Multi-dimensional urban sensing in sparse mobile crowdsensing

Wenbin Liu, Yongjian Yang, En Wang, Leye Wang, Djamal Zeghlache, Daqing Zhang

IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received May 6, 2019, accepted June 9, 2019, date of publication June 21, 2019, date of current version July 8, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2924184

Multi-Dimensional Urban Sensing in Sparse Mobile Crowdsensing

WENBIN LIU[1,4], YONGJIAN YANG[1], EN WANG[1], LEYE WANG[2,3], DJAMAL ZEGHLACHE[4], AND DAQING ZHANG[2,3,4], (Fellow, IEEE)

[1]College of Computer Science and Technology, Jilin University, Changchun 130012, China
[2]Key Laboratory of High Confidence Software Technologies, Peking University, Beijing 100871, China
[3]School of Electronic Engineering and Computer Science, Peking University, Beijing 100871, China
[4]R2SM, Télécom SudParis/IMT, 91000 Evry, France

Corresponding author: Daqing Zhang (daqing.zhang@telecom-sudparis.eu)

This work was supported in part by the National Natural Science Foundation of China under Grant 61772230, in part by the Natural Science Foundation of China for Young Scholars under Grant 61702215, in part by the China Postdoctoral Science Foundation under Grant 2017M611322 and Grant 2018T110247, in part by the Chinese Scholarship Council under Grant 201706170165, in part by the NSFC under Grant 61572048 and Grant 71601106, and in part by the Hong Kong ITF under Grant ITS/391/15FX.

ABSTRACT Sparse mobile crowdsensing (MCS) is a promising paradigm for the large-scale urban sensing, which allows us to collect data from only a few areas (cell selection) and infer the data of other areas (data inference). It can significantly reduce the sensing cost while ensuring high data quality. Recently, large urban sensing systems often require multiple types of sensing data (e.g., publish two tasks on temperature and humidity respectively) to form a multi-dimensional urban sensing map. These multiple types of sensing data hold some inherent correlations, which can be leveraged to further reduce the sensing cost and improve the accuracy of the inferred results. In this paper, we study the multi-dimensional urban sensing in sparse MCS to jointly address the data inference and cell selection for multi-task scenarios. We exploit the intra- and inter-task correlations in data inference to deduce the data of the unsensed cells through the multi-task compressive sensing and then learn and select the most effective ⟨cell, task⟩ pairs by using reinforcement learning. To effectively capture the intra- and inter-task correlations in cell selection, we design a network structure with multiple branches, where branches extract the intra-task correlations for each task, respectively, and then catenates the results from all branches to capture the inter-task correlations among the multiple tasks. In addition, we present a two-stage online framework for reinforcement learning in practical use, including training and running phases. The extensive experiments have been conducted on two real-world urban sensing datasets, each with two types of sensing data, which verify the effectiveness of our proposed algorithms on multi-dimensional urban sensing and achieve better performances than the state-of-the-art mechanisms.

INDEX TERMS Sparse mobile crowdsensing, reinforcement learning, compressive sensing, urban sensing.

I. INTRODUCTION

With rapid development of mobile devices and wireless communications, a practical sensing paradigm, called Mobile CrowdSensing (MCS) [1], has been proposed to leverage the large number of mobile devices carried by the users to perform the large-scale urban sensing tasks, such as monitoring of environment [2], [3], infrastructure status [4] and traffic congestion [5], etc. In order to provide high-quality urban sensing services, traditional MCS applications need to recruit a great number of users to cover all the urban subareas (cells) [2], [4]–[7]. However, in practice, we cannot afford to recruit so many users due to the large cost involved and sometimes there are no available users in the required sensing areas. Hence, researchers propose the Sparse MCS [8]–[12], which can collect data from only a few cells and infer the data of the rest cells. In this way, the number of required users can be significantly reduced while high quality urban sensing services can still be achieved.

Recently, large urban sensing systems require multiple types of sensing data (*e.g.*, publish two tasks on temperature and humidity respectively) to form a multi-dimensional urban sensing map [3], [13], [14]. Under these multi-task scenarios, there exist both intra-task correlations within the same type of sensed data (*e.g.*, the closer cells often have the similar sensed temperature/humidity readings) and inter-task correlations

The associate editor coordinating the review of this manuscript and approving it for publication was Huan Zhou.

among different types of sensed data (*e.g.*, a higher temperature usually leads to a lower humidity reading [15]). Exploiting the intra- and inter-task correlations can enhance the Sparse MCS, which further helps to reduce the number of sensed areas and improve the accuracy of the inferred results.

However, most of existing works in Sparse MCS focus on the single-task scenarios while ignoring the multi-task requirements [8], [9], [12], [16]–[18]. Rana *et al*. [16] used incomplete and random crowdsourcing data to recover the urban noise map. Zhu *et al*. [17] focused on traffic estimation from the periodically collected data by probe vehicles. Wang *et al*. [8], [9] presented a Sparse MCS framework which achieved good performances on temperature, air quality and traffic monitoring datasets, respectively. He and Shin [18] used Bayesian compressive sensing to construct the urban signal map. With Sparse MCS, these works select a few effective cells to sense (*cell selection*), and then use the sensed data to infer the full map with high quality (*data inference*). However, all of them focus on the single-task scenarios, without considering the multi-task requirements on both cell selection and data inference.

In this paper, we propose several approaches to jointly address the data inference and cell selection in multi-task scenarios, in order to make full use of the intra- and inter-task correlations and provide high-quality urban sensing services. The basic idea is to try out all of the possible ⟨cell, task⟩ pairs[1] to sense, exploit intra- and inter-task correlations to deduce the unsensed values and record the inferred errors by using the historical data. Thus, we can sense the ⟨cell, task⟩ pairs which can help most through trial and error, and then use the sensed values from the most effective ⟨cell, task⟩ pairs to deduce the unsensed values. In fact, trial and error is exactly the fundamental idea of Reinforcement Learning (RL) [19], which takes a sequence of *actions* under certain *states* so as to maximize the cumulative *rewards*. RL learns and obtains that sequence by trying out different actions and observes the rewards under each state. In our multi-task Sparse MCS, we select a sequence of ⟨*cell, task*⟩ *pairs to be sensed* (actions) considering the *data already collected* (states), in order to minimize the average *inferring errors* (rewards). In this way, we first exploit intra- and inter-task correlations in data inference, then compare and select the ⟨cell, task⟩ pairs which perform best on data inference, by applying RL to jointly address the cell selection and data inference.

In fact, the previous works mainly focus on data inference, without jointly considering cell selection. Wang *et al*. [8], [9] proposed an online method, called Query-By-Committee (QBC), which uses several inference algorithms to deduce the data of all unsensed cells, and then chooses the most uncertain one to sense, in which the inferred data of various algorithms have the largest variance. He and Shin [18] and Liu *et al*. [20] intended to select the cells with more difference between the last and current cycles under the same sensing cost. However, the uncertain or hard to infer cells are not definitely equal to the effective ones. For example, a central cell may be not hard to infer, but it usually helps the most on data inference. Also, the cell selection problem should be more complicated under multi-task scenarios. By using RL, we can directly connect the cell selection with data inference, *i.e.*, select the ⟨cell, task⟩ pairs which can help most through trial and error, and thus achieve better performance.

To effectively employ RL to capture intra- and inter-task correlations in cell selection, we further design a network structure with multiple branches to approximate the actions' rewards under a certain state. Each branch can be seen as a single-task network, which is used to extract intra-task correlations to approximate the reward for each task. Then, we concatenate the results from all branches to fuse the information from the multiple tasks. Finally, we capture the inter-task correlations among different tasks to estimate the rewards for all tasks. The network structure with multiple branches deals with the large action space (⟨cell, task⟩ pairs) and effectively captures intra- and inter-task correlations. Moreover, we first train the branches in parallel and then train the whole network for multiple tasks, in order to reduce the training workload and avoid over fitting. Furthermore, new tasks can be easily added by adding new branches and leveraging the fine-tuning techniques to train them.

For the training of RL, the traditional way is to collect a large amount of training data and then conduct an offline training. In this way, RL can try out all of the possible sensed ⟨cell, task⟩ pairs and approximate their rewards under certain states, while it is obviously ineffective. Fortunately, according to our observations of the selected ⟨cell, task⟩ pairs, we find that only a few ⟨cell, task⟩ pairs are often chosen. These few ⟨cell, task⟩ pairs are more effective under most cases. Thus, we don't need too much data to train all of the possible ⟨cell, task⟩ pairs, while we can use a small amount of data to train these effective ones, which is in fact quite effective and can achieve good enough performance.[2] Based on it, we further conduct the training of RL in an online manner, *i.e.*, we design a two-stage framework for RL-assisted Sparse MCS, including training and running phases. In the training phase, we use the online method, *i.e.*, QBC used in [8], [9], to select and sense some effective ⟨cell, task⟩ pairs to provide the sensing services with acceptable quality, simultaneously, provide the training data for RL. QBC selects the approximate effective ⟨cell, task⟩ pairs, and we use these data to train the RL model effectively. After the model is well trained and achieves better performance than QBC, we move to the running phase, in which we switch the cell selection method to the RL-based algorithm.

[1] In this paper, we consider a general case that we select the ⟨cell, task⟩ pair ($task_i$ in $cell_j$) to sense, since not all participants can (or will) perform multiple tasks in the same cell.

[2] We collect data from about 20% cells in the first 100 cycles by QBC, and then use these data for RL training and can achieve the very close performances with the well-trained RL model.

In summary, this work has the following contributions:

- We formalize the joint cell selection and data inference problem in a multi-dimensional urban Sparse MCS paradigm, and propose the RL-assisted algorithms to jointly address data inference and cell selection for multi-task Sparse MCS. We exploit intra- and inter-task correlations in data inference and learn to select the most effective ⟨cell, task⟩ pairs by RL. We also design a network structure with multiple branches to capture intra- and inter-task correlations in cell selection.
- Traditional RL methods need to firstly collect a large amount of data for training, which is very ineffective for MCS tasks. Hence, we propose a novel two-stage online framework to eliminate this costly initial data collection process. In the first stage, we use QBC to select cells, while at the same time collect the data for RL training. After the RL training is finished, we move to the second stage, *i.e.*, switch to RL-based cell selection method to perform Sparse MCS.
- We evaluate the proposed algorithms on two typical urban sensing datasets with multiple tasks, and show the effectiveness of our proposed algorithms on reducing the number of sensed values and improving the accuracy of the inferred results for the multi-dimensional urban sensing.

II. RELATED WORK

A. URBAN SENSING VIA SPARSE MOBILE CROWDSENSING

Mobile CrowdSensing becomes a promising paradigm which uses the mobile devices carried by users to perform the large-scale urban sensing tasks [1], [2]. Most of the existing works need to recruit a large number of users to cover all the sensing areas, in order to provide the sensing services with high data quality [6], [7]. However, these methods cost a lot and even sometimes we cannot find participants in some areas. In order to deal with these problems, some researchers proposed to sense data in some areas and apply the data inference algorithms to infer the data of unsensed areas from the sensed data, this mobile sensing paradigm is called Sparse Mobile CrowdSensing.

Many Sparse MCS systems have been developed for urban sensing. Rana *et al*. [16] used compressive sensing to recover the urban noise map from the incomplete and random crowdsourcing data. Zhu *et al*. [17] proposed an approach for traffic estimation from the periodically collected locations and speeds in probe vehicles. Leye Wang *et al*. [8], [9] presented a complete framework for Sparse MCS by using the compressive sensing, Bayesian inference, and active learning techniques, which achieved very good performances on temperature, air quality and traffic monitoring datasets. He and Shin [18] proposed a signal map crowdsensing framework, which uses Bayesian compressive sensing to construct the urban signal map. All these works used the compressive sensing as the data inference method, but they focused on the single-task scenarios. Wang *et al*. [11] first considered the intradata and interdata correlations, while they only exploited the correlations to the data inference without jointly considering the cell selection. In this paper, we jointly address the data inference and cell selection for multi-task scenarios, combining compressive sensing and reinforcement learning. We use compressive sensing to infer data and use reinforcement learning to select the cells which would perform best for data inference.

B. REINFORCEMENT LEARNING

Reinforcement Learning (RL) [19] is a technique of machine learning which learns to make a sequence of decisions via trial and error. Recently, combining with the deep learning, RL becomes one of the most popular research topics and has shown its effectiveness on a wide variety of sequential decision making tasks. Mnih *et al*. [21] proposed the first deep RL model and applied it to play seven Atari 2600 games. Silver *et al*. [22], [23] applied RL and presented the famous *AlphaGo*, which was the first program to defeat world-class players in Go.

More recently, researchers pay more attention to apply RL in various areas to solve practical problems. Xiao *et al*. [24] formulate the interactions between a server and vehicles as a vehicular crowdsensing game. Then they propose the Q-learning based strategies to help server and vehicles make the optimal decisions for the dynamic game. Moreover, Liang Xiao *et al*. [25] apply Deep Q-learning to derive the optimal policy for the Stackelberg game between a MCS server and a number of smartphone users. These works mainly apply RL to deal with the games in MCS, especially the dynamic and imperfect information games. In this paper, we introduce RL into a new problem domain, Sparse MCS, where it is appropriate to apply RL to jointly address the cell selection and data inference.

III. SYSTEM MODEL AND PROBLEM FORMULATION

In this section, we first introduce the system model of the multi-dimensional urban sensing via Sparse MCS paradigm. Then, we mathematically formulate the research problem and provide an example to explain it in more details.

A. SYSTEM MODEL

We consider a multi-dimensional urban sensing scenario where the requester wants to get the multiple types of data (*e.g.* temperature, humidity) in the large-scale urban regions. The main notations used throughout this paper are illustrated in Table 1. Specifically, we have k heterogeneous sensing tasks on the large-scale target sensing areas. These target sensing areas are split into m cells (subareas) and each sensing task is divided into n sensing cycles for the whole sensing campaign. The sizes of cells and the lengths of cycles are determined according to the requirements of requesters.[3]

[3]Note that the k sensing tasks may have different sizes of cells and lengths of cycles. We can use some numerical interpolation methods to obtain the unified representation for each task, in order to better capture the inter-task correlations.

TABLE 1. Main notations used throughout the paper.

Symbol	Meaning
$D_{m\times n}$, $C_{m\times n}$, $S_{m\times n}$, $\hat{D}_{m\times n}$	The ground truth data matrix, cell selection matrix, collected data matrix and inferred data matrix, with m cells and n cycles for a certain task.
$D, C, S, \hat{D}$	The set of ground truth data matrix, cell selection matrix, collected data matrix and inferred data matrix for multiple tasks.
k, l	The number of tasks and the number of selected cells at each cycle for a certain task.
$\mathbb{T}$, $\mathbb{S}$ and $\mathbb{V}$	The temporal, spatial and value constraint matrices.
$\mathbf{s}$, $\mathbf{a}$, $\mathbf{r}$	The state, action, and reward.
$\mathbf{S}$, $\mathbf{A}$	The whole set of states and actions.
α, γ, θ, $\mathbf{D}$	The learning rate, discount factor, network parameters and memory pool.

For example, if the requester would like to know the fine-grained temperature variation in the urban regions, the cells and cycles should be set to the proper sizes, such as $1km*1km$ for each cell and update data every one hour [11].

We formalize the multi-dimensional urban sensing as the multi-task Sparse MCS, which allows us to sense only a few cells and effectively provide the multi-dimensional sensing services. For each sensing task, we use the ground truth data matrix $D_{m\times n}$ to record the true data in m cells at n cycles. The set of ground truth data matrices for k sensing tasks can be denoted by $D = \{D_1, D_2, \ldots, D_k\}$. At each cycle, we select l_i cells out of the total m cells and recruit the participants in these cells (or steer the nearby participants by some incentive mechanisms) to perform task i. We study the multi-task Sparse MCS under two constraints in practical use, *e.g.*, budget constraint and accuracy constraint, respectively. With the budget constraint, we consider a general budget $B = \sum_{i=0}^{k} l_i$ for the selected ⟨cell, task⟩ pairs at each sensing cycle, which is practical and implementable.[4] With the accuracy constraint, we should select the ⟨cell, task⟩ pairs one by one until the inferred result is expected to achieve the accuracy requirement.

We mark the cell selection results for each task by the matrix $C_{m\times n}$, where $C[i, j] = 1$ means that the cell i is selected to be sensed for a certain task and $C[i, j] = 0$ means not. The set for k sensing tasks is denoted as $C = \{C_1, C_2, \ldots, C_k\}$. Hence, we have the collected data matrix $S_{m\times n} = D \circ C$ and the set $S = \{S_1, S_2, \ldots, S_k\}$, where $\circ$ denotes the element-wise product of two matrices. Using the set of collected data matrices, we utilize the multi-task compressive sensing to exploit both the intra- and inter-task correlations to infer the values in the unsensed cells for all of the k tasks. The inferred data for certain tasks are recorded in the inferred data matrix $\hat{D}_{m\times n}$, and the set is denoted by $\hat{D} = \{\hat{D}_1, \hat{D}_2, \ldots, \hat{D}_k\}$. Therefore, we obtain the weighted inference error rate for all tasks as follow:

$$\text{Error}(D, \hat{D}) = \sum_{i=0}^{k} \|(D_i - \hat{D}_i) \circ (1/D_i)\|_1 \cdot \omega_i, \qquad (1)$$

where $D_i \circ C_i = \hat{D}_i \circ C_i$ represents the collected data and ω_i is the weight of task i. Note that we use the error rate to deal with the different metrics between multiple tasks, and other metric conversion algorithms could be easily modified. Also, we use the weights to balance the importance between different tasks.

B. PROBLEM FORMULATION

Based on the above system model, we define our research problem and focus on the multi-task Sparse MCS.

Problem (Multi-Task Sparse MCS): Under the budget constraint, given k Sparse MCS tasks with m cells and n cycles, we select a total of B ⟨cell, task⟩ pairs at each sensing cycle and use these collected data to infer the unsensed data for all tasks, with the objective of minimizing the inference errors during the whole sensing process:

$$\min \ \text{Error}(D, \hat{D}) \qquad (2)$$

subject to: satisfy $B = \sum_{i=0}^{k} l_i$ for each cycle.[5]

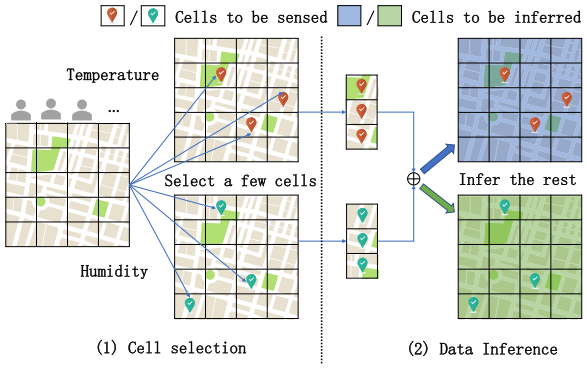


FIGURE 1. An example of multi-task Sparse MCS.

We now use an example to illustrate our multi-task Sparse MCS in more details, as shown in Figure 1. Consider that the multi-task Sparse MCS needs to get two types of sensing data, *e.g.*, temperature and humidity, in the target sensing areas, which are spilt into $5*4$ cells. For each sensing cycle, we can select a total of 6 ⟨cell, task⟩ pairs to sense and we need to infer the other $5*4*2-6=34$ values. (1) At this sensing cycle, we select 3 cells to sense the temperature readings and 3 cells to sense the humidity readings. (2) We use these collected data to infer the other 34 values at this cycle. Our goal

[4]The other cases, such as a total budget, would be easily modified by reshaping the reward in the RL-based algorithms discussed in Section IV.

[5]Similarly, the problem with accuracy constraint can be formulated as: select a minimal subset of ⟨cell, task⟩ pairs (min B) while satisfying the accuracy requirement (satisfy Error$(D, \hat{D}) \geq e$, where e represents the accuracy constraint). Actually, the problems with budget or accuracy constraints have the same goal to improve the inference accuracy, which can reduce the inference errors under the limited sensed values (budget constraint) or sense less ⟨cell, task⟩ pairs to satisfy the pre-defined accuracy requirement (accuracy constraint).

is to minimize the error rates of these 34 inferred values, by jointly addressing the data inference and cell selection in the multi-task urban sensing scenarios, *i.e.*, using multi-task compressive sensing to infer the unsensed data for all tasks simultaneously and utilize reinforcement learning-based cell selection algorithm to select the most helpful ⟨cell, task⟩ pairs for data inference, which will be introduced in Section IV.

IV. REINFORCEMENT LEARNING-ASSISTED MULTI-TASK SPARSE MOBILE CROWDSENSING

In this section, we present the reinforcement learning-assisted multi-task Sparse MCS to address the data inference and cell selection. First, we propose the multi-task compressive sensing, which can exploit the intra- and inter-task correlations to infer the unsensed values for all tasks together. Then, the reinforcement learning-based cell selection algorithm will be presented with the mathematically modeling of state, action and reward. We also design a general network structure with multiple branches for the multi-task scenarios.

A. DATA INFERENCE VIA COMPRESSIVE SENSING

1) COMPRESSIVE SENSING

Compressive sensing is a novel data inference method which has shown its effectiveness on the large-scale urban sensing tasks, such as the monitoring of traffic [17], [26], environment [8], [9], [11] and air pollution [20]. For a certain task, it can infer the full sensing matrix $\hat{D}$ from only a few collected data, based on the low-rank property:

$$\min \operatorname{rank}(\hat{D}) \tag{3}$$

$$\text{s.t.}, \quad \hat{D} \circ C = S. \tag{4}$$

With the help of Singular Value Decomposition (SVD), *i.e.*, $\hat{D} = LR^T$, we can convert the above optimization problem as follow [26]:

$$\min \lambda(\|L\|_F^2 + \|R\|_F^2) + \|LR^T \circ C - S\|_F^2, \tag{5}$$

where the regularization parameter λ allows a tunable tradeoff between rank minimization and accuracy fitness. Furthermore, we consider three important correlations in terms of temporal, spatial and value dimensions in the above optimization problem, in order to capture the intra-task correlations as follow:

$$\begin{aligned}\min\ & \lambda_r(\|L\|_F^2 + \|R\|_F^2) + \|LR^T \circ \mathcal{C} - \mathcal{S}\|_F^2 \\ & + \lambda_t\|(LR^T)\mathbb{T}^T\|_F^2 + \lambda_s\|\mathbb{S}(LR^T)\|_F^2 + \lambda_v\|\mathbb{V}(LR^T)\|_F^2,\end{aligned} \tag{6}$$

where $\mathbb{T}$, $\mathbb{S}$ and $\mathbb{V}$ are temporal, spatial and value constraint matrices, while λ_r, λ_t, λ_s and λ_v are chosen to balance the weights of different elements in the optimization problem. Then we use an alternating least squares [26] procedure to estimate L and R iteratively, in order to get the optimal $\hat{\mathcal{D}}$ ($\hat{\mathcal{D}} = LR^T$). The temporal, spatial and value constraint matrices are shown below:

- **Temporal correlation**: We consider a temporal constraint matrix $\mathbb{T}$ to capture the temporal correlations in a certain task. We choose a simple temporal constraint matrix as $T = \text{Toeplitz}(0, 1, -1)$, which intuitively expresses that two continuous sensed values from the same cell are often similar. Moreover, if we have more domain knowledge or enough historical data, we should conduct a more sophisticated matrix to express more correlations, such as the periodicity in some tasks.
- **Spatial correlation**: We use a spatial correlation matrix $\mathbb{S}$ to express the spatial correlations in one task. In most urban sensing scenarios, the closer cells usually have the similar sensed values. Thus, we use the Euclidean distance (*i.e.*, $\text{distance}(i, j) = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$) to model the spatial correlations, as follow:

$$\mathbb{S}[i, j] = \exp(-\text{distance}(i, j)/\sigma_s^2), \quad \text{if } i \neq j. \tag{7}$$

  Then, we normalize the matrix $\mathbb{S}$ as $\sum_{j=1, j\neq i}^{m} \mathbb{S}[i, j] = 1$ and set $\mathbb{S}[i, i] = -1, \forall i = \{1, \ldots, m\}$.
- **Value correlation**: Besides the temporal and spatial correlations, in the urban sensing scenarios, some cells still would have the similar sensed values, since they have the similar surroundings or some other reasons. We conduct the value correlation matrix $\mathbb{V}$ as follow:

$$\mathbb{V}[i, j] = \exp(-|v_i - v_j|/\sigma_v^2), \quad \text{if } i \neq j, \tag{8}$$

  where v_i is the sensed value of cell i at one sensing cycle. Similar to $\mathbb{S}$, $\mathbb{V}$ is later normalized. Then, we learn and update the matrix $\mathbb{V}$ at each cycle until converge to capture the value correlations, which directly reflect on the sensed values, as follow:

$$\mathbb{V} = \mathbb{V} + \gamma_v(\mathbb{V}' - \mathbb{V}), \tag{9}$$

  where $\mathbb{V}'$ is calculated by Eq. 8 at the current cycle and γ_v is the learning rate.

Note that our compressive sensing consider three typical correlations in urban sensing tasks. As discussed above, other constraint matrices or metrics characterizing correlations can be easily applied in this modified compressive sensing. While we should notice that considering more constraints may improve the inference accuracy but cost more time to converge. In this paper, we have tested the running time of the compressive sensing in Section VI, and it can converge quickly after a few iterations (less than 20 iterations and costs $\sim 0.5s$), which is totally acceptable in practical use.

2) MULTI-TASK COMPRESSIVE SENSING

In modern cities, large urban sensing systems require multiple types of sensing data to form a multi-dimensional urban sensing map, in order to illustrate the comprehensive state of the urban scene. In many cases, these multiple types of sensing data present some inherent correlations. For example, a higher temperature usually leads to a lower humidity [15] and the poor air quality may obtain the higher readings on both PM2.5 and PM10 [27]. Therefore, we propose using the multi-task compressive sensing to exploit both the intra- and inter-task correlations, in order to enhance the

inference performances and simultaneously infer the full maps for multiple tasks.

Inspired by transfer learning [28], we introduce the collective matrix factorization [29] to the singular value decomposition. The basic idea is to share part of hyperparameters between the multiple task, *i.e.*, we share one factor matrix (L or R) among k tasks during the matrices decomposition. Assume that we share L among k tasks, we can obtain the multi-task optimization problem as follow:

$$\min \sum_{t=1}^{k} \big(\lambda_r(\|L\|_F^2 + \|R_t\|_F^2) + \|LR_t^T \circ \mathcal{C} - \mathcal{S}\|_F^2 + \lambda_t \|(LR_t^T)\mathbb{T}^T\|_F^2 + \lambda_s \|\mathbb{S}(LR_t^T)\|_F^2 + \lambda_v \|\mathbb{V}(LR_t^T)\|_F^2\big). \tag{10}$$

Then, we can calculate the inferred data matrices for k tasks simultaneously, denoted as $\hat{D}_L$. Similarly, we can share R and obtain $\hat{D}_R$, and then aggregate them with a weighted ω in Eq. 11.

$$\hat{D} = \omega \hat{D}_L + (1-\omega)\hat{D}_R. \tag{11}$$

Note that the weight ω represents the weighted trade-off between the shared factor matrix L and R. We also conduct some experiments to evaluate the setting of the weight ω in Section VI. In addition, we test the running time of the multi-task compressive sensing in Section VI. The results show that it even costs less time than the total running time of all tasks by single-task compressive sensing, with the help of the inter-task correlations.

B. CELL SELECTION WITH REINFORCEMENT LEARNING

We use the multi-task compressive sensing to infer the full sensing matrices from a few collected data. How can we decide which cells (⟨cell, task⟩ pairs) to sense so as to improve the inference accuracy? Note that we cannot obtain the certain results before we really collect these sensing values, and also the multiple tasks make the problem more difficult. In this paper, we propose the model free reinforcement learning(RL)-based algorithm to deal with the cell selection problem by trials and errors. Also, we design a general network structure with multiple branches for the multi-task scenarios.

1) STATE, ACTION AND REWARD

Before we present the RL-based cell selection algorithm, three key concepts, *i.e.*, state, action and reward, should be formulated first, as shown in Figure 2. Briefly, under a certain state, we can use a Q-table or Neural Network (illustrated in Section IV) to estimate the rewards for all actions. It means that under the current condition, we can do some actions, and each action will give us one reward. We would like to select the action which will give us the largest reward, and it is the best choice for us. Next we model these three concepts for cell selection in details as follow:

(1) **State** describes the current situation, denoted as $\mathbf{s} \in \mathbf{S}$, based on which we can decide our actions. In Sparse MCS,

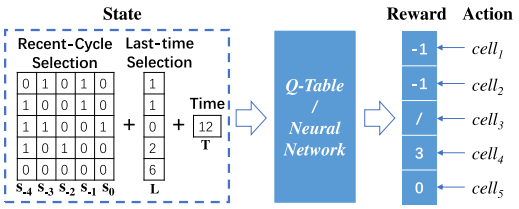


FIGURE 2. State, action and reward in cell selection.

the current situation can be naturally modeled as the cell selection matrix C, since it contains both sensing cycles and sensing cells for the whole sensing process.

However, C would become larger over time. Also the selections long before are of low value for data inference and even may disturb the results. Thus, we hold the recent k cycles of the cell selection matrix as the recent-cycle selection and maintain a last-time selection vector, which records how long one cell has not been selected, to replace the whole cell selection matrix C. Moreover, some necessary information for urban sensing tasks, such as time, should be added into the state.

A simple example of state is shown in Figure 2. Here we keep recent 5 cycles of C as the recent-cycle selection. The last-time selection vector records that how many cycles one cell has not been selected. Note that $cell_5$ has not been selected during the recent 5 cycles, so we record $5+1=6$ cycles for it. Also, the time is set as $\{0, 1, \ldots, 23\}$ to represent the sensing period of 24 hours.[6] In the multi-task scenarios, we design a general network structure with multiple branches, and the state for each task will be fed into one branch respectively, then we concatenate the results to obtain the rewards for all ⟨cell, task⟩ pairs, *i.e.*, the actions introduced below.

(2) **Action** is what we decide to do under each state, denoted as $\mathbf{a} \in \mathbf{A}$. In Sparse MCS, it is actually the next selected cell. Note that we select the cell one by one but not to select a set, in order to avoid the large action space. While RL will add the expected rewards attainable from future states to the current reward, and thus it approximately selects the optimal set of cells after enough training. In addition, for a certain task, we won't select one cell twice at the same cycle. As shown in Figure 2, $cell_3$ has been selected at this cycle and thus it won't be selected again. In the multi-task scenarios, actions can actually be seen as the ⟨cell, task⟩ pairs, *i.e.*, $\langle cell_5, task_1\rangle$.

(3) **Reward** reflects how good an action is under a certain state, denoted as $\mathbf{r}$. In our problem, the reward is the improved inference accuracy by one action, and we formulate the reward as follow:

$$\mathbf{r} = \exp\big(-\text{Error}(D, \hat{D})/\sigma_e^2\big) \tag{12}$$

$$= \exp\Big(-\sum_{i=0}^{k} \|(D_i - \hat{D}_i) \circ (1/D_i)\|_1 \cdot \omega_i/\sigma_e^2\Big), \tag{13}$$

[6]Time is very important in urban sensing, such as the periodicity in traffic data. Thus we add it into state, and it can be set according to the specific tasks.

where $D_i \circ C_i = \hat{D}_i \circ C_i$ and ω_i is the weight of task i. In Eq. 13, the higher accuracy means the higher reward. We use the error rate to deal with the different metrics between multiple tasks, and other metric conversion algorithms could be easily modified. Moreover, as discussed above, RL will add the expected rewards attainable from the future states to the current reward, which can be simply seen as $\mathbf{r} = \mathbf{r} + \mathbf{r}'$, where $\mathbf{r}'$ is the next reward which should be calculated iteratively.

With the modeling of state, action and reward, our cell selection problem can be formulated as a sequential decision making problem, and we propose a RL-based algorithm to solve it, as shown in Figure 2. Specifically, under a certain state, we use Q-table/neural networks to estimate the rewards for all cells (⟨cell, task⟩ pairs) and select $cell_4$ as the next selected cell, since it has the highest reward under the current state.

2) REINFORCEMENT LEARNING-BASED CELL SELECTION

Actually, RL is to learn the mappings between state-action pairs and rewards. Next we introduce how can we estimate the reward for each action under a certain state.

Q-table: The traditional reinforcement learning algorithm, such as Q-learning, uses a table to record all the rewards for the state-action pairs, called Q-table. We denote the Q-table as $Q_{\mathbf{S}\times\mathbf{A}}$ and the rewards are filled into it as $Q[\mathbf{s}, \mathbf{a}]$. Under a certain state $\mathbf{s}$, the algorithm selects the largest $Q[\mathbf{s}, \mathbf{a}]$, $\forall \mathbf{a} \in \mathbf{A}$. Then we perform the selected action $\mathbf{a}$, turn to the next state $\mathbf{s}'$ and update the table according to the following equations:

$$Q[\mathbf{s}, \mathbf{a}] = (1-\alpha)Q[\mathbf{s}, \mathbf{a}] + \alpha\big(\mathbf{r} + \gamma V(\mathbf{s}')\big), \tag{14}$$

$$V(\mathbf{s}') = max_{\mathbf{a}'} Q[\mathbf{s}', \mathbf{a}'], \forall \mathbf{a}' \in \mathbb{A} \tag{15}$$

where $V(\mathbf{s}')$ provides the highest expected reward of the next state $\mathbf{s}'$; $\gamma \in [0, 1]$ is the discount factor indicating the myopic view of the Q-learning regarding the future reward; $\alpha \in (0, 1]$ is the learning rate.

Note that $Q[\mathbf{s}', \mathbf{a}']$ in $V(\mathbf{s}')$ will be updated when $\mathbf{a}'$ has been selected under $\mathbf{s}'$. In fact, the values filled in Q-table will be calculated iteratively. We consider the greedy $V(\mathbf{s}')$ as the future reward, and each $Q[\mathbf{s}, \mathbf{a}]$ holds the expected reward obtained from the current action $\mathbf{a}$ and the future reward $V(\mathbf{s}')$. Thus, the RL-based algorithm can approximate the optimal strategy after enough training.

Neural Network: If the state space is small, the traditional reinforcement learning algorithm can work well. However, in our problem, the state space is very large and the Q-table can hardly deal with it. Suppose that we have 50 cells and only keep the recent 5 cycles as the state for one task, the state space achieves $|\mathbf{S}| = 2^{5\times 50} = 2^{250}$. To overcome this problem, we turn to use Deep Q-Learning, which uses a neural network to replace the table Q. Under a certain state $\mathbf{s}$, we don't need to search the large table Q but use neural network to estimate the $Q[\mathbf{s}, \mathbf{a}]$ directly, as shown in Eq. 16.

$$Q(\mathbf{s}, \mathbf{a}) = \mathbb{E}\big[\mathbf{r} + \gamma max_{\mathbf{a}'} Q(\mathbf{s}', \mathbf{a}')\big]. \tag{16}$$

Specifically, we design a neural network with two dense layers, in order to deal with heterogeneous inputs and catch the comprehensive correlations in our state. The current state s is fed into the neural network and a linear layer outputs the $Q(\mathbf{s}, \mathbf{a})$ for all $\mathbf{a} \in \mathbf{A}$. Our goal is to train this neural network parameterized by θ to approximately achieve $Q_\theta(\mathbf{s}, \mathbf{a}) \approx Q[\mathbf{s}, \mathbf{a}], \forall \mathbf{s} \in \mathbf{S}, \mathbf{a} \in \mathbf{A}$. Therefore, according to Eq. 15 and 15, we use the stochastic gradient algorithm and the loss function is defined as follow:

$$L(\theta_t) = \mathbb{E}_{\langle \mathbf{s}, \mathbf{a}, \mathbf{r}, \mathbf{s}' \rangle}\Big[\big(\mathbf{r} + \gamma max_{\mathbf{a}'} Q_{\theta_t}(\mathbf{s}', \mathbf{a}') - Q_{\theta_t}(\mathbf{s}, \mathbf{a})\big)^2\Big]. \tag{17}$$

Thus

$$\nabla_{\theta_t} L(\theta_t) = \mathbb{E}_{\langle \mathbf{s}, \mathbf{a}, \mathbf{r}, \mathbf{s}' \rangle}\Big[\big(\mathbf{r} + \gamma max_{\mathbf{a}'} Q_{\theta_t}(\mathbf{s}', \mathbf{a}') - Q_{\theta_t}(\mathbf{s}, \mathbf{a})\big)\nabla_{\theta_t} Q_{\theta_t}(\mathbf{s}, \mathbf{a})\Big]. \tag{18}$$

Algorithm 1 Reinforcement Learning-Based Cell Selection

Initialization:

$t = 0$, $\mathbf{D} = \emptyset$, ϵ, REPLACE_ITER, Initialize two neural networks with random weights θ_t and $\theta' = \theta_t$

1: **while** True **do**
2: Get $\mathbf{s}$
3: Calculate $Q(\mathbf{s}, \mathbf{a})$ $\forall \mathbf{a} \in \mathbb{A}$
4: **if** isTrain **then**
5: Select $\mathbf{a}$ with ϵ-greedy algorithm
6: Get $\mathbf{r}$, $\mathbf{s}'$
7: $\mathbf{e}_t = \langle \mathbf{s}, \mathbf{a}, \mathbf{r}, \mathbf{s}' \rangle \rightarrow \mathbf{D}$
8: Randomly select some $\mathbf{e}$ from $\mathbf{D}$
9: Calculate θ_t via Eq. 19
10: t++
11: **if** t%REPLACE_ITER == 0 **then**
12: $\theta' = \theta_t$
13: **end if**
14: **else**
15: Select $\mathbf{a}$ with the largest $Q(\mathbf{s}, \mathbf{a})$
16: **end if**
17: **end while**

The RL-based cell selection algorithm is summarized in Algorithm 1. First, we get the current state $\mathbf{s}$. Then, we use the neural network to calculate the $Q(\mathbf{s}, \mathbf{a})$ for all $\mathbf{a} \in \mathbf{A}$. If neural network has been trained well, we directly select the action $\mathbf{a}$ with the largest $Q(\mathbf{s}, \mathbf{a})$ (line 15). For each sensing cycle, we select a total of B ⟨cell, task⟩ pairs under the budget constraint, or select a minimal subset while satisfying the predefined accuracy requirement. Otherwise, we should use the ϵ-greedy algorithm for each selection (line 5), in which we select the best $\mathbf{a}$ with a probability $1-\epsilon$ and select a random action with the probability ϵ. This algorithm used here is to balance the explore and exploit. Then, we obtain the current reward $\mathbf{r}$ and the next state $\mathbf{s}'$. This experience $\mathbf{e}_t = \langle \mathbf{s}, \mathbf{a}, \mathbf{r}, \mathbf{s}' \rangle$ will be added into the memory pool $\mathbf{D}$. For each training step, we will randomly select some experiences to learn and

update the network parameters θ_t (line 7-8). We also use the fixed Q-targets [30], which holds a target network with the parameters θ' cloned from the primary network but updates θ' periodically (line 12). The target network is used to calculate the reward of the next state $\mathbf{s}'$ as shown in Eq. 19.

$$\nabla_{\theta_t} L(\theta_t) = \mathbb{E}_{\langle \mathbf{s},\mathbf{a},\mathbf{r},\mathbf{s}' \rangle}\Big[\big(\mathbf{r} + \gamma max_{\mathbf{a}'} Q_{\theta'}(\mathbf{s}', \mathbf{a}') - Q_{\theta_t}(\mathbf{s}, \mathbf{a})\big)\nabla_{\theta_t} Q_{\theta_t}(\mathbf{s}, \mathbf{a})\Big] \quad (19)$$

3) NETWORK STRUCTURE FOR MULTIPLE TASKS

In order to better capture the intra- and inter-task correlations in cell selection and select the most effective ⟨cell, task⟩ pairs. We then design a general neural networks with multiple branches to approximate the actions' rewards under a certain state, as shown in Figure 3.

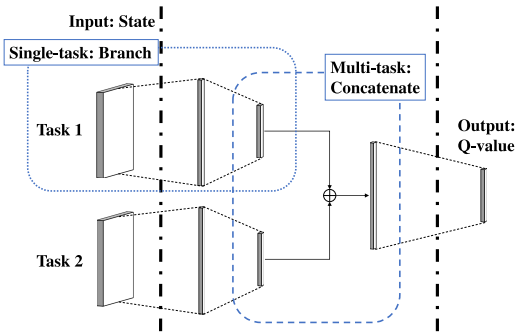


FIGURE 3. The network structure with multiple branches.

In our design, we adopt multiple branches to catch the intra-task correlations first. The goal and process are the same with the single-task scenario, and thus the network structures of the branches are the same with the structures in single-task scenarios. The states of tasks are fed into different branches respectively, which keep different weights since they deal with different tasks and extract intra-task correlations to approximate the reward for each task. Then, we concatenate the results and employ two dense layers to fuse the information and capture the inter-task correlations. Finally, a linear layer outputs the rewards (Q-values) for all actions (⟨cell, task⟩ pairs). The network structure with multiple branches deals with the large action space (⟨cell, task⟩ pairs) and effectively captures intra- and inter-task correlations.

In order to ease the difficulty of training, we first train the network for each task, respectively. Then, we use the weights to initialize the branches and train the whole network for multiple tasks. In this way, the network converges quickly and the new tasks can be easily added by adding new branches and leveraging the fine-tuning techniques to train them. In addition, the various metrics in multiple tasks should be considered. In our model, we simply use the weighted error rate in the reward shaping, and thus we concatenate the outputs of multiple branches directly.

V. TRAINING AND ONLINE FRAMEWORK

In this section, we first introduce the training for the RL-assisted Sparse MCS. According to the observation that a few cells has been selected frequently, we propose to use a small amount of data to train our model. Then, we propose our two-stage online framework with training and running phases and explain how the framework deal with the online training for RL in practical use.

A. TRAINING FOR REINFORCEMENT LEARNING

When we have enough historical data or we can conduct a preliminary study for a long time, we can use these collected data to train our RL-based algorithm in an offline manner. In order to enhance the training data, for each sensing cycle, we randomly select part of values, give a specified order and calculate the error rates. This process will repeat many times and thus we can obtain a large number of experiences, *i.e.*, $\mathbf{e}_t = \langle \mathbf{s}, \mathbf{a}, \mathbf{r}, \mathbf{s}' \rangle$ in Algorithm 1. Then, we periodically add the experiences into the memory pool $\mathbf{D}$, and RL randomly selects some experiences for training. In this way, we can train our model effectively by using enough historical data.

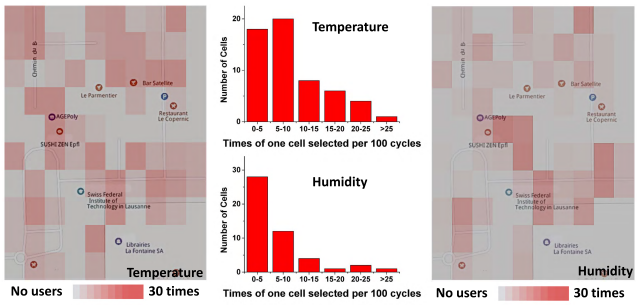


FIGURE 4. The times of one cell selected per the 100 cycles in *Sensor-Scope* dataset (Temperature-Humidity).

However, in many cases, the large amount of training data is hard to obtain and we cannot have such an unlimited historical data set for training. Then, can we use a small amount of data to train our model and achieve a good enough performance? In most urban sensing systems, the answer is yes. As shown in Figure 4, we try out all of the possible ⟨cell, task⟩ pairs and count the times of one cell selected per the 100 cycles in our experiments over *Sensor-Scope* [31] dataset in Section VI. We can see that 30% of cells have been selected more than 10 times while the others are less than 10 times. Moreover, the central cells have been selected more frequently than the corner ones, which fits our analysis and proves the effectiveness of cell selection. Therefore, we consider that a few cells are more important and they can help more on data inference. We don't need too much data to train all of the possible states and actions from all cells. We can use a small amount of data (part of data from 100 sensing cycles in our experiments in Section VI.C) to train our model on the common states and actions from the important cells,

which is more effective and can achieve a good enough performance.

B. ONLINE FRAMEWORK

Based on the observations, we design a two-stage online framework including training and running phase, as shown in Figure 5. The basic idea is to use an online method to select ⟨cell, task⟩ pairs, while at the same time collect the data for RL training. After the RL training is finished, we move to the second stage, *i.e.*, switch to RL-based cell selection method to perform Sparse MCS. Next we introduce the two stages in details.

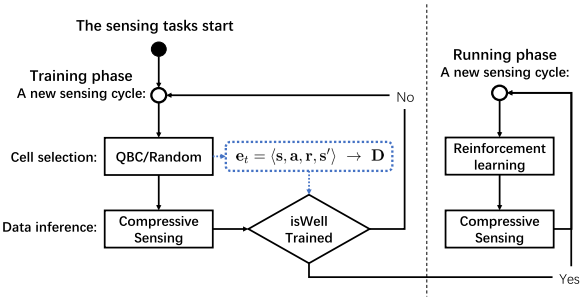


FIGURE 5. The online framework including training and running phase.

In the training phase, we use an online cell selection method, called Query-By-Committee (QBC). The QBC method has been used in [8], [9], [11], which uses various inference algorithms (such as compressive sensing and K-Nearest Neighbors (KNN) [32]) to infer all the values and chooses the ⟨cell, task⟩ pair which has the largest variance. Note that QBC selects the most uncertain ⟨cell, task⟩ pair but not the most effective one, thus some of its selections may not be the right ones. For example, one corner cell may be hard to infer, while the sensed data from it can help less on the inferring for other values. Although the QBC doesn't select the best ⟨cell, task⟩ pairs, it removes the bad ones and collects data from the approximate effective ones, and then we use these data to train the RL model effectively. Additionally, we should notice that in some large-scale urban sensing systems, the QBC may yield a long running time since it needs to run various inference algorithms for many times. We may turn to use the random method instead in our training phase, which randomly selects a ⟨cell, task⟩ pair to sense, and the performances mainly depend on the compressive sensing.

The online method can achieve the acceptable performances and provide the training data for our RL-based method. After sensing and collecting data by QBC for some cycles (100 cycles in our experiments in Section VI.C), the RL model is trained well, *i.e.*, the average error rate obtained by RL is lower than the online method's in several testing cycles (5 cycles in our experiments). We then turn to the running phase and use our RL-based algorithm as the cell selection method. Note that in practice, we do not know all the ground truth data in the runtime, and thus we cannot directly obtain the errors between the ground truth and the inferred data. In this paper, we use the well known k-fold cross-validation [33] to estimate the error from the sensed values. Also, both the data inference and cell selection may cause some errors, thus we use the average error rate and periodically judge if the RL is better. Moreover, in the running phase, we should continue to train the better model for a period of time, and also we should retrain and update the model periodically to adapt the possible new changes.

VI. PERFORMANCE EVALUATION

In this section, we conduct extensive experiments over two real-world datasets, which contains multiple types of urban sensing data, such as temperature-humidity and PM2.5-PM10.

A. DATASETS

We adopt two well-known urban sensing datasets, *Sensor-Scope* [31] and *U-Air* [34]. *Sensor-Scope* contains some environment readings (temperature and humidity) and *U-Air* has the air quality data (PM2.5 and PM10), which are the representative urban sensing datasets. These data were collected by the fixed sensors, while the mobile devices can also be used to obtain them. Thus, we can use them to evaluate our proposed algorithms effectively. The detailed statistics of the two datasets are listed in Table 2 and the descriptions are shown as follow:

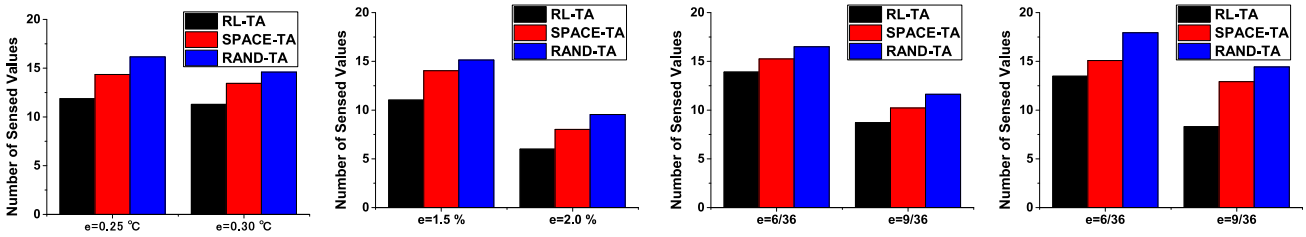
TABLE 2. Statistics of two evaluation datasets.

	Datasets	
	Sensor-Scope	***U-Air***
City	Lausanne (Switzerland)	Beijing (China)
Data	Temperature-Humidity	PM2.5-PM10
Cell size	$50*30m^2$	$1000*1000m^2$
Number of cells	57	36
Cycle length	$0.5h$	$1h$
Duration	$7d$	$11d$
Mean Âś Std.	$6.04 \pm 1.87°C$ (T)	79.11 ± 81.21 (PM2.5)
	$84.52 \pm 6.32\%$ (H)	63.12 ± 48.56 (PM10)

Sensor-Scope [31]: The *Sensor-Scope* dataset contains various environment readings collected in the EPFL campus. We select two representative sensing data, temperature and humidity, to evaluate our methods for multi-dimensional urban sensing. Actually, temperature and humidity present some inherent correlations. For example, a higher temperature usually leads to a lower humidity [15]. Our methods can capture their intra- and inter-task correlations well.

U-Air [34]: The *Sensor-Scope* dataset collects the air quality data, *i.e.*, PM2.5 and PM10, in Beijing. As shown in Table 2, the air quality readings have the large fluctuations and we use the air quality index category [34] instead of the original readings.[7] The PM2.5 and PM10 have the strong inter-task correlations, especially for the bad air quality, both

[7] Six categories: Good (0-50), Moderate (51-100), Unhealthy for Sensitive Groups (101-150), Unhealthy (150-200), Very Unhealthy (201-300), and Hazardous (>300).



(a) Temperature (b) Humidity (c) PM2.5 (d) PM10

FIGURE 6. Number of selected ⟨cell, task⟩ pairs on temperature-humidity and PM2.5-PM10.

of them may get the higher readings [27]. Thus, our method can achieve the better performances.

B. BASELINE ALGORITHMS AND CONFIGURATIONS

We compare our RL-based algorithms to two existing methods: SPACE-TA and RAND-TA [11]. SPACE-TA selects the most uncertain cells determined by "committee" (various data inference algorithms) to sense for multiple tasks in parallel. Then, it uses a compressive sensing algorithm to deduce the full sensing maps. RAND-TA will randomly select cells and infer the rest by using an inference algorithm. Note that RAND-TA actually achieves a competitive performance since the random selection can already provide a lot of information to the powerful inference technologies as compressed sensing.

Furthermore, we study and evaluate the performance of proposed algorithms in data inference and cell selection, respectively. **For data inference**, we use the CS, KNN-S and KNN-T [11] to evaluate the multi-task compressive sensing (Multi-CS) algorithm. Compressive sensing (CS) has been described in detail at Section IV.A and K-Nearest-Neighbors (KNN) is the classic method which infers value by using one's weighted average of k nearest neighbors. Similar with the modified compressive sensing, we also consider the nearest neighbors on Spatial/Temporal dimensions, called KNN-S and KNN-T. The weights for KNN-S are set as the spatial correlation matrix $\mathbb{S}$. The Temporal weights are simply set as the normalized $\exp(-|i-j|)$, where i and j are the numbers of cycles, since we don't conduct the more sophisticated temporal matrix. Besides, we don't consider the value correlations, since we cannot find the nearest values with the one which needs to be inferred.

Since our multi-task RL-based cell selection method (Multi-RL) can help on data inference, all of the compared algorithms use it as the cell selection method, in order to evaluate the performances improved by data inference. We set $\lambda_r = 0.3, \lambda_{s/t/v} = 0.1$ for Eq. 6 and 10, $\sigma_{s/v/e} = 1.5$ for Eq. 7 and 8. We also change the ω to evaluate the weighted Multi-CS.

For cell selection, we use the RL, QBC and Random as the baseline algorithms. RL is the reinforcement learning-based cell selection for single task. QBC uses 4 inference algorithms, including CS, Multi-CS, KNN-S and KNN-T, to infer all the values and choose the cell which has the largest variance. Random will randomly select several cells to sense. Note that the random method actually achieves a competitive performance since the data inference algorithms can work well based on the sparse and enough random selections. Similar with data inference, we use Multi-CS as the inference method for all the cell selection algorithms.

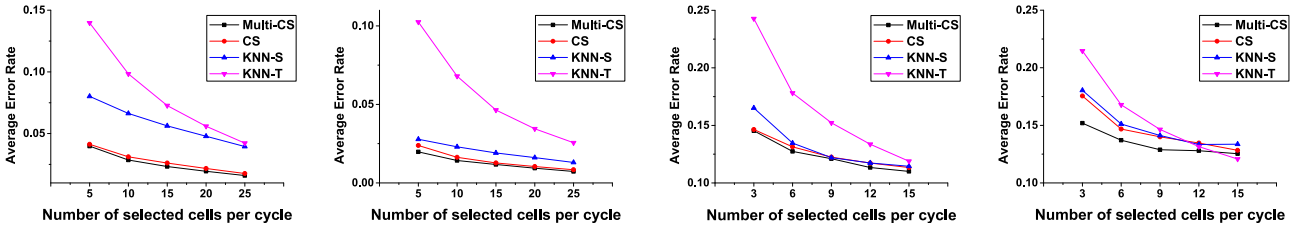
The network structure is shown in Figure 3 in Section IV.B, which has 2 branches and 2 fully connected layers to fuse the information from branches. For *Sensor-Scope*, we keep recent 5 cycles as recent-cycle selection and the time T in state is set as $\{0, 1, \ldots, 47\}$ for *Sensor-Scope*. Thus, the size of input (state) for each branch is $5 \times 57 + 1 \times 57 + 1 = 343$ and the size of output (action) is 57. Similarly, for *U-Air*, we also keep recent 5 cycles, T is set as $\{0, 1, \ldots, 23\}$ and the sizes are 217 and 36. For the other parameters, we set discount factor $\gamma = 0.9$ and learning rate $\alpha = 0.05$ in Eq. 15 and dynamically adjust ϵ from 1 to 0.1 for whole process of training. At the beginning of the training, we set a relatively large ϵ so that we can try more; then, with the training process proceeds, we gradually reduce ϵ until the model is converged. We also set all of the weights/hyper-parameters $\omega = 1$ to avoid confusion.

C. EXPERIMENT RESULTS

1) RL-ASSISTED SPARSE MCS

We first evaluate the performance of our RL-assisted Sparse MCS algorithms, which jointly addresses data inference and cell selection. The existing method, SPACE-TA, focuses on the task allocation problem with accuracy constraint, which aims to select a minimal subset of ⟨cell, task⟩ pairs while satisfying the accuracy requirement. Thus, we first evaluate the average number of sensed values per sensing cycle. The results are shown in Figure 6.

For the temperature-humidity in *Sensor-Scope*, we set the accuracy requirements e to $0.25/0.30°C$ and 1.5/2.0%. Thus, for each cycle, we should select users one by one until the expected inference errors (estimated by k-fold cross-validation [33]) are smaller than $0.25/0.30°C$ and 1.5/2.0%. The average numbers of selected ⟨cell, task⟩ pairs per cycle have been shown in Figure 6 (a) and (b). Our RL-assisted algorithms can sense 17.3/16.0% and 21.4/25.0% fewer ⟨cell, task⟩ pairs than SPACE-TA for the temperature and humidity tasks, respectively. For the PM2.5-PM10 in *U-Air*, we obtain the similar tendency in Figure 6 (c) and (d) under the accuracy



(a) Temperature (b) Humidity (c) PM2.5 (d) PM10

FIGURE 7. Data inference on temperature-humidity and PM2.5-PM10.

requirements 6/36 and 9/36. These results show that our RL-assisted algorithms can effectively reduce the sensed values of Sparse MCS. Next, we will further study and evaluate the performances of the algorithms in data inference and cell selection, respectively.

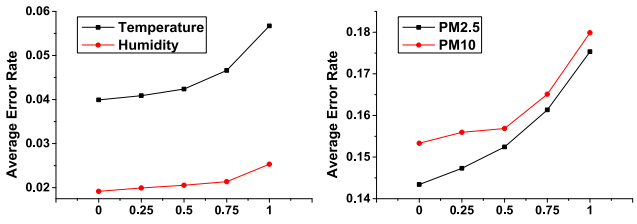
2) DATA INFERENCE

We evaluate the data inference on two datasets and each with two sensing tasks. For all the compared algorithms, we use RL-based cell selection algorithm to select the next sensed cells. In order to reflect the improvement of inference accuracy in data inference and cell selection respectively, we change the number of selected cells for each sensing cycles and evaluate the average error rates for each methods. The results are shown in Figure 7.

The average error rates over four urban sensing tasks have the similar tendencies and our Multi-CS achieves the best performance. Along with the increase of the number of selected cells, the average error rates become lower, since the more selected cells provide more information to help the data inference. The CS and Multi-CS are relatively closer and achieve better performances when we have less selected cells. The reason is that the compressive sensing can deal with the sparse data well. However, when we have sensed many cells in a cycle, the compressive sensing may lose to the traditional algorithms, as shown in Figure 7(d).

The Multi-CS can utilize the inter-task correlations and perform better than CS, especially when we have less selected cells. Note that we want to give the comparisons between all the tasks with different metrics, thus we display the error rates with small values. These small values, especially on the baseline algorithms, can also prove the effectiveness of our RL-based cell selection. In fact, the Multi-CS and CS can achieve the $0.238/0.249°C$ error in Temperature and $1.67/2.02\%$ in Humidity when we only sense 3 out of 57 cells by RL for each tasks, which are good enough in most cases. Similarly, the Multi-CS and CS get the average errors of 0.29/0.30 and 0.30/0.35 on the air quality index.

Moreover, we change the ω in Eq. 11, in order to show how the weight ω impacts the inference accuracy for Multi-CS. The results are shown in Figure 8. We can see that when $\omega = 0$, all the tasks achieve the best performances. It means that the factor L shared in multi-tasks plays an important role and obtains the smallest inference error.



(a) Temperature-Humidity (b) PM2.5-PM10

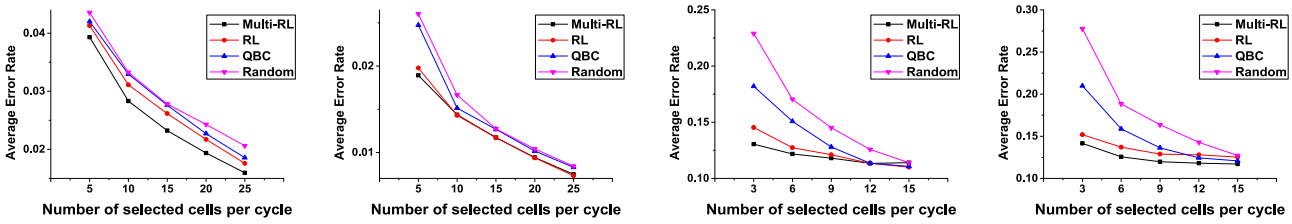
FIGURE 8. ω on temperature-humidity and PM2.5-PM10.

3) CELL SELECTION

Then, we evaluate the cell selection over two multi-dimensional urban sensing datasets, Temperature-Humidity and PM2.5-PM10, as shown in Figure 9. Similar with data inference, we use Multi-CS for all compared algorithms. We change the number of selected cells for each sensing cycles and evaluate the average error rates for each methods. Note that our Multi-RL selects the ⟨cell, task⟩ pairs for multiple tasks, which means that we may select more cells for the task which will help more.

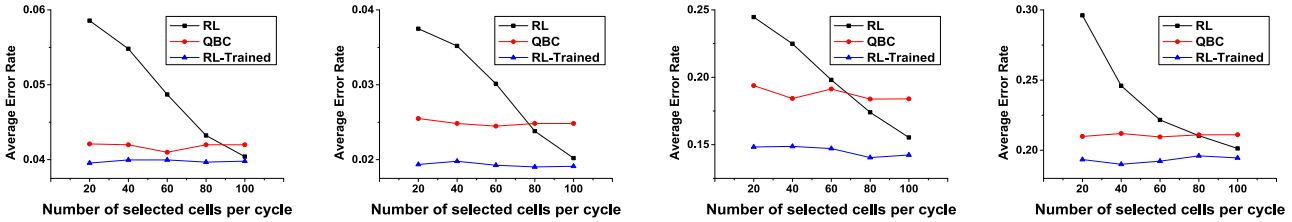
Along with the increase of the number of selected cells, the results have the similar tendencies with data inference, since the more selected ⟨cell, task⟩ pairs may provide more information to improve the accuracies. Our Multi-RL always achieves the best performance, especially when we only select less ⟨cell, task⟩ pairs. For Temperature-Humidity, since we set the weight $\omega = 1$, the Multi-RL performs a better performance on Temperature while keeps an acceptable performance on Humidity. Actually, we select 7.52 cells for Temperature and 2.48 cells for Humidity on average. The reason is that the reward in Multi-RL encourages the algorithm to select more cells for Temperature, since it has the higher error rate. Similarly, we select 3.11 cells for PM10 and 2.89 for PM2.5, since their error rates are close, which also have been reflected in Figure 6.

Similar with the data inference, we display the average error rates, in order to show the comparisons between the tasks with different metrics. Thus, the results are relatively small. Moreover, we use RL as the cell selection methods for all compared algorithms to evaluate the data inference, and use compressive sensing to evaluate cell selection. We jointly address the data inference and cell selection and our proposed methods always achieve the best performances, which proves the effectiveness of our methods.



(a) Temperature (b) Humidity (c) PM2.5 (d) PM10

FIGURE 9. Cell selection on temperature-humidity and PM2.5-PM10.



(a) Temperature (b) Humidity (c) PM2.5 (d) PM10

FIGURE 10. Online training on temperature-humidity and PM2.5-PM10.

4) ONLINE FRAMEWORK

As shown in Figure 4 of Section V.A, a few cells may be important in the urban sensing scenarios, and these cells will be selected more frequently. We consider that the training on these important cells would be effective and thus we propose the online framework with training and running phases. Here, we test our online framework and show the changes of error rates during the training and running phases and illustrate the transition point of our online framework for Temperature-Humidity and PM2.5-PM10, as shown in Figure 5.

Note that both data inference and cell selection may cause some errors, thus we use the average error rate and periodically judge if the RL-based algorithm is better. We judge if the RL can perform better in the next 5 cycles than Random and QBC every 20 cycles. At the 100th cycles, we turn to the running stage for Temperature-Humidity and PM2.5-PM10, and their performances are closed to the RL models which have been trained offline. We can see that when we use only 20 cycles for training, the error rates are very high. The reason is that our RL has learned that a few cells are good, and always select them, which makes the data inference so difficult. In addition, since our datasets are limited as shown in Table 2, we only use the first 100 cycles for training and the rest are used for testing. In our future work, we would like to conduct our experiments on some larger datasets, which may achieve better performances.

5) RUNNING TIME

Finally, we test the average running time for each method, as shown in Table 3. Our experiment platform is equipped with Intel Xeon CPU E2630 v4 @ 2.20GHz and 32 GB RAM. The Muti-CS costs 0.91s and 0.66s to infer the full sensing data for Temperature-Humidity and PM2.5-PM10, which are totally acceptable in real-life deployments. For cell selection, we implement our RL-based algorithms in TensorFlow (CPU version) and they obtain the great advantages than QBC on the running times. RL-based algorithms only need less than 1ms during the running stage. The training can be conducted offline/online, which costs around $10-30$ minutes to converge, since we design a simple but effective network structure with multiple branches.

TABLE 3. Runtime for each method.

	Temperature	Humidity	PM2.5	PM10
CS	0.49s	0.50s	0.35s	0.37s
Multi-CS	0.91s		0.66s	
QBC	1.04s	1.18s	0.91s	0.94s
RL	0.033ms	0.034ms	0.031ms	0.031ms
Multi-RL	0.60ms		0.40ms	

VII. CONCLUSION

In this paper, we investigate the multi-dimensional urban sensing issue in Sparse Mobile CrowdSensing, which allows us to sense only a few cells while inferring the data of the rest of city areas with high data quality. First, we exploit intra- and inter-task correlations and use the multi-task compressive sensing to infer the full map of the sensing data with only a few collected values. Then, we present the reinforcement learning-based algorithm for multiple tasks to select the ⟨cell, task⟩ pairs which would perform best on data inference. Finally, we present a two-stage online framework including training and running phase. Extensive evaluations on two real-world data sets have verified the effectiveness of our proposed algorithms on multi-dimensional urban sensing, which achieve better performances than the state-of-the-art

mechanisms. In the future work, we would like to introduce the user mobility [6] and data uploading [35], [36] into a more practical Sparse MCS. Also, the privacy protection mechanism [10] would also be explored and added into our reinforcement learning-assisted solutions.

REFERENCES

[1] R. K. Ganti, F. Ye, and H. Lei, "Mobile crowdsensing: Current state and future challenges," *IEEE Commun. Mag.*, vol. 49, no. 11, pp. 32–39, Nov. 2011.
[2] D. Zhang, L. Wang, H. Xiong, and B. Guo, "4 W 1 H in mobile crowd sensing," *IEEE Commun. Mag.*, vol. 52, no. 8, pp. 42–48, Aug. 2014.
[3] C. Zhu, H. Zhou, V. C. M. Leung, K. Wang, Y. Zhang, and L. T. Yang, "Toward big data in green city," *IEEE Commun. Mag.*, vol. 55, no. 11, pp. 14–18, Nov. 2017.
[4] H. Aly, A. Basalamah, and M. Youssef, "Automatic rich map semantics identification through smartphone-based crowd-sensing," *IEEE Trans. Mobile Comput.*, vol. 16, no. 10, pp. 2712–2725, Oct. 2017.
[5] Z. Liu, S. Jiang, P. Zhou, and M. Li, "A participatory urban traffic monitoring system: The power of bus riders," *IEEE Trans. Intell. Transp. Syst.*, vol. 18, no. 10, pp. 2851–2864, Oct. 2017.
[6] W. Liu, Y. Yang, E. Wang, Z. Han, and X. Wang, "Prediction based user selection in time-sensitive mobile crowdsensing," in *Proc. 14th Annu. IEEE Int. Conf. Sens., Commun., Netw. (SECON)*, Jun. 2017, pp. 1–9.
[7] E. Wang, Y. Yang, J. Wu, W. Liu, and X. Wang, "An efficient prediction-based user recruitment for mobile crowdsensing," *IEEE Trans. Mobile Comput.*, vol. 17, no. 1, pp. 16–28, Jan. 2018.
[8] L. Wang, D. Zhang, A. Pathak, C. Chen, H. Xiong, D. Yang, and Y. Wang, "CCS-TA: Quality-guaranteed online task allocation in compressive crowdsensing," in *Proc. ACM Int. Joint Conf. Pervasive Ubiquitous Comput.*, Sep. 2015, pp. 683–694.
[9] L. Wang, D. Zhang, Y. Wang, C. Chen, X. Han, and A. M'hamed, "Sparse mobile crowdsensing: Challenges and opportunities," *IEEE Commun. Mag.*, vol. 54, no. 7, pp. 161–167, Jul. 2016.
[10] L. Wang, D. Zhang, D. Yang, B. Y. Lim, and X. Ma, "Differential location privacy for sparse mobile crowdsensing," in *Proc. IEEE 16th Int. Conf. Data Mining (ICDM)*, Dec. 2016, pp. 1257–1262.
[11] L. Wang, D. Zhang, D. Yang, A. Pathak, C. Chen, X. Han, H. Xiong, and Y. Wang, "SPACE-TA: Cost-effective task allocation exploiting intradata and interdata correlations in sparse crowdsensing," *Acm Trans. Intell. Syst. Technol.*, vol. 9, no. 2, Jan. 2017, Art. no. 20.
[12] L. Wang, W. Liu, D. Zhang, Y. Wang, E. Wang, and Y. Yang, "Cell selection with deep reinforcement learning in sparse mobile crowdsensing," in *IEEE 38th Int. Conf. Distrib. Comput. Syst. (ICDCS)*, Jul. 2018, pp. 1543–1546.
[13] Y. Zheng, L. Capra, O. Wolfson, and H. Yang, "Urban computing: Concepts, methodologies, and applications," *ACM Trans. Intell. Syst. Technol.*, vol. 5, no. 3, p. 38, 2014.
[14] C. Zhu, V. C. M. Leung, J. J. P. C. Rodrigues, L. Shu, L. Wang, and H. Zhou, "Social sensor cloud: Framework, greenness, issues, and outlook," *IEEE Netw.*, vol. 32, no. 5, pp. 100–105, Sep. 2018.
[15] R. G. Allen *et al.*, "Crop evapotranspiration—Guidelines for computing crop water requirements—FAO Irrigation and drainage paper 56," *Fao, Rome*, vol. 300, no. 9, p. D05109, 1998.
[16] R. K. Rana, C. T. Chou, S. S. Kanhere, N. Bulusu, and W. Hu, "Ear-phone: An end-to-end participatory urban noise mapping system," in *Proc. 9th ACM/IEEE Int. Conf. Inf. Process. Sensor Netw.*, Apr. 2010, pp. 105–116.
[17] Y. Zhu, Z. Li, H. Zhu, M. Li, and Q. Zhang, "A compressive sensing approach to urban traffic estimation with probe vehicles," *IEEE Trans. Mobile Comput.*, vol. 12, no. 11, pp. 2289–2302, Nov. 2013.
[18] S. He and K. G. Shin, "Steering crowdsourced signal map construction via Bayesian compressive sensing," in *Proc. IEEE INFOCOM IEEE Conf. Comput. Commun.*, Apr. 2018, pp. 1016–1024.
[19] R. S. Sutton, F. Bach, and A. G. Barto, *Reinforcement Learning—An Introduction (Adaptive Computation and Machine Learning series)*. Cambridge, MA, USA: MIT Press, 2005.
[20] T. Liu, Y. Zhu, Y. Yang, and F. Ye, "Incentive design for air pollution monitoring based on compressive crowdsensing," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2016, pp. 1–6.
[21] V. Mnih *et al.*, "Playing Atari with deep reinforcement learning," 2013, *arXiv:1312.5602*. [Online]. Available: https://arxiv.org/abs/1312.5602
[22] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. V. D. Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, and M. Lanctot,, "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, p. 484, 2016.
[23] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, and T. A. Lillicrap, "A general reinforcement learning algorithm that masters chess, shogi, and go through self-play," *Science*, vol. 362, no. 6419, pp. 1140–1144, Dec. 2018.
[24] L. Xiao, T. Chen, C. Xie, H. Dai, and H. V. Poor, "Mobile crowdsensing games in vehicular networks," *IEEE Trans. Veh. Technol.*, vol. 67, no. 2, pp. 1535–1545, Feb. 2017.
[25] L. Xiao, Y. Li, G. Han, H. Dai, and H. V. Poor, "A secure mobile crowdsensing game with deep reinforcement learning," *IEEE Trans. Inf. Forensics Security*, vol. 13, no. 1, pp. 35–47, Jan. 2018.
[26] M. Roughan, Y. Zhang, W. Willinger, and L. Qiu, "Spatio-temporal compressive sensing and Internet traffic matrices," *IEEE/ACM Trans. Netw. (ToN)*, vol. 20, no. 3, pp. 662–676, Jun. 2012.
[27] R. Kumar and A. E. Joseph, "Air pollution concentrations of $PM_{2.5}$, PM_{10} and NO_2 at ambient and kerbsite and their correlation in metro city—Mumbai," *Environ. Monitor. Assessment*, vol. 119, nos. 1–3, pp. 191–199, Aug. 2006.
[28] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Trans. Knowl. Data Eng.*, vol. 22, no. 10, pp. 1345–1359, Oct. 2010.
[29] A. P. Singh and G. J. Gordon, "Relational learning via collective matrix factorization," in *Proc. 14th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2008, pp. 650–658.
[30] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, and G. Ostrovski, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, p. 529, 2015.
[31] F. Ingelrest, G. Barrenetxea, G. Schaefer, M. Vetterli, O. Couach, and M. Parlange, "SensorScope: Application-specific sensor network for environmental monitoring," *Acm Trans. Sensor Netw.*, vol. 6, no. 2, Feb. 2010, Art. no. 17.
[32] T. Cover and P. Hart, "Nearest neighbor pattern classification," *IEEE Trans. Inf. Theory*, vol. IT-13, no. 1, pp. 21–27, Jan. 1967.
[33] R. Kohavi, "A study of cross-validation and bootstrap for accuracy estimation and model selection," in *Proc. Int. Joint Conf. AI*, Aug. 1995, vol. 14. no. 2, pp. 1137–1145.
[34] Y. Zheng, F. Liu, and H.-P. Hsieh, "U-Air: When urban air quality inference meets big data," in *Proc. 19th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2013, pp. 1436–1444.
[35] H. Zhou, H. Wang, X. Chen, X. Li, and S. Xu, "Data offloading techniques through vehicular ad hoc networks: A survey," *IEEE Access*, vol. 6, pp. 65250–65259, 2018.
[36] C.-M. Huang, Y.-F. Chen, S. Xu, and H. Zhou, "The vehicular social network (VSN)-based sharing of downloaded GEO data using the credit-based clustering scheme," *IEEE Access*, vol. 6, pp. 58254–58271, 2018.

WENBIN LIU received the B.S. degree in physics from Jilin University, Changchun, China, in 2012, where he is currently pursuing the Ph.D. degree with the College of Computer Science and Technology. He is a Visiting Ph.D. Student with the Wireless Networks and Multimedia Services Department, Telecom SudParis/Institut Mines-Telecom, Evry, France. His research interests include mobile crowdsensing and ubiquitous computing.

YONGJIAN YANG received the B.E. degree in automatization from the Jilin University of Technology, Changchun, Jilin, China, in 1983, the M.E. degree in computer communication from the Beijing University of Post and Telecommunications, Beijing, China, in 1991, and the Ph.D. degree in software and theory of computer from Jilin University, Changchun, in 2005. He is currently a Professor and a Ph.D. Supervisor with Jilin University, the Vice Dean of the Software College, Jilin University, the Director of Key Laboratory of the Ministry of Information Industry, the Standing Director of the Communication Academy, and a member of the Computer Science Academy of Jilin Province. His research interests include network intelligence management, wireless mobile communication and services, and wireless mobile communication.

EN WANG received the B.E. degree in software engineering, the M.E. degree in computer science and technology, and the Ph.D. degree in computer science and technology from Jilin University, Changchun, in 2011, 2013, and 2016, respectively, where he is currently an Associate Professor with the Department of Computer Science and Technology. He is also a Visiting Scholar with the Department of Computer and Information Sciences, Temple University, Philadelphia. His current research interests include the efficient utilization of network resources, scheduling, and drop strategy in terms of buffer-management, energy-efficient communication between human-carried devices, and mobile crowdsensing.

LEYE WANG received the Ph.D. degree in computer science from the Institut Télécom SudParis and University Paris 6, France, in 2016. He was a Postdoctoral Researcher with The Hong Kong University of Science and Technology. He is currently an Assistant Professor with the Key Laboratory of High Confidence Software Technologies, Peking University, China. His research interests include ubiquitous computing, mobile crowdsensing, and urban computing.

DJAMAL ZEGHLACHE received the Ph.D. degree in electrical engineering from SMU, Dallas, TX, USA, in 1987. He joined Cleveland State University as an Assistant Professor, in 1987. From 1990 to 1991, he worked with the NASA Lewis Research Centre on mobile satellite terminals, systems, and applications. In 1992, he joined the Networks and Services Department, Télécom SudParis, Institut Telecom, where he is currently a Professor and the Head of the Wireless Networks and Multimedia Services Department. He is also the Dean of Research of Télécom SudParis. He has coauthored around 100 publications in ranked international conferences and journals. His research interest includes a broad spectrum related to fixed and wireless networks and services. His current focus is on network architectures, protocols, and interfaces to ensure smooth evolution towards loosely coupled future Internet, cloud networking, and cloud architectures. He is currently addressing interdomain cooperation and federation challenges for these networks, related modeling for resource optimization of wireless networks (5G vision), of infrastructures and platforms offered as a service to users and providers. He was an Editor of the IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS.

DAQING ZHANG received the Ph.D. degree from the University of Rome La Sapienza, Italy, in 1996. He is currently a Professor with Télécom SudParis, France. His research interests include context-aware computing, urban computing, and mobile computing. He has served as the General Chair or the Program Chair for more than ten international conferences. He is also an Associate Editor of the ACM Transactions on Intelligent Systems and Technology, the IEEE TRANSACTIONS ON BIG DATA, and so on.