplicated genes and the genes with more than L "1"s are eliminated. Then, we check the *disadvantage* of each existing gene, which is defined as $\bar{J}$ of the corresponding location set based on the given power control and environment. According to the disadvantage of each existing gene, we keep the best H_1 ones in the gene pool, and then randomly choose H_2 genes according to each gene's *weighted probability*. Such probability is defined as

$$p_g \propto \left[\left(\max_g \bar{J}\right) - \bar{J}_g\right], \qquad \forall g \in \mathcal{G}. \tag{19}$$

Therefore, the gene pool only keeps $H = H_1 + H_2$ genes after each round of genetic selection.

Evolution and termination: For each round of genetic evolution, we first perform the genetic mutation for each existing gene and the genetic recombination for all the gene pairs. We then perform genetic selection to keep only H genes and record the lowest *disadvantage* value. If the best performance hasn't been improved for 6 rounds, the evolution process terminates. Otherwise, we repeat the above process.

VI. Emulation

The parameters σ_0, σ_d, $\{\mu_{k,k'}\}$ and $\{\sigma^2_{k,k'}\}$ are extracted from our collected data, which is based on 30 air quality sensing devices deployed in Peking University for 7 months [19]. A detailed calculation of these parameters are provided in our journal paper [18]. The sensing interval (the length of the time slot) is 10 minutes, and each collected data is an integer representing the detected PM2.5 value at the given location and time. The deep neural network is designed to have 5 hidden layers. Given the location number K and device number L, each hidden layer is designed to have $4K+L, 4K, 3K, 2K$, and K neurons, respectively (from the input side to the output side). A more detailed parameter setting is listed in Table I.

TABLE I
Simulation Parameters

Normalized measurement variance, σ_0^2	0.1734
temporal deviation variance, σ_d^2	14.44
Normalized mean relation, $\{\mu_{k,k'} \vert k \neq k'\}$	between -0.15 and 0.15
Normalized variance relation, $\{\sigma^2_{k,k'} \vert k \neq k'\}$	between 0.12 and 0.38
Air quality values, $\{\mu_t\}$	between 1 and 508
Total number of time slots, T	between 500 and 10000
Total number of locations, K	30
Number of available devices, L	between 5 and 25
Energy budget of each device, E	between 100 and 1000
Maximum allowable sleep time slots, S	between 10 and 12
Random action probability, ϵ	0.1
Size of gene pool, H	between 40 and 100
Two types of genetic selection, H_1 and H_2,	$0.1H$ and $0.9H$
Copy number during genetic mutation, M	3
Mutation probability for each bit, p_m	0.1

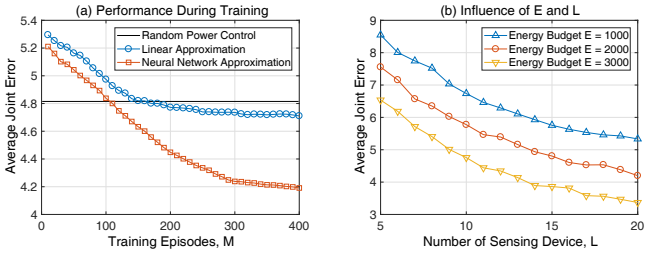

Fig. 3. The performance power control. (a) shows performance improvement during the training procedure. (b) shows the influence of the energy budget and the number of available devices.

Power control: Fig. 3 provides the performance of the proposed power control strategy based on Q-learning. Subplot (a) shows the performance improvement during the iterative training process, with $T = 10000$, $E = 2000$, $S = 12$, $K = 30$ and $L = 20$. The linear approximation only has a minor advantage over the random power control scheme, while the proposed deep Q-learning scheme shows a much promising performance. Subplot (b) presents the influence of the number of sensing devices and the energy budget of the sensing devices, with $T = 10000$, $S = 12$, and $K = 30$. The number of devices L and the energy budget E both have negative correlations with the average joint error $\bar{J}$.

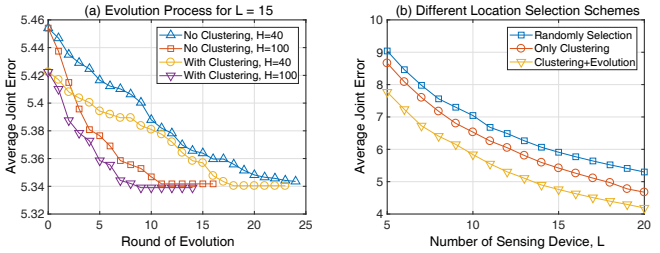

Fig. 4. The performance of location selection. (a) shows the revolution process with/without initial clustering and different size of gene pool H. (b) shows the influence of the number of available devices L.

Location selection: In Fig. 4, we present the location selection strategy based on the proposed genetic algorithm, with $p_m = 0.1$ and $M = 3$. Subplot (a) shows the improvement of performance during the evolution. It can be seen that by using the clustering algorithm, the starting point of the system has a better performance and therefore leads to a shorter convergence time. In addition, the 100-sized gene pool has a quicker evolution speed compared with the 40-sized gene pool, at the cost of a higher memory occupation. Subplot (b) shows that both clustering and evolution contribute to the performance. With more devices being deployed, the performance also gets better.

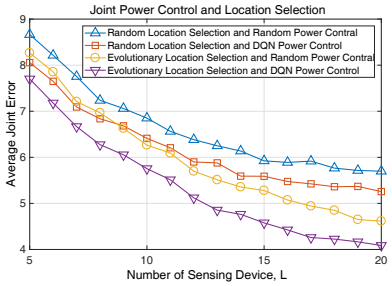

Fig. 5. he result of the combination of location selection and power control.

Combination of power control and location selection: The joint usage of location selection and power control strategies is presented in Fig. 5, with $T = 10000$, $S = 12$, $E = 2000$, $H = 100$, and $K = 30$. The upmost curve that represents the random location selection and random power control has the highest average joint error $\bar{J}$. For a specific power control strategy, the evolutionary location selection outperforms the random location selection. And for a specific location selection strategy, the deep Q-learning power control outperforms the random power control. In addition, the evolutionary location selection has a more obvious gain compared to the deep Q-learning power control when the value of L is high, which causes the crossover of the second and the third curve.

VII. Conclusion

In this paper, we present the implementation and optimization of our real-time fine-grained air quality sensing system. The proposed power control strategy was based on deep Q-learning by re-modeling the problem as a MDP. And the proposed location selection strategy was based on genetic evolutionary algorithm which widely search the solution space. To evaluate the proposed solution, we extracted the properties from our data set based on our own air quality sensing system deployed in Peking University. The simulation result showed that the proposed power control strategy provided a satisfying performance after learning 200 episodes. And the proposed location selection could quickly achieve a suboptimal solution only by using a small gene pool with the size of 100.

VIII. Acknowledgement

This work was supported by the National Nature Science Foundation of China under grant number 61625101 and CERNET Innovation Project NGII 20161011.

References

[1] World Health Organization, "7 Million Premature Deaths Annually Linked to Air Pollution," *Air Quality Climate Change*, vol. 22, no. 1, pp. 53-59, Mar. 2014.

[2] G. Kyrkilis *et al.*, "Development of an Aggregate Air Quality Index for an Urban Mediterranean Agglomeration: Relation to Potential Health Effects", *Environment International*, vol. 33, no. 5, pp. 670-676, 2007.

[3] Beijing MEMC. (Jul. 2018). *Beijing Municipal Environmental Monitoring Center*. [Online]. Available: http://www.bjmemc.com.cn/

[4] T. Quang *et al.*, "Vertical Particle Concentration Profiles Around Urban Office Buildings," *Atmospheric Chemistry and Physics*, vol. 12, no. 11, pp. 5017-5030, May 2012.

[5] T. N. Quang *et al.*, "Vertical Particle Concentration Profiles around Urban Office Buildings," *Atmos. Chem. Phys.*, vol. 12, no. 11, pp. 5017-5030, May 2012.

[6] Y. Hu *et al.*, "BlueAer: a Fine-Grained Urban PM2.5 3D Monitoring System using Mobile Sensing," in *Proc. IEEE INFOCOM*, San Francisco, CA, Jul. 2016.

[7] Y. Zheng *et al.*, "U-Air: When Urban Air Quality Inference Meets Big Data," in *Proc. 19th ACM SIGKDD Int. Conf. Knowl. Disc. Data Min. (KDD)*, Chicago, IL, USA, Aug. 2013.

[8] Y. Yang *et al.*, "Real-Time Profiling of Fine-Grained Air Quality Index Distribution Using UAV Sensing," *IEEE Internet of Things*, vol. 5, no. 1, pp. 186-198, Feb. 2018.

[9] Y. Yang *et al.*, "Arms: A Fine-Grained 3D AQI Realtime Monitoring System by UAV," in *Proc. IEEE Global Communications Conference*, Singapore, Dec. 2017.

[10] C. Borrego *et al.*, "How Urban Structure can Affect City Sustainability from an Air Quality Perspective," *Environmental modelling & software*, vol. 21, no. 4, pp. 461-467, Apr. 2006.

[11] L. Zhou, *et al.*, "When Computation Hugs Intelligence: Content-Aware Data Processing for Industrial IoT," *IEEE Internet of Things Journal*, vol. 5, no. 3, pp. 1657-1666, June 2018.

[12] Y. Zheng *et al.*, "Forecasting Fine-Grained Air Quality Based on Big Data," in *Proc. ACM Int. Conf. on Knowledge Discovery and Data Mining*, Sydney, Australia, Aug. 2015.

[13] Y. Li *et al.*, "Prediction of High Resolution Spatial-Temporal Air Pollutant Map from Big Data Sources," in *Proc. Int. Conf. Big Data Comput. Commun.*, Taiyuan, China, pp. 273C282, Jul. 2015.

[14] Y. Yang *et al.*, "AQNet: Fine-Grained 3D Spatio-Temporal Air Quality Monitoring by Aerial-Ground WSN", in *Proc. IEEE Int. Conf. on Comput. Commun.*, Honolulu, USA, Apr. 2018.

[15] H. Hsieh *et al.*, "Inferring air quality for station location recommendation based on urban big data," in *Proc. ACM Int. Conf. on Knowledge Discovery and Data Mining*, Sydney, Australia, Aug. 2015.

[16] Y. Yang *et al.*, "Sensor Deployment Recommendation for 3D Fine-Grained Air Quality Monitoring using Semi-Supervised Learning," in *Proc. IEEE Int. Conf. on Commun.*, Kansas, USA, May 2018.

[17] Z. Hu *et al.*, "Aerial-Ground Air Quality Sensing: Architecture, Technologies and Implementation, submitted to *IEEE Network Magazine*, currently available on https://arxiv.org/abs/1809.03746.

[18] Z. Hu *et al.*, "Real-Time Fine-Grained Air Quality Sensing Networks in Smart City: Design, Implementation and Optimization, submitted to *IEEE IoT Journal*, available on http://arxiv.org/abs/1810.08514.

[19] Dataset Collected from Peking University. [Online]. Available: https://github.com/pku-hzw/PKU-Air-IoTJournal-2018.

[20] V. Mnih *et al.*, "Playing Atari with Deep Reinforcement Learning", *arXiv preprint*, arXiv:1312.5602, Dec. 2013.

[21] D. E. Goldberg *et al.*, "Genetic Algorithm in Search Optimization and Machine Learning," *Machine learning*, vol. 3, no. 2, pp. 95-99, 1988.