Reinforcement learning-based cell selection in sparse mobile crowdsensing

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

To cite this version:

Wenbin Liu, Leye Wang, En Wang, Yongjian Yang, Djamal Zeghlache, et al.. Reinforcement learning-based cell selection in sparse mobile crowdsensing. Computer Networks, Elsevier, 2019, 161, pp.102-114. 10.1016/j.comnet.2019.06.010 . hal-02321018

HAL Id: hal-02321018

https://hal.archives-ouvertes.fr/hal-02321018

Submitted on 23 Oct 2019

Computer Networks 161 (2019) 102–114

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet

Reinforcement learning-based cell selection in sparse mobile crowdsensing

Wenbin Liu[a,d,1], Leye Wang[b,c,1], En Wang[a,*], Yongjian Yang[a], Djamal Zeghlache[d], Daqing Zhang[b,c,d]

[a] College of Computer Science and Technology, Jilin University, Changchun, Jilin, China
[b] Key Lab of High Confidence Software Technologies, Peking University, Beijing, China
[c] School of Electronic Engineering and Computer Science, Peking University, Beijing, China
[d] RS2M, Telecom SudParis, Evry, France

ARTICLE INFO

Article history:
Received 27 March 2019
Revised 28 May 2019
Accepted 11 June 2019
Available online 12 June 2019

Keywords:
Mobile crowdsensing
Cell selection
Reinforcement learning
Compressive sensing

ABSTRACT

Sparse Mobile Crowdsensing (MCS) is a novel MCS paradigm which allows us to use the mobile devices to collect sensing data from only a small subset of cells (sub-areas) in the target sensing area while intelligently inferring the data of other cells with quality guarantee. Since selecting sensed data from different cell sets will probably lead to diverse levels of inference data quality, *cell selection* (i.e., choosing which cells in the target area to collect sensed data from participants) is a critical issue that will impact the total amount of data that requires to be collected (i.e., data collection costs) for ensuring a certain level of data quality. To address this issue, this paper proposes the reinforcement learning-based cell selection algorithm for Sparse MCS. First, we model the key concepts in reinforcement learning including state, action, and reward, and then propose a Q-learning based cell selection algorithm. To deal with the large state space, we employ the deep Q-network to learn the Q-function that can help decide which cell is a better choice under a certain state during cell selection. Then, we modify the Q-network to a deep recurrent Q-network with LSTM to catch the temporal patterns and handle partial observability. Furthermore, we leverage the transfer learning techniques to relieve the dependency on a large amount of training data. Experiments on various real-life sensing datasets verify the effectiveness of our proposed algorithms over the state-of-the-art mechanisms in Sparse MCS by reducing up to 20% of sensed cells with the same data inference quality guarantee.

1. Introduction

Mobile crowdsensing (MCS) [3] is a novel sensing mechanism, which allows us to use the ubiquitous mobile devices to address various urban monitoring needs in environment and traffic monitoring [29]. While the traditional MCS applications usually recruit many participants in order to cover all the *cells* (i.e., sub-areas) of the target area to ensure sensing quality. This costs a lot and may even be impossible (e.g., there is no participant in some cells) [18,19,28]. To deal with these problems, a new MCS paradigm, namely *Sparse MCS*, is proposed recently [21,23], which collects data from only a subset of cells while intelligently inferring the data of other cells with quality guarantee (i.e., the error of inferred data is lower than a threshold).

In Sparse MCS, one key issue is *cell selection — which cells the organizer needs to choose and collect sensed data from participants* [21]. To show the importance of cell selection, Fig. 1 (left part) gives an illustrative example of two different cell selection cases in a city, which is split into 4 × 4 cells. In Case 1.1, all the selected cells are gathered in one corner of the city; in Case 1.2, the collected data is evenly distributed in the whole city. As the data of most sensing tasks has spatial correlations (i.e., nearby cells may have similar data), e.g., air quality [30], the cell selection of Case 1.2 will generate a higher inference quality of the inferred data than Case 1.1. Moreover, a MCS campaign usually lasts for a long time (i.e., sensing every one hour), so that not only spatial correlations, but also temporal correlations need to be carefully considered in cell selection. As shown in Fig. 1 (right part), sensing the same cells in continuous cycles (Case 2.1) may not be as efficient as sensing the different cells (Case 2.2) considering the inference

[*] Corresponding author.
E-mail addresses: liuwb16@mails.jlu.edu.cn (W. Liu), leyewang@sei.pku.edu.cn (L. Wang), wangen@jlu.edu.cn (E. Wang), yyj@jlu.edu.cn (Y. Yang), djamal.zeghlache@telecom-sudparis.eu (D. Zeghlache), daqing.zhang@telecom-sudparis.eu (D. Zhang).
[1] Wenbin Liu and Leye Wang contributed equally.

https://doi.org/10.1016/j.comnet.2019.06.010
1389-1286/© 2019 Published by Elsevier B.V.

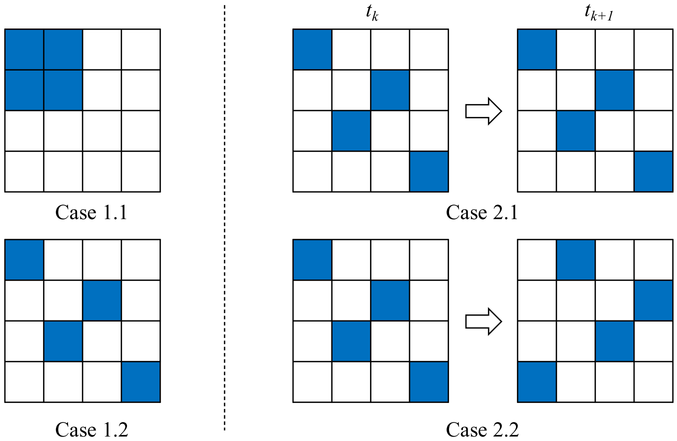


Fig. 1. Different cell selection cases.

quality. Therefore, the data of different MCS applications may involve diverse spatio-temporal correlations, which is hard to model and determine, so the proper cell selection strategy is a non-trivial task.

Existing works on Sparse MCS mainly leverage *Query-By-Committee* (QBC) [20,23] in cell selection. QBC first uses various inference algorithms to deduce the data of all the unsensed cells, and then chooses the cell where the inferred data of various algorithms has the largest variance as the next cell for sensing. Briefly, QBC chooses the most uncertain cell considering a committee of inference algorithms, which deals with the cell selection skilfully and has shown its effectiveness as a whole [20,23]. However, QBC only chooses the cell which is the most uncertain at that moment, and ignores whether the current selection would help the inferring in the future or not. For example, as shown in Fig. 1 (right part), if we select one cell at time t_k, it would help the inferring not only for this moment but also for the subsequent instant t_{k+1}.

To overcome these limitations, in this paper, we study the critical cell selection problem in Sparse MCS, with reinforcement learning, which can capture the spatio-temporal correlations in the sensing data and approximate the global optimal strategy for cell selection. In recent years, reinforcement learning has shown its successes in decision making problems in diverse areas such as robot control and game playing [11,16], which can be abstracted as 'an *agent* needs to decide the *action* under a certain *state*, in order to maximize some notions of cumulative *reward*'. Reinforcement learning tries out different actions, observes the rewards and thus learns the optimal decisions for each state. Our cell selection problem can be actually interpreted as '*an MCS server* (agent) needs to *choose the next cell for sensing* (action) considering the *data already collected* (state), in order to *minimize the number of sensed cells under a quality guarantee* (reward)'. In this regard, it is appropriate to apply reinforcement learning on the cell selection problem in Sparse MCS.

By using reinforcement learning, the cell selection problem in Sparse MCS can be well solved. First of all, a model-free reinforcement learning method can record which cell would help most under a certain state through trial and error. In fact, trial and error is exactly the fundamental idea of reinforcement learning. After sufficient training, reinforcement learning would record all the rewards for each state-action pair and select the action which has the biggest reward under the state. Moreover, reinforcement learning adds the reward attainable from the future state to the reward in its current state, effectively influencing the current selection by the potential reward in the future. Thus, reinforcement learning can approximate the global optimal strategy for cell selection in Sparse MCS.

To effectively employ reinforcement learning in cell selection, we face several issues. *(1) How to mathematically model the state, action, and reward*, which are key concepts in reinforcement learning [17]. Briefly speaking, reinforcement learning attempts to learn a *Q-function* which takes the current *state* as input, and generates *reward* scores for each possible *action* as output. Then, we can take the action with the highest reward score as our decision. *(2) How to learn the Q-function.* Traditional Q-learning techniques in reinforcement learning use tables to store rewards for each state-action pair. It works well in the scenarios where the number of states and actions is limited. However, in Sparse MCS, the number of states is actually quite large. We propose to use a neural network to replace the table, i.e., leveraging deep reinforcement learning to learn Q-function for our cell selection problem. *(3) The training data scarcity issue.* Usually, deep reinforcement learning requires a lot of training data to learn Q-function. However, in MCS, we could only obtain a small amount of data for training. To deal with this problem, we propose to collect a small amount of redundant data to conduct the effective training by random combination. Moreover, we try to introduce the transfer learning technique, in order to make use of the well trained Q-function and reduce the required training data for heterogeneous sensing tasks in similar target area.

In summary, this work has the following contributions:

(1) To the best of our knowledge, this work is the first research that attempts to leverage the reinforcement learning to address the critical cell selection issue in Sparse MCS. We believe that using reinforcement learning would be a promising way to solve such kind of decision making problems, especially when we cannot obtain a direct solution and the decisions have long-term utilities.

(2) We propose the reinforcement learning-based algorithms for cell selection in Sparse MCS. First, we model the *state, action*, and *reward* and propose a tabular Q-learning based algorithm, which records the reward scores for each state-action pair in tables. Considering the extremely large state space, we employ a neural network instead of tables and learn a Q-function to calculate the reward scores. Since the neural network cannot catch the temporal patterns and handle partial observability well, we propose a recurrent deep neural network structure, which uses a Long-Short-Term-Memory layer instead of the dense layer. Finally, we collect a small amount of redundant data to conduct the effective training by random combination and propose a transfer learning method between heterogeneous sensing tasks, in order to relieve the dependence on a large amount of training data.

(3) Experiments with applications in temperature, humidity, air quality and traffic monitoring have verified the effectiveness of our proposed algorithms. In particular, our proposed algorithms can outperform the state-of-the-art mechanism QBC by selecting up to 20% fewer cells while guaranteeing the same quality in Sparse MCS.

The remainder of the paper is organized as follows. Firstly, we review related works in Section 2. The problem formulation are introduced in Section 3. In Section 4, we propose the reinforcement learning-based cell selection algorithms and discuss the training and transfer learning method. Then, the performances of the proposed algorithms are evaluated through extensive simulations over three real world datasets in Section 5. Finally, we conclude this paper in Section 6.

2. Related works

2.1. Sparse mobile crowdsensing

MCS is proposed to utilize widespread crowds to perform large-scale sensing tasks [3,29]. Existing works in MCS mainly recruit many participants to ensure sensing quality[18,19,28], which costs a lot and may even be impossible. To minimize sensing cost while ensuring data quality, some MCS tasks involve inference algorithms to fill missing data of unsensed cells, such as noise sensing [12], traffic monitoring [31], and air quality sensing [20]. It is worth noting that in such MCS tasks, compressive sensing has become the *de facto* choice of the inference algorithm [12,20,23,27,31]. Recently, by extracting the common research issues involved in such tasks involving data inference, Wang et al. [21] proposed a new MCS paradigm, called *Sparse MCS*. Besides the inference algorithm, Sparse MCS also abstracts other critical research issues such as *cell selection* and *quality assessment*. Later, privacy protection mechanism was also added into Sparse MCS [22]. In this paper, we focus on the *cell selection* and aim to use deep reinforcement learning techniques to address it.

2.2. Reinforcement learning

Reinforcement Learning (RL) [17] is concerned with how to map states to actions so as to maximize the cumulative rewards. It utilizes rewards to guide the agent to do the better sequential decisions, and has substantive and fruitful interactions with other engineering and scientific disciplines. Recently, many researchers focus on combining deep learning with reinforcement learning to enhance RL in order to solve concrete problems in the sciences, business, and other areas. Mnih et al. [10] proposed the first deep reinforcement learning model (DQN) to deal with the high-dimensional sensory input successfully and apply it to play seven Atari 2600 games. More recently, Silver et al. [15] applied DQN and present *AlphaGo*, which was the first program to defeat world-class players in Go. Moreover, to deal with the partially observable states, Hausknecht and Stone [6] introduced a deep recurrent neural network (DRQN), and applied it to play Atari 2600 games. Lample and Chaplot [9] even used DRQN to play FPS Games.

Although the reinforcement learning has already been used in a variety of areas, like object recognition, robot control, and communication protocol [17], MCS researchers just began to apply it very recently. Xiao et al. [24] formulated the interactions between a server and vehicles as a vehicular crowdsensing game. Then they proposed the Q-learning based strategies to help server and vehicles make the optimal decisions for the dynamic game. Moreover, Xiao et al. [25] applied Deep Q-Network to derive the optimal policy for the Stackelberg game between a MCS server and the smartphone users. As far as we know, this paper is the first research attempts to use reinforcement learning in cell selection of sparse MCS, so as to reduce the recruited participants while still guaranteeing the data quality.

3. System model and problem formulation

First, we define several key concepts, and introduce the compressive sensing for data inference and Bayesian inference for quality assessment briefly. Then we mathematically formulate the cell selection problem in Sparse MCS. Finally, a running example is illustrated to explain our problem in more details.

3.1. Definitions

Definition 1. Sensing Area. We suppose that the target sensing area can be split into a set of cells (e.g., 1 km × 1 km grids [23,30]). The objective of a sensing task is to get a certain type of data (e.g., temperature, air quality) of all the cells in the target area.

Definition 2. Sensing Cycle. We suppose the sensing tasks can be split into equal-length cycles, and the cycle length is determined by the MCS organizers according to their requirements [23,26]. For example, if an organizer wants to update the data of the target sensing area every one hour, then he can set the cycle length to one hour.

Definition 3. Ground Truth Data Matrix. Suppose we have m cells and n cycles, then for a certain sensing task, the ground truth data matrix is denoted by $\mathcal{D}_{m\times n}$, where $\mathcal{D}[i, j]$ is the true data in cell i at cycle j.

Definition 4. Cell Selection Matrix. In Sparse MCS, we will only select partial cells in each cycle for data collection, while inferring the data for the rest cells. Cell selection matrix, denoted as $\mathcal{C}_{m\times n}$, marks the cell selection results. $\mathcal{C}[i, j] = 1$ means that the cell i is selected at cycle j for data collection; otherwise, $\mathcal{C}[i, j] = 0$.

Definition 5. Collected Data Matrix. A collected sensing data matrix $\mathcal{S}_{m\times n}$ records the actual collected data: $\mathcal{S}_{m\times n} = \mathcal{D} \circ \mathcal{C}$, where $\circ$ denotes the element-wise product of two matrices.

Definition 6. Inferred Data Matrix. In Sparse MCS, when an organizer decides not to collect any more data in the current cycle, the data of unsensed cells will then be inferred. Then, we denote the inferred data of the kth cycle as $\hat{\mathcal{D}}[:, k]$, and thus the inferred data of all the cycles as a matrix $\hat{\mathcal{D}}_{m\times n}$. Note that in Sparse MCS, compressive sensing is the *de facto* choice of the data inference algorithm nowadays [12,20,23,27,31], and we also use it in this work.

Definition 7. (e, p)-quality [23]. In Sparse MCS, the quality guarantee is called *(e, p)-quality*, meaning that in $p \cdot 100\%$ of cycles, the inference error (e.g., mean absolute error) is not larger than e. Formally,

$$|\{k|error(\mathcal{D}[:, k], \hat{\mathcal{D}}[:, k]) \le e, 1 \le k \le n\}| \ge n \cdot p, \tag{1}$$

where n is the number of total sensing cycles.

Note that in practice, since we do not know the ground truth data matrix $\mathcal{D}$, we also cannot know whether $error(\mathcal{D}[:, k], \hat{\mathcal{D}}[:, k])$ is smaller than e in the current cycle with 100% confidence. This is why we include p in the quality requirement, as it is impossible to ensure 100% of cycles' error less than e. To ensure (e, p)-quality, certain quality assessment method is needed in Sparse MCS to estimate the probability of the error less than e for the current cycle. If the estimated probability is larger than p, then the current cycle satisfies (e, p)-quality and no more data will be collected (we will then move to the next sensing cycle). In Sparse MCS, leave-one-out based Bayesian inference method is often leveraged for quality assessment [20,21,23], and we also use it in this work.

3.2. Data inference

Compressive sensing is the *de facto* choice to infer the full sensing matrix from the partially collected sensing values and has shown its effectiveness in some scenarios [20,23]. It reconstructs the full sensing matrix $\hat{\mathcal{D}}$ based on the low-rank property:

$$\min rank(\hat{\mathcal{D}}) \tag{2}$$

$$\text{s.t.}, \hat{\mathcal{D}} \circ \mathcal{C} = \mathcal{S}, \tag{3}$$

where $\hat{\mathcal{D}} \circ \mathcal{C}$ is the element-wise product of the inferred full sensing matrix and cell selection matrix, and $\mathcal{S}$ is the collected data matrix.

With the help of the *singular value decomposition, i.e.*, $\hat{\mathcal{D}} = LR^T$, we convert the above optimization problem as follows [31]:

$$\min \lambda(\|L\|_F^2 + \|R\|_F^2) + \|LR^T \circ \mathcal{C} - \mathcal{S}\|_F^2. \tag{4}$$

Moreover, in order to better capture the spatio-temporal correlations in the sensing data, we further add the explicit spatiotemporal correlations into compressive sensing [8,13], and the optimization function is denoted by Eq. (5):

$$\begin{aligned}\min\ & \lambda_r(\|L\|_F^2 + \|R\|_F^2) + \|LR^T \circ \mathcal{C} - \mathcal{S}\|_F^2 \\ & + \lambda_s\|\mathbb{S}(LR^T)\|_F^2 + \lambda_t\|(LR^T)\mathbb{T}^T\|_F^2,\end{aligned} \tag{5}$$

where $\mathbb{S}$ and $\mathbb{T}$ are spatial and temporal constraint matrices, while λ_r, λ_s, and λ_t are chosen to balance the weights of different elements in the optimization problem. Then we use an alternating least squares [8,13] procedure to estimate L and R iteratively, in order to get the optimal $\hat{\mathcal{D}}$ ($\hat{\mathcal{D}} = LR^T$).

3.3. Quality assessment

In this paper, the leave-one-out based Bayesian inference is used to assess the inference quality. First, we use the leave-one-out resampling to obtain the set of inferred-true data pairs. Then, comparing the inferred data to the true collected data, the Bayesian inference is leveraged to assess whether the current data quality can satisfy the predefined (e, p)-quality requirement or not.

The basic idea of leave-one-out resampling is simple but effective. Consider that we collect sensing data from m' out of all the m cells and thus we have m' observations. For each time, we leave one observation out and infer it based on the rest $m' - 1$ observations by using compressive sensing. After running this process for all m' observations, we obtain m' inferred-true data pairs.

Based on the m' inferred-true data pairs, we can use Bayesian inference to estimate the probability distribution of the inference error $\mathcal{E}$ in all the m cells, which can help quality assessment. Actually, satisfying the (e, p)-quality can be seen as $P(\mathcal{E} \le e) \ge p$. We regard $\mathcal{E}$ as an unknown parameter and update the probability distribution of $\mathcal{E}$ based on our observation θ (m' inferred-true data pairs). Therefore, we can approximate $P(\mathcal{E} \le e)$:

$$P(\mathcal{E} \le e) \approx \int_{-\infty}^{e} g(\mathcal{E}|\theta) d\mathcal{E}, \tag{6}$$

where $g(\mathcal{E}|\theta)$ is the estimated probability distribution of $\mathcal{E}$. For two widely used error metrics, mean absolute error (for continuous value) and classification error (for classification label), calculating the $g(\mathcal{E}|\theta)$ based on the observation can be seen as the classic Bayesian statistics problem: *inferring normal mean with unknown variance* and *Coin Flipping*, and then we can calculate the $g(\mathcal{E}|\theta)$ by t-distribution [1] and *Beta* distribution [4], respectively.

3.4. Problem formulation

Based on the previous definitions and the brief introduction on compressive sensing and Bayesian inference used in this paper, we define our research problem and focus on the cell selection.

Problem [Cell Selection]: Given a Sparse MCS task with m cells and n cycles, using compressive sensing as data inference method and leave-one-out based Bayesian inference as quality assessment method, we aim to select a minimal subset of sensing cells during the whole sensing process (minimize the number of non-zero entries in the *cell-selection matrix* $\mathcal{C}$), while satisfying (e, p)-quality:

$$\min \sum_{i=1}^{m} \sum_{j=1}^{n} \mathcal{C}[i,j]$$

s.t., satisfy $(e, p)-$quality

We now use a running example to illustrate our problem in more details, as shown in Fig. 2. (1) Consider that the MCS task only have 5 cells and it is currently in the 5th cycle; (2) By using the cell selection algorithms, we select cell 3 to collect the sensing data, and then use the compressive sensing and Bayesian inference to assess whether the selected cells in this cycle can satisfy (e, p)-quality; (3) Since the current cycle cannot satisfy the quality requirement, we continue to select cell 5 for collecting data; (4) The quality requirement is now satisfied, so the data collection is terminated for the current cycle, and the data of the unsensed cells is inferred by compressive sensing. In this example, we totally obtain 11 data submissions for these 5 cycles and our objective is exactly to minimize the number of data submissions while ensuring the quality. In addition, we should notice that maybe some cells cannot be sensed at the current sensing cycle (e.g., there are no users in these cells). In practical use, we first update a candidate cell set, in which cells can be sensed in the current sensing cycle, and then select the next cell to sense from the candidate cell set.

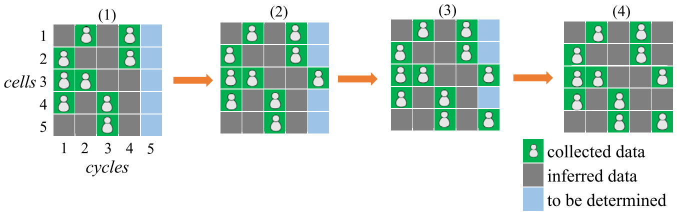


Fig. 2. Running example.

4. Methodology

In this section, we propose the reinforcement learning-based algorithms to address the cell selection problem in Sparse MCS. First, we will mathematically model the state, reward, and action. Then, with a simplified MCS task example (i.e., there are only a few cells in the target area), we explain how traditional reinforcement learning find the most appropriate cell for sensing based on our state, action, and reward modeling. Afterward, we elaborate how deep learning can be combined with reinforcement learning to work on more realistic cases of cell selection where the target area can include a large number of cells. Finally, we describe the training stage and explain how we can collect a small amount of redundant data to conduct the effective training by random combination. Moreover, we introduce transfer learning technique to help us generate a cell selection strategy with only a little training data under some specific conditions.

4.1. Modeling state, action, and reward

To apply reinforcement learning on cell selection, we first model the key concepts in terms of state, action, and reward, as shown in Fig. 3. Specifically, under the *state* (consists of the current data collection and some additional information), we should learn a *Q-function* (will be elaborated in the next few subsections), which calculate the *reward* score for each *action* (choosing which cell to collect the sensing data). If an action gets a higher reward score, it may be a better choice. Next we formally model the three concepts.

(1) **State** represents the current data collection condition. In Sparse MCS, *cell selection matrix* (Definition 4) can naturally model the state well, as it records both where and when we have collected data from the target sensing area during the whole task.

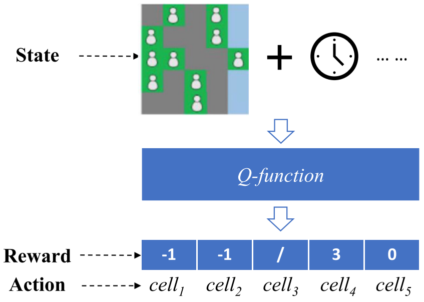
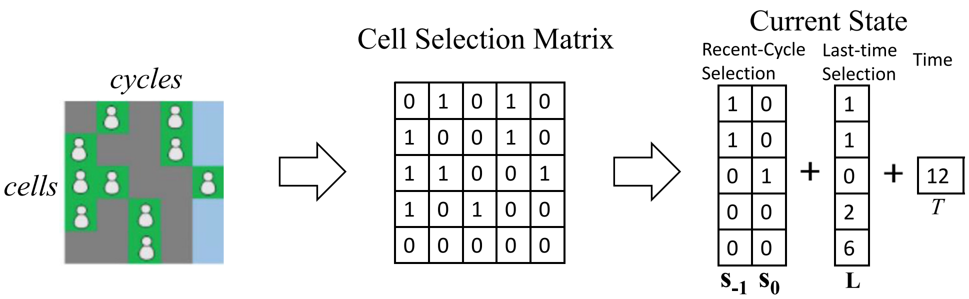


Fig. 3. State, action, reward in cell selection.



$$\begin{array}{|c|c|c|c|c|} \hline 0 & 1 & 0 & 1 & 0 \\ \hline 1 & 0 & 0 & 1 & 0 \\ \hline 1 & 1 & 0 & 0 & 1 \\ \hline 1 & 0 & 1 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 \\ \hline \end{array} \Rightarrow \begin{array}{|c|c|} \hline 1 & 0 \\ \hline 1 & 0 \\ \hline 0 & 1 \\ \hline 0 & 0 \\ \hline 0 & 0 \\ \hline \end{array} + \begin{array}{|c|} \hline 1 \\ \hline 1 \\ \hline 0 \\ \hline 2 \\ \hline 6 \\ \hline \end{array} + \boxed{12}$$

Fig. 4. An example of state model.

In this paper, we keep the recent k cycles' cell selection matrix and the last-time selection vector instead of the cell selection matrix, called the recent-cycle selection and the last-time selection, denoted as $[\mathbf{s}_{-k+1},\ldots,\mathbf{s}_{-1},\mathbf{s}_0,\mathbf{L}]$. $\mathbf{s}_0$ represents the cell selection vector of the current cycle (1 means selected and 0 means no), $\mathbf{s}_{-1}$ represents last cycle, and so on; $\mathbf{L}$ records how long the cells have not been selected. The recent-cycle selection only keeps the recent selections, which avoids that the previous selections of low value for data inference disturb the results. The last-time selection has gathered more previous selections without missing too much information. In addition, we should also add some necessary information into the state, e.g., the time T for the strong temporal correlations existing in many sensing tasks such as traffic monitoring.

Fig. 4 shows an example of how we encode the current data collection condition into the state model. In this example, the recent two cycles, a total of five cycles' last-time selection, and time are considered, and the state can be denoted as $\mathbf{S}=[\mathbf{s}_{-1},\mathbf{s}_0,\mathbf{L},T]$. Note that the value for cell 5 in last-time selection is 6, which means that the last-time selection for cell 5 is out of range (a total of five cycles) and we set it as $5+1=6$. And the time T is set according to the specific scene. For example, the data is collected every one hour and the time T can be set as $\{0,1,\ldots,23\}$, in order to capture the strong temporal correlations.

In addition, we use $\mathbb{S}$ to denote the whole set of states. As an easy example, suppose that we only consider the recent-cycle selection (two cycles) and ignore the last-time selection and time. There are totally five cells in the target area, then the number of possible states, i.e., $|\mathbb{S}|=2^{2\times5}=1024$, which is such a large state space.

(2) **Action** means all the possible decisions that we may make in cell selection. Suppose there are totally m cells in the target sensing area, then our next selected cell can have m choices, leading to the whole action set $\mathbb{A}=\{1,2,\cdots,m\}$. In practice, we will not select one cell for more than once in one cycle, to make the action set consistent under different states, we assume that the possible action set is always the complete set of all the cells under any states. More specifically, if some cells have already been selected in the current cycle, then the probability of choosing these cells is zero.

Note that we select cells one after another, since the multiple cell selections at a time may lead to the large action space. Also the reinforcement learning algorithms consider the potential rewards in the future. After sufficient training, the one-by-one selection would achieve the largest total reward, which is the same goal of the multiple selections at a time.

(3) **Reward** is used to indicate how good an action is. In each sensing cycle, we select actions one by one until the selected cells can satisfy the quality requirement in the current cycle (i.e., inference error less than eSatisfying this quality requirement is the goal of cell selection and should be reflected in the reward modeling. Hence, a positive reward, denoted by R, would be given to an action under a state $\mathbf{S}$ if the quality requirement is satisfied in the current cycle after the action is taken. In addition, as selecting participants to collect data incurs cost, we also put a negative score $-c$ in the reward modeling of an action. Then, the reward can be written as $\mathbf{R}=q\cdot R-c$, where $q\in\{0,1\}$ means whether the action makes the current cycle satisfy the inference quality requirement.

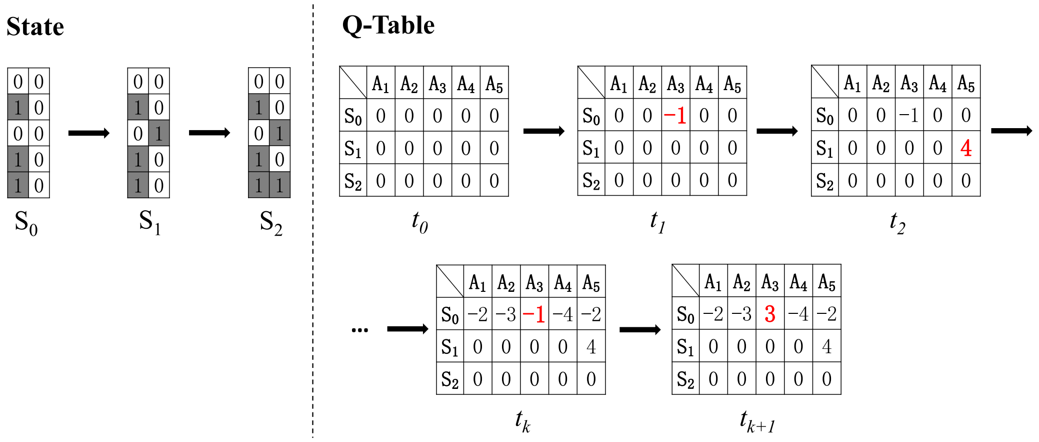
While this reward is actually the immediate utility for one state-action pair. Considering that the current selection would help the inferring in the future, we should add the reward attainable from the future state to the reward in its current state, which can be simply denoted as $\mathbf{R}=\mathbf{R}+\mathbf{R}'$. $\mathbf{R}'$ represents the next reward, which will be calculated iteratively. Suppose that we have n cycles and select m cells for each cycle in average to satisfy the quality requirement, then we obtain the final reward as $\mathbf{R}=n(R-m\cdot c)$. The different actions under a certain state would face the same n, R, and c, while the action which will incur smaller m will achieve a larger reward. Thus, our reward mechanism would guide the agent to minimize the number of selected cells while ensuring the data quality. We would like to set a positive reward to accelerate convergence, i.e., set $R\geq m\cdot c$. While the values of R and c would not influence the performance after the Q-function has been well trained, since the difference value between rewards only depends on the number of selected cells.

With the above modeling, we then need to learn the *Q-function* (see Fig. 3) which can output the reward score of every possible action under a certain state. In the next subsection, we will first use a traditional reinforcement learning method, tabular Q-learning, to illustrate a simplified case where a small number of cells exist in the target sensing area.

4.2. Training Q-function with tabular Q-Learning

In traditional reinforcement learning, the tabular Q-learning has been widely used to obtain the Q-function. In this method, we can use a Q-table to represent the Q-function. The Q-table, denoted as $\mathrm{Q}_{|\mathbb{S}|\times|\mathbb{A}|}$, records the reward score for each possible action $\mathbf{A}\in\mathbb{A}$ under the state $\mathbf{S}\in\mathbb{S}$. The objective of learning the Q-function is then equivalent to filling all the elements in the Q-table, or called Q-value.

The tabular Q-learning based cell selection algorithm is shown in Algorithm 1. Under the current state $\mathbf{S}$, the algorithm first updates the candidate action set $\mathbb{A}_c$, in which cells can be sensed by users in the current sensing cycle and have not been selected. Then, it checks the Q-table and selects the action who has the largest value from $Q[\mathbf{S},\mathbf{A}]$, $\forall\ \mathbf{A}\in\mathbb{A}_c$ (in fact, not always the best action is selected, will be elaborated later). After the action has been conducted, *i.e.* the cell has been selected and the data of the cell has been collected, the current state will change to the next state $\mathbf{S}'$. Note that if the current cycle satisfies the quality requirement (i.e., inference error less than e), the next state will shift to a new cycle. For the selected action, we would get the real reward $\mathbf{R}$ (immediate utility) considering whether the inference quality re-



S0	
0	0
1	0
0	0
1	0
1	0

S1	
0	0
1	0
0	1
1	0
1	0

S2	
0	0
1	0
0	1
1	0
1	1

t_0

	A1	A2	A3	A4	A5
S0	0	0	0	0	0
S1	0	0	0	0	0
S2	0	0	0	0	0

t_1

	A1	A2	A3	A4	A5
S0	0	0	−1	0	0
S1	0	0	0	0	0
S2	0	0	0	0	0

t_2

	A1	A2	A3	A4	A5
S0	0	0	−1	0	0
S1	0	0	0	0	4
S2	0	0	0	0	0

t_k

	A1	A2	A3	A4	A5
S0	−2	−3	−1	−4	−2
S1	0	0	0	0	4
S2	0	0	0	0	0

t_{k+1}

	A1	A2	A3	A4	A5
S0	−2	−3	3	−4	−2
S1	0	0	0	0	4
S2	0	0	0	0	0

Fig. 5. An illustrative example of tabular Q-learning.

Algorithm 1 Tabular Q-learning based cell selection.

```
Initialization:
  Q-table: $Q[\mathbf{S},\mathbf{A}]=0,\ \forall\mathbf{S}\in\mathbb{S},\ \forall\mathbf{A}\in\mathbb{A}$, $\mathbf{L}$, $T$, $\mathbb{A}_c$
while True do
    $\mathbf{S}=[\mathbf{s}_{-k},\ldots,\mathbf{s}_{-1},\mathbf{s}_0,\mathbf{L},T]$
    Update $\mathbb{A}_c$, in which cells can be sensed in the current sensing cycle and have not been selected.
    Check Q-table, select and perform $\mathbf{A}$ from $\mathbb{A}_c$, which has the largest Q-value via the $\epsilon$-greedy algorithm.
    if Satisfy the $(e,p)$-quality then
        // Next cycle
        $\mathbf{s}_1=\mathbf{0}_{m\times 1}$, $\mathbf{S}'=[\mathbf{s}_{-k+1},\ldots,\mathbf{s}_0,\mathbf{s}_1,\mathbf{L}',T']$
        $\mathbf{R}=R-c$
    else
        $\mathbf{s}_0'=\mathbf{s}_0+[0,\cdots,0,1,0,\cdots,0]^T$ (1 is in the $\mathbf{A}$th element), $\mathbf{S}'=[\mathbf{s}_{-k},\ldots,\mathbf{s}_{-1},\mathbf{s}_0',\mathbf{L}',T]$
        $\mathbf{R}=-c$
    end if
    Update Q-table via (7) and (8).
end while
```

quirement of the current cycle is satisfied. Then we should add the possible reward that we might get in the future iteratively and update Q-table according to the equations as follows

$$Q[\mathbf{S},\mathbf{A}]=(1-\alpha)Q[\mathbf{S},\mathbf{A}]+\alpha\big(\mathbf{R}+\gamma V(\mathbf{S}')\big), \tag{7}$$

$$V(\mathbf{S}')=\max_{\mathbf{A}'}Q[\mathbf{S}',\mathbf{A}'],\forall\mathbf{A}'\in\mathbb{A} \tag{8}$$

where $V(\mathbf{S}')$ provides the highest expected reward score 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 if we always select the action with the largest reward score in the Q-table, the algorithm may get a local optima. To address this issue, we need to *explore* during training, i.e., sometimes trying actions other than the best one. We thus use the $\epsilon-$greedy algorithm before selection. More specifically, under a certain state, we select the best action according to the Q-table with a probability $1-\epsilon$ and randomly select one of the other actions with the probability ϵ. Following the existing literature, 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 Q-table is converged and then Algorithm 1 is terminated.

Fig. 5 illustrates an example of our proposed tabular Q-learning based cell selection algorithm. For simplicity, the discount factor γ and the learning rate α are set to 1, and we only consider two recent cycles (i.e., the last and current one) as the states in this example. We suppose that there are five cells in the target area, and hence the state $\mathbf{S}$ has the dimension of $2^{5\times 2}$, as shown in $\mathbf{S}_0$, $\mathbf{S}_1$, and $\mathbf{S}_2$. The value 1 means that the cell has been selected and 0 means not. First, we initialize the table, i.e., all the values in the Q-table are set to 0. When we first meet some states, e.g., $\mathbf{S}_0$, scores of all the actions in the Q-table under $\mathbf{S}_0$ are 0 (Q-table: t_0 in Fig. 5). We then randomly select one action since all the values are equal. If we choose the action $\mathbf{A}_3$ (select the cell 3), the state turns to $\mathbf{S}_1$. Then we update Q[$\mathbf{S}_0$, $\mathbf{A}_3$] as the current score plus the maximum score of the next state $\mathbf{S}_1$ (i.e., future reward). The current reward is $-c$ since the current cycle cannot satisfy the quality requirement ($c=1$ in the example). The maximum score for the state $\mathbf{S}_1$ is 0 in the Q-table. Hence, we get $Q[\mathbf{S}_0,\mathbf{A}_3]=-1+0=-1$ (Q-table: t_1 in Fig. 5). Similarly, under $\mathbf{S}_1$, we choose $\mathbf{A}_5$. If these selections could satisfy the quality, we get the current reward is $R-c=4$ (R is set to 5, i.e., total number of cells). Also, the maximum possible reward of the next state $\mathbf{S}_2$ is 0 in the current Q-table. Then we update $Q[\mathbf{S}_1,\mathbf{A}_5]=5-1+0=4$ (Q-table: t_2 in Fig. 5). After some rounds, we have met $\mathbf{S}_0$ many times and maybe selecting other actions under $\mathbf{S}_0$ is not good choice. And the Q-table would be changed to Q-table: t_k in Fig. 5. This time, under $\mathbf{S}_0$, we check Q-table and find that $\mathbf{A}_3$ has the largest value, so we choose and perform $\mathbf{A}_3$. Then, we update $Q[\mathbf{S}_0,\mathbf{A}_3]=-1+4=3$, since the maximum reward score of the next state $\mathbf{S}_1$ is 4 (Q-table: t_{k+1} in Fig. 5). Therefore, at the next times when we meet $\mathbf{S}_0$ again, we would probably choose the action $\mathbf{A}_3$, since it has the largest reward score, which means that under the state $\mathbf{S}_0$, the action $\mathbf{A}_3$ would give us the most return.

The tabular Q-learning based algorithm can work well for an MCS task in a target area including a small number of cells, as shown in the above example, while the practical MCS applications usually contain a large number of cells. Suppose there are 50 cells in the target area and we only consider recent 2 cycles to model states, then the state space will become extremely huge, $|\mathbb{S}|=2^{2\times 50}=2^{100}$, which is intractable in practice. Moreover, if we add the last-time selection and some necessary information to give a more comprehensive representation of the state, the state space will be even larger, known as the “curse of dimensionality”. To overcome this difficulty, in the next subsection we will propose to leverage deep learning with reinforcement learning to train the decision function for cell selection in Sparse MCS.

4.3. Training Q-function with deep reinforcement learning

4.3.1. Deep Q-Network

To overcome the problem incurred by the extremely large state space in the cell selection, we then turn to use the Deep Q-Network (DQN), which combines Q-learning with neural networks. The difference between DQN and tabular Q-learning is that a neural network is used to replace the Q-table to deal with the dimension curse. In DQN, we do not need the Q-table lookups, but calculate Q[**S, A**] for each state-action pair selection. More specifically, the DQN inputs the current state and action, then it uses a neural network to obtain an estimated value of Q[**S, A**], shown as

$$Q(\mathbf{S},\mathbf{A}) = \mathbb{E}\Big[\mathbf{R} + \gamma \max_{\mathbf{A}'} Q(\mathbf{S}',\mathbf{A}')\Big] \tag{9}$$

In DQN, how to design the network structure impacts the effectiveness of the learned Q-function. One common way is using dense layers to connect the input (*state*) and output (*a reward score vector of all possible actions*). Actually, the network structure with dense layers is appropriate for cell selection. It can handle heterogeneous inputs (consists of the recent two cycles, the last-time selection, and time) and catch the comprehensive correlations in our state. Thus, we use a neural network parameterized by θ to calculate the Q-function, which consists of two fully connected layers.[2] The state is fed into the fully connected layer and a linear layer outputs the Q-values for all possible actions.

Algorithm 2 DQN/DRQN-based cell selection.

```
Initialization:
    $t = 0$, $\mathbf{D} = \emptyset$, $\mathbf{L}$, $T$, $\mathbb{A}_c$
    Initialize DQN/DRQN with random weights $\theta$
while True do
    $\mathbf{S} = [\mathbf{s}_{-k}, \ldots, \mathbf{s}_{-1}, \mathbf{s}_0, \mathbf{L}, T]$
    Update $\mathbb{A}_c$, in which cells can be sensed in the current sensing cycle and have not been selected.
    Calculate Q-value by DQN/DRQN with $\theta_t$ via (9), select $\mathbf{A}$ from $\mathbb{A}_c$ with $\epsilon$-greedy algorithm.
    if Satisfy $(e, p)$-quality then
        // Next cycle
        $\mathbf{s}_1 = \mathbf{0}_{m\times 1}$, $\mathbf{S}' = [\mathbf{s}_{-k+1}, \ldots, \mathbf{s}_0, \mathbf{s}_1, \mathbf{L}', T']$
        $\mathbf{R} = R - c$
    else
        $\mathbf{s}'_0 = \mathbf{s}_0 + [0, \cdots, 0, 1, 0, \cdots, 0]^T$ (1 is in the $\mathbf{A}$th element), $\mathbf{S}' = [\mathbf{s}_{-k}, \ldots, \mathbf{s}_{-1}, \mathbf{s}'_0, \mathbf{L}', T]$
        $\mathbf{R} = -c$
    end if
    $\mathbf{e}_t = \langle \mathbf{S}, \mathbf{A}, \mathbf{R}, \mathbf{S}' \rangle \rightarrow \mathbf{D}$
    Randomly select some $\mathbf{e}$ from $\mathbf{D}$
    Calculate $\theta_t$ via (12)/(14)
    $t$++
    if $t$%RPLACE_ITER == 0 then
        $\theta' = \theta_t$
    end if
end while
```

The DQN-based cell selection algorithm, i.e., *D-Cell*, is summarized in Algorithm 2. Same as Q-learning, we first update the current state **S**. The state **S** is fed into the neural network and obtain the Q-values. Then, we update the candidate action set $\mathbb{A}_c$ and select the action from $\mathbb{A}_c$ with $\epsilon-$greedy algorithm, which is also used in Q-learning to balance the exploration and exploitation. To obtain the estimation of the Q-value which approximates the expected one in (9), our proposed DQN-based algorithm uses the *experience replay* technique [11]. After one selection, we obtain the experience at current time step t, denoted as $\mathbf{e}_t = \langle \mathbf{S}, \mathbf{A}, \mathbf{R}, \mathbf{S}' \rangle$, and the memory pool is $\mathbf{D} = \{\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_t\}$. Then, the algorithm randomly chooses part of the experiences to learn and update the network parameters θ. The goal is to calculate the best θ to obtain $Q_\theta \approx Q$. The stochastic gradient algorithm is applied with the learning rate α and the loss function is defined as follow,

$$L(\theta_t) = \mathbb{E}_{\langle \mathbf{S},\mathbf{A},\mathbf{R},\mathbf{S}'\rangle}\Big[\big(\mathbf{R} + \gamma \max\nolimits_{\mathbf{A}'} Q_{\theta_t}(\mathbf{S}',\mathbf{A}') - Q_{\theta_t}(\mathbf{S},\mathbf{A})\big)^2\Big] \tag{10}$$

Thus

$$\nabla_{\theta_t} L(\theta_t) = \mathbb{E}_{\langle \mathbf{S},\mathbf{A},\mathbf{R},\mathbf{S}'\rangle}\Big[\big(\mathbf{R} + \gamma \max_{\mathbf{A}'} Q_{\theta_t}(\mathbf{S}',\mathbf{A}') - Q_{\theta_t}(\mathbf{S},\mathbf{A})\big)\nabla_{\theta_t} Q_{\theta_t}(\mathbf{S},\mathbf{A})\Big] \tag{11}$$

For each update, D-Cell randomly chooses part of experiences from **D**, then calculates and updates the network parameters θ. Moreover, to avoid the oscillations (i.e., the Q-function changes quite rapidly in training), we apply the *fixed Q-targets technique* [11]. More specifically, we do not always use the latest network parameter θ_t to calculate the maximum possible reward of the next state (i.e., $\max_{\mathbf{A}'} Q_{\theta_t}(\mathbf{S}',\mathbf{A}')$), but update the corresponding parameter θ' every a few iterations, i.e.,

$$\nabla_{\theta_t} L(\theta_t) = \mathbb{E}_{\langle \mathbf{S},\mathbf{A},\mathbf{R},\mathbf{S}'\rangle}\Big[\big(\mathbf{R} + \gamma \max_{\mathbf{A}'} Q_{\theta'}(\mathbf{S}',\mathbf{A}') - Q_{\theta_t}(\mathbf{S},\mathbf{A})\big)\nabla_{\theta_t} Q_{\theta_t}(\mathbf{S},\mathbf{A})\Big] \tag{12}$$

4.3.2. Deep recurrent Q-Network

In DQN-based cell selection, we use a neural network with two dense layers to catch the correlations in our state. However, the temporal correlations also exist in our states, but the DQN only focus the single state and thus cannot catch the temporal pattern well. Moreover, the real-world tasks often feature incomplete and noisy state information resulting from partial observability, leading to a decline in the DQN's performance. We thus propose to use LSTM (Long-Short-Term-Memory) layers instead of dense layers in DQN so as to catch the temporal patterns in our states and handle partial observability, which is also called *Deep Recurrent Q-Network* (DRQN) [6]. More specifically, in DRQN-based cell selection, Q-function can be defined as,

$$Q_{\theta_t}(\mathbf{S}, \mathbf{H}_{t-1}, \mathbf{A}) \tag{13}$$

where $\mathbf{H}_{t-1}$ is the extra input returned by the LSTM network from the previous time step $t-1$. Same as D-Cell, the loss function is defined as follow,

$$\nabla_{\theta_t} L(\theta_t) = \mathbb{E}_{\langle \mathbf{S},\mathbf{A},\mathbf{R},\mathbf{S}'\rangle}\Big[\big(\mathbf{R} + \gamma \max_{\mathbf{A}'} Q_{\theta'}(\mathbf{S}',\mathbf{H}'_{t-1},\mathbf{A}') - Q_{\theta_t}(\mathbf{S},\mathbf{H}_{t-1},\mathbf{A})\big)\nabla_{\theta_t} Q_{\theta_t}(\mathbf{S},\mathbf{H}_{t-1},\mathbf{A})\Big] \tag{14}$$

Different from DQN, DRQN uses a LSTM layer instead of the first fully connected layer. Sequential states $\mathbf{S}_{t-k}$,..., $\mathbf{S}_{t-1}$ and $\mathbf{S}_t$ are processed through time by the LSTM layer and output the Q-values after the last fully connected layer. Note that we use the LSTM layer to train our network to understand temporal dependencies, so we can't randomly choose experiences from **D** like DQN. Hence, we randomly choose some traces of experiences of a given length, i.e., randomly select 2 traces of 2 continuous experiences, such as $\mathbf{e}_1$, $\mathbf{e}_2$ and $\mathbf{e}_9$, $\mathbf{e}_{10}$. Despite the changes in the neural network, the DRQN-based algorithm, i.e., *DR-Cell*, is almost same as D-Cell, and we summarize these two algorithm together in Algorithm 2.

4.4. Training data and transfer learning

With deep reinforcement learning, we can get the Q-function that outputs reward scores for all the possible actions under a certain state, then we can choose the cell that has the largest score in

[2] How to design the network structure is an important research problem, while it is not the main concern of this paper. Some other network structures could be modified for cell selection to deal with the heterogeneous inputs. For example, we can use a convolutional neural network to pretrain the recent-cycle selection. Then, the results and the rest of our state are fed into the fully connected layers.

Table 1
Statistics of three datasets.

	Sensor-Scope	U-Air	TaxiSpeed
City	Lausanne	Beijing	Beijing
Data	temperature, humidity	PM2.5	traffic speed
Cell size	$50{*}30$ m^2	$1000{*}1000$ m^2	road segment
Cell number	57	36	118
Cycle length	0.5 h	1 h	0.5 h
Duration	7 days	11 days	4 days
Mean ± Std.	6.04 ± 1.87 °C 84.52 ± 6.32 %	79.11 ± 81.21	13.01 ± 6.97m/s

cell selection. Obviously, the Q-function learning algorithm mentioned in the previous sections may need a large amount of training data, while in MCS, we cannot have an unlimited historical data for training. Then, can we reduce the amount of training data under certain circumstances?

The easiest way to deal with this problem is using a small amount of historical data to conduct the effective training set by random combination. The historical data can be obtained by a preliminary study on the target sensing area, i.e., collecting data from some cells for a short time before running. We randomly combine the sensed cells from the same cycles, and obtain many experiences, i.e., $\mathbf{e}_t = \langle \mathbf{S}, \mathbf{A}, \mathbf{R}, \mathbf{S}' \rangle$, to train our model. Note that we would like to select some redundant cells for each cycle, as an extreme example, we collect the data from all the cells for a short time. We use these redundant data to conduct various combinations of selected cells in one cycle which can satisfy our (*e, p*)-quality, which ensures the effectiveness of training.

In a practical application, we do not need to collect data from many cells, since the efficient cells under a certain state are finite, and thus the effective combinations which can satisfy the quality are limited. Therefore, we can select[3] a small amount of redundant cells to conduct a smaller but effective training set, which contains enough experiences for training. We have conducted some experiments in the Section 5.4 to show that our method can collect a small amount of redundant data to train our Q-function and achieve a good enough performance. However, this method still requires a preliminary study, and too much training on the a small amount of data may get a local optima.

Moreover, in a practical application, we further consider the periodic retraining as a supplement to our system. On the one hand, periodic retraining makes the system better able to deal with the environment changes. On the other hand, the system has collected more data after running a period of time, which can be used in the new training and further improve the performance of reinforcement learning. Note that the periodic retraining can be conducted in an offline manner, without affecting the availability of the online system. Moreover, our proposed transfer learning/fine-tuning techniques can be used to significantly reduce the re-training cost.

In order to make better use of the well trained Q-function and further reduce the amount of training data, we try to introduce the transfer learning technique into our problem. In reality, many types of data have inter-data correlations, e.g., temperature and humidity [23]. Then, if there are multiple correlated sensing tasks in a target area, probably the cell selection strategy learned for one task can benefit another task. With this intuition, we present a transfer learning method for learning the Q-function of an MCS task (*target* task) with the help of the cell selection strategy learned from another correlated task (*source* task). We assume that the *source* task has adequate training data, while the *target* task has only a little training data. Inspired by the fine-tuning techniques widely used in image processing with deep neural networks, for training the Q-function of the target task, we initialize the parameters of its DRQN to the parameter values of the source task DRQN (learned from the adequate training data of the source task). Then, we use the limited training data of the target task to continue the DRQN learning process (Algorithm 2). In this way, we can make use of the well trained Q-function and reduce the amount of training data required for obtaining a good cell selection strategy of the target task.

5. Evaluation

In this section, we conduct extensive experiments based on three real-world datasets, which contain various types of sensed data, including temperature, humidity, air quality, and traffic speed.

5.1. Datasets

We adopt three real-life datasets, *Sensor-Scope* [7], *U-Air* [30], and *TaxiSpeed* [14] to evaluate the performance of our proposed cell selection algorithms *D-Cell* and *DR-Cell*. These three datasets contain various types of sensed data, including temperature, humidity, air quality, and traffic speed. The detailed settings of three datasets are shown in Table 1. Although these sensed data in three datasets are collected from static sensors or stations, the mobile devices can also be used to obtain them (as in [2,5]). Thus, we can treat them as the data sensed by smartphones and use these datasets in our experiments to show the effectiveness of our algorithms.

Sensor-Scope [7]: The *Sensor-Scope* dataset contains the temperature and humidity readings for 7 days collected from the EPFL campus with an area about 500 m × 300 m. This target area is divided into 100 cells with the size 50 m × 30 m. The average temperature/humidity readings and their distributions are shown in Fig. 6. Since only 57 out of these 100 cells are deployed with valid sensors, we just use the sensed data at the 57 cells to evaluate our algorithms. The inference error is measured by mean absolute error.

U-Air [30]: The *U-Air* dataset collected the air quality data for 11 days from Beijing by existing monitor stations. Same as [30], we split the Beijing into cells where each cell is $1km \times 1km$. Then, there are 36 cells with the sensed air quality readings. With this dataset, we conduct the experiment of PM2.5 sensing, and try to infer the air quality index *category*[4] of unsensed cells. The inference error is measured by classification error.

TaxiSpeed [14]: The *TaxiSpeed* dataset contains the speed information in 4 days for road segments in Beijing. The dataset has more than 33,000 trajectories collected by GPS on taxis. Same as

[3] Without loss of generality, we randomly select cells for each cycle to collect data for training. Actually, the reinforcement learning-based algorithms also randomly select cells for the early stages as discussed in the previous sections.

[4] Six categories [30]: Good (0–50), Moderate (51–100), Unhealthy for Sensitive Groups (101–150), Unhealthy (150–200), Very Unhealthy (201–300), and Hazardous (> 300)

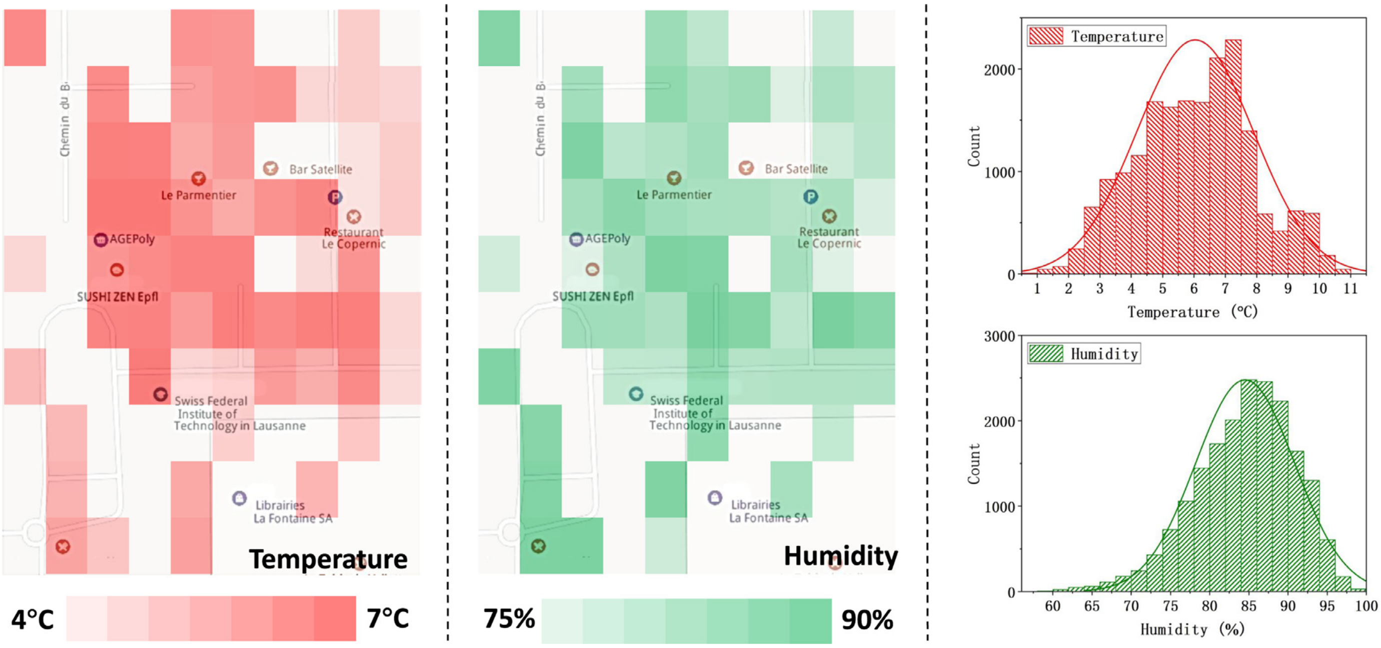


Fig. 6. The average temperature/humidity readings and their distributions in *Sensor-Scope*.

[31], we consider the road segments as the cells, and 118 road segments with the valid sensed values are selected to evaluate our algorithms. The inference error is measured by measure the mean absolute error.

5.2. Baseline algorithms

We compare D-Cell and DR-Cell to two existing methods: QBC and RANDOM.

QBC: Existing works on Sparse MCS mainly leverage Query by Committee in cell selection [20,23]. QBC selects the salient cell determined by "committee" to allocate the next task. More specifically, QBC attempts to use some different data inference algorithms (such as compressive sensing and K-Nearest Neighbors) to infer the full sensing matrix. Then, it chooses the cell where the inferred data of various algorithms has the largest variance as the next selection for sensing.

RANDOM: In each sensing cycle, RANDOM will randomly select cells one by one until the selected cells can ensure a satisfying inference accuracy. Note that RANDOM actually achieves a competitive performance since the random selection can already provide a lot of information to the powerful inference technologies as compressed sensing. Hence, we consider that RANDOM is suitable as a baseline.

5.3. Experiment process

To learn our proposed reinforcement learning-based algorithms, we use the first 10 h to 2 day data of each dataset to train our Q-function, i.e., we suppose that the MCS organizers will conduct a 10 h to 2 day preliminary study to collect data from the cells. Then we train our Q-function on the training data set by conducting various experiences until the Q-function is converged. We also vary the proportion of selected cells for each cycle, in order to show that a small amount of data can be conducted to an effective training set without loss of performance. Besides, we also conduct some experiments to evaluate our state and reward settings. We set discount factor $\gamma = 0.9$ and learning rate $\alpha = 0.05$ in Eq. (7) and dynamically adjust ϵ from 1 to 0.1 for whole process of training.

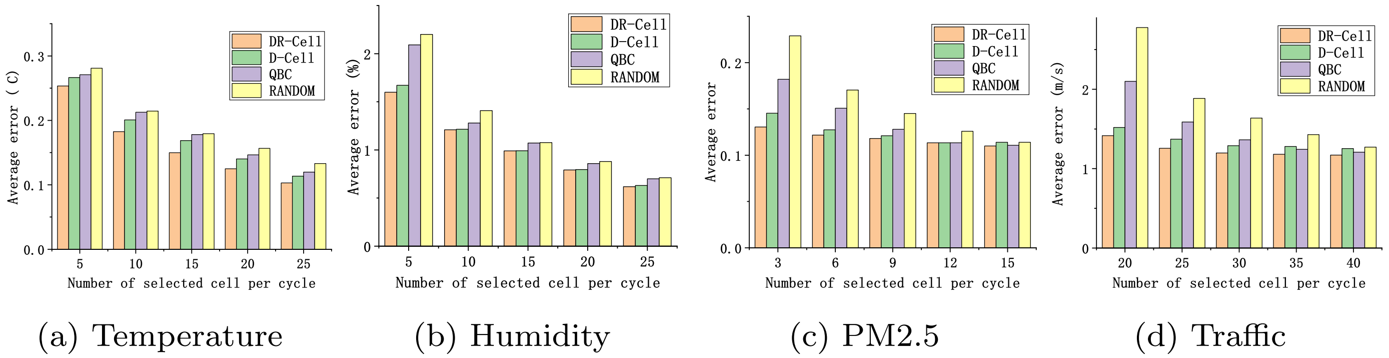
After the training stage, we obtain the well trained Q-function and enter the running stage. For each sensing cycle, we use the proposed cell selection algorithms to select the cells for sensing until the selected cells can satisfy the (e, p)-quality. Note that satisfying (e, p)-quality means that in $p \cdot 100\%$ of cycles, the inference error is not larger than e, which is practical in real world applications. Here, p should be set to a large value as 0.9 and 0.95, and we set e to a small value according to the sensing tasks, such as 0.25 °C for temperature. These large p and small ϵ build up a more reasonable and realistic scenario for Sparse MCS and could evaluate the effectiveness of our proposed algorithms well. Thus, our objective is to select cells as few as possible with the quality guarantee, and we will compare the number of cells selected by D-Cell, DR-Cell and baseline methods to verify the effectiveness of our proposed reinforcement learning-based algorithms.

5.4. Experiment results

We evaluate the performance by using the temperature and humidity data in *Sensor-Scope*, the PM2.5 data in *U-Air*, and the traffic speed data in *TaxiSpeed*, respectively. Without loss of generality, we first evaluate the performance without considering (e, p)-quality. We compare our inferred values with the real value to obtain the average inference error, while changing the number of selected cells for each cycle. As shown in Fig. 7, the results show the similar tendencies over four types of sensing tasks. Along with the increase of the number of selected cells, the average errors become smaller, since the more selected cells provide more information to help the data inference. Our proposed DR-Cell and D-Cell achieve the better performance than the other baseline algorithms, especially when the number of selected cells is small, which proves the effectiveness of our algorithms. Next, we will evaluate and discuss the performances of our cell selection algorithms considering (e, p)-quality, which is practical in real world applications.

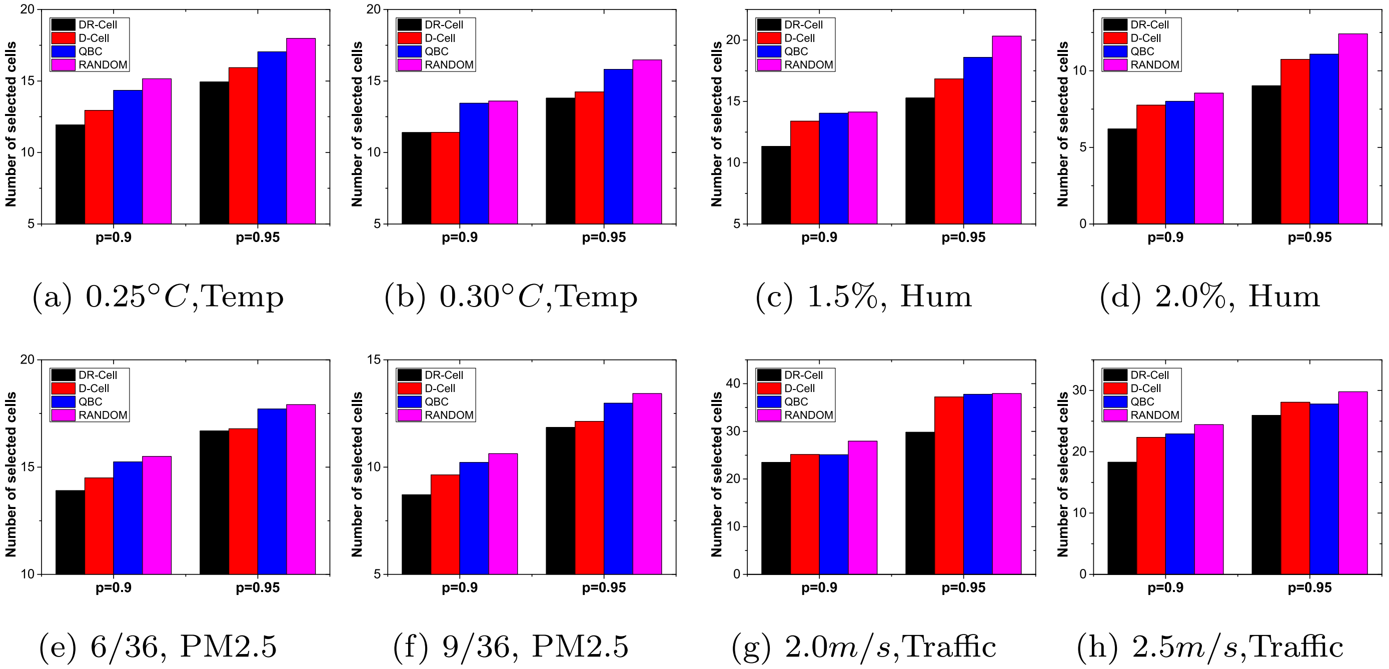
5.4.1. Number of selected cells

We consider the recent 5 cycles (the last 4 cycles and the current cycle), the last-time selection, and time as our state. The results are shown in Fig. 8 and Table 2.



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

Fig. 7. Average inference error for Temperature, Humidity, PM2.5, and Traffic Speed sensing tasks.



(a) $0.25°C$,Temp (b) $0.30°C$,Temp (c) 1.5%, Hum (d) 2.0%, Hum

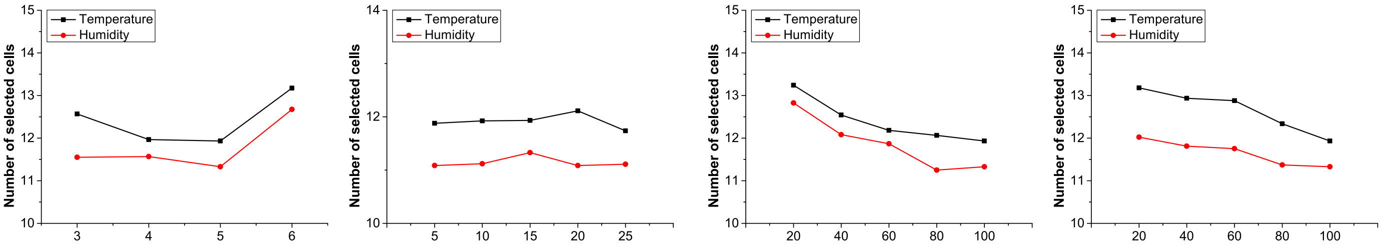
(e) 6/36, PM2.5 (f) 9/36, PM2.5 (g) $2.0m/s$,Traffic (h) $2.5m/s$,Traffic

Fig. 8. Number of selected cells for Temperature, Humidity, PM2.5, and Traffic Speed sensing tasks.

For the temperature in *Sensor-Scope*, we set the error bound e to 0.25 °C or 0.3 °C and p to 0.9 or 0.95 as the predefined (e, p)-quality. Thus, the quality requirement in this scenario is that the inference error is smaller than 0.25 °C or 0.3 °C for around 90% or 95% of cycles. The average numbers of selected cells for each sensing cycles have been shown in Fig. 8(a) and (b), where DR-Cell and D-Cell always outperform two baseline methods. Specifically, when (e, p) = (0.25 °C, 0.9), DR-Cell and D-Cell can select 16.8% and 9.7% fewer cells than QBC, and achieve 21.3% and 14.6% fewer cells than RANDOM. In general, DR-Cell only needs to select 11.93 out of 57 cells for each sensing cycle when ensuring the inference error below 0.25 °C in 90% of cycles. When we improve the quality requirement to $p = 0.95$, DR-Cell and D-Cell needs to select more cells to satisfy the higher requirement. Particularly, DR-

Table 2
Proportion of the cycles which satisfy the (e, p)-quality.

Temperature				Humidity			
(e, p)	D-Cell	DR-Cell	QBC	(e, p)	D-Cell	DR-Cell	QBC
(0.25 °C,0.9)	0.906	0.892	0.919	(1.5%, 0.9)	0.861	0.879	0.896
(0.25 °C,0.95)	0.957	0.948	0.965	(1.5%, 0.95)	0.933	0.957	0.940
(0.30 °C,0.9)	0.910	0.904	0.948	(2.0%, 0.9)	0.926	0.901	0.956
(0.30 °C,0.95)	0.976	0.957	0.974	(2.0%, 0.95)	0.969	0.961	0.975
PM2.5				Traffic Speed			
(e, p)	D-Cell	DR-Cell	QBC	(e, p)	D-Cell	DR-Cell	QBC
(6/36, 0.9)	0.901	0.896	0.930	(2.0 m/s, 0.9)	0.886	0.861	0.895
(6/36, 0.95)	0.951	0.957	0.961	(2.0 m/s, 0.95)	0.928	0.935	0.977
(9/36, 0.9)	0.918	0.909	0.925	(2.5 m/s, 0.9)	0.852	0.883	0.906
(9/36, 0.95)	0.968	0.944	0.950	(2.5 m/s, 0.95)	0.940	0.947	0.987



(a) Number of cycles kept in state

(b) R in reward

(c) Number of collected cycles for training

(d) Collected data in each training cycle (%)

Fig. 9. State, reward and training data for temperature and humidity sensing tasks ($e = 0.25°\text{C}/1.5\%$, $p = 0.9$).

Cell and D-Cell selects 14.93 and 15.93 out of 57 cells under the (0.25 °C, 0.95)-quality and achieves better performances by selecting 12.3%/6.4% and 16.9%/11.3% fewer cells than QBC and RANDOM, respectively. When we improve the error bound to $e = 0.3$ °C, DR-Cell and D-Cell need to select less cells since we have a lower quality requirement. Here, DR-Cell and D-Cell have the closer performances, and the number of sensed cells is reduced by 10.0% to 16.2%. For humidity in *Sensor-Scope*, a similar tendency is observed in Fig. 8(c) and (d), with quality requirement as (1.5%/2.0%, 0.9/0.95). Note that DR-Cell and D-Cell achieve better performances than QBC and RANDOM and DR-Cell performs better than D-Cell, since it would better capture the temporal patterns and handle partial observability in humidity of *Sensor-Scope*.

For the other two scenarios, i.e., PM2.5 in *U-Air* and traffic speed in *TaxiSpeed*, we get the similar observations, as shown in Fig. 8 (e)–(h). For the PM2.5 scenario, we set e as 6/36 or 9/36 and p as 0.9 or 0.95. When e is 6/36 and p is 0.9/0.95, DR-Cell selects 13.9/16.7 out of 36 cells and reduces 8.8%/5.8%, and 10.3%/6.8% of selected cells than QBC and RANDOM, respectively. When e is 9/36, the number of sensed cells is reduced by 8.7% to 18.0%. For traffic speed, we set e as 2 m/s or 2.5 m/s and achieve a reduced proportion as 6.4% to 20.0%. Note that D-Cell may underperform since the traffic speed has such a strong correlation with time, which can be better processed by our DR-Cell.

Table 2 shows the actual proportion of the cycles which satisfies the (e, p)-quality. We see that most of the values in the table are larger than its predefined p, which means that our proposed DR-Cell and D-Cell can provide the accurate inferences most of the time. Note that some results are slightly less than the predefined p, since compressive sensing and Bayesian inference in our algorithms have the intrinsic probabilistic characteristics and would cause some minor errors, which is within the acceptable range. Based on these results, we could say that our proposed algorithms can achieve a satisfactory performance.

5.4.2. State and reward

Then we evaluate the state and reward settings in reinforcement learning based cell selection, i.e., DR-Cell. We conduct some experiments on two MCS scenario, i.e., temperature and humidity monitoring. The state in our work consists of the recent-cycle selection, the last-time selection, and the time. Since the recent-cycle selection makes up the largest percentage and has the greatest impact on the next cell selection, we vary the last 3–6 cycles while keeping the others fixed in state, as shown in Fig. 9(a). We can see that when we keep the recent 4 or 5 cycles, our algorithms achieve the better performances, while the less or more cycles (3 or 6) would reduce them. This is probably due to the fact that the more cycles kept in state provide too much information of low value, which may disturb the outcome.

For the rewards, we would like to illustrate that the different values of R and c would not influence the performance after the Q-function has been well trained. In this paper, we consider all the costs c are the same and set the cost c as 1, without loss of generality. Note the case where the data collection costs of different cells are diverse could be considered and modified in the future work, by providing a more complex reward function. Then, we vary the R from 5 to 25, as shown in Fig. 9 (b), where the performances under different R are very close if the Q-function have enough training, and the small changes are most likely due to the randomness in our experiments.

5.4.3. Training data

Fig. 9 (c) and (d) illustrate that we could use a small amount of training data to train our Q-function while keep a good enough performance. We first study how the change of required cycles for training will impact the evaluation results. We collect data from all the cells and vary the cycles from 20 to 100, i.e., conduct a 10 h to 2 day (50 h) preliminary study in temperature and humidity monitoring tasks. As shown in Fig. 9(c), the reinforcement learning-based algorithm achieves a better performance with the increase in the number of cycles. When we have enough cycles for training, i.e., 80–100 cycles, the performances are very close. The reason could be that our proposed algorithms would capture the temporal correlation well by using a 2 day training data, while using the 10 h data cannot behave well. Then we use the 2 day data but randomly select part of cells for each cycle and conduct the training set. The results are shown in Fig. 9(d). The numbers of selected cells are increased along with the reduced proportions of collected data in each cycle for training, since the less collected data cannot conduct a comprehensive training set. However, the performances by using part of training data are even good enough. The DR-Cell using 20% training data still achieves better performances by selecting 8.8%/15.0% and 14.3%/14.9% fewer cells than QBC/RANDOM on temperature and humidity tasks, respectively.

5.4.4. Transfer learning

We then conduct the experiments on the multi-task MCS scenario, i.e., temperature-humidity monitoring, in *Sensor-Scope* to verify the transfer learning performance. We use DR-Cell to conduct 2-way experiments, *i.e.* temperature as the source task and humidity as the target task; and vice versa. More specifically, for the source task, we still suppose that we obtain 2 day data for training; but for the target task, we suppose that we only obtain 10 cycles (i.e., 5 h) of training data. Moreover, we add two compared methods to verify the effectiveness of our transfer learning method: **NO-TRANSFER** and **SHORT-TRAIN**. NO-TRANSFER is the method that directly uses the Q-function of the source task to the target task, and SHORT-TRAIN means that the target task model is only trained on the 10-cycle training data.

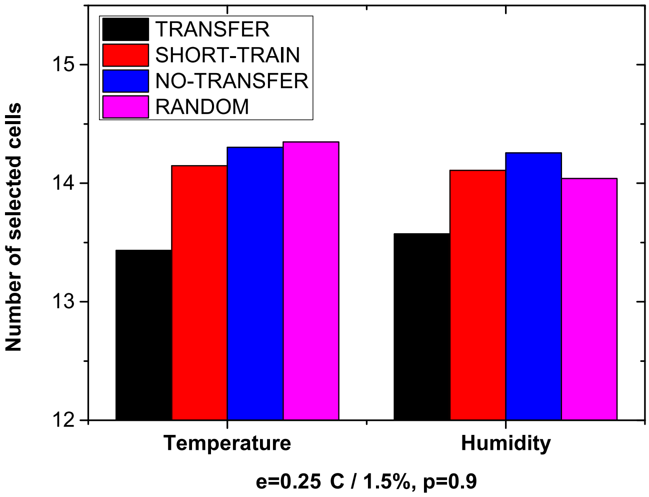


Fig. 10. Number of selected cells for temperature and humidity sensing tasks (transfer learning).

The quality requirement of temperature is (0.25°C, 0.9)-quality and the humidity is (1.5%, 0.9)-quality. Fig. 10 shows the average numbers of selected cells. When temperature is seen as the target task, TRANSFER can achieve better performance by reducing 5.0%, 6.0%, and 6.4% selected cells compared with NO-TRANSFER, SHORT-TRAIN, and RANDOM, respectively. When humidity is the target task, similarly, TRANSFER can select 4.0%, 5.0%, and 3.4% fewer cells than NO-TRANSFER, SHORT-TRAIN, and RANDOM, respectively. Note that NO-TRANSFER and SHORT-TRAIN even perform worse than RANDOM in this case. It emphasizes the importance of having an adequate amount of training data for DR-Cell. By using transfer learning, we can significantly reduce the training data required for learning a good Q-function in DR-Cell, and thus further reducing the data collection costs of MCS organizers.

5.4.5. Computation time

Finally, we report the computation time of DR-Cell. Our experiment platform is equipped with Intel Xeon CPU E2630 v4 @ 2.20 GHz and 32 GB RAM. We implement our D-Cell and DR-Cell training algorithms in TensorFlow (CPU version). In our experiment scenarios, the training time consumes around 2–4 h, which is totally acceptable in real-life deployments as the training is an offline process. Table 3 shows the running time of the online process, i.e., the testing stage in our experiments. Compared with ‘Cell Selection’, the ‘Quality Assessment’ costs the most since it needs to run the ‘Data Inference’ for some times to estimate the current quality by leave-one-out based Bayesian inference. In ‘Cell Selection’, although our algorithms need the off-line training, DR-Cell and D-Cell only need very little time (∼ 0.002s) to decide the next selected cell during the online processing, while QBC need ∼ 1s since it has to run various inference algorithms. We believe that it is worthy to conduct a ∼ 4 h offline training in order to achieve a faster and more efficient cell selection strategy.

Table 3
Runtime for each stage.

	Temperature	Humidity	PM2.5	Traffic Speed
Data Inference	0.49 s	0.50 s	0.35 s	0.97 s
Quality Assessment	4.43 s	4.46 s	4.75 s	8.01 s
DR-Cell	0.0015 s	0.0016 s	0.0007 s	0.0026 s
D-Cell	0.0014 s	0.0018 s	0.0009 s	0.0028 s
QBC	1.04 s	1.18 s	0.91 s	1.39 s

6. Conclusion

In this paper, we propose the novel reinforcement learning-based cell selection algorithms to improve the cell selection efficiency in Sparse MCS. First, we model the state, reward, and action for cell selection and propose a Q-learning based cell selection algorithm. To deal with the large state space, we use a neural networks to replace the Q-table, which is the DQN-based cell selection algorithm, and then modify the DQN with LSTM to catch the temporal patterns in our state and handle partial observability. Furthermore, we collect a small amount of redundant data to conduct the effective training by random combination and propose a transfer learning method to relieve the dependence on a large amount of training data. Extensive experiments verify the effectiveness of our proposed algorithms in reducing the data collection costs. In the future work, we would like to study how can we conduct the reinforcement learning-based cell selection in a completely online manner, so that we do not need a preliminary study stage for collecting the training data any more.

Declaration of Competing Interest

None.

Acknowledgements

This work is supported by the National Natural Science Foundation of China under Grant No. 61772230 and Natural Science Foundation of China for Young Scholars No. 61702215, Chinese Scholarship Council No. 201706170165, and China Postdoctoral Science Foundation No. 2017M611322 and No. 2018T110247. This work is supported in part by the NSFC under Grant No. 61572048 and 71601106, Hong Kong ITF Grant No. ITS/391/15FX.

Supplementary material

Supplementary material associated with this article can be found, in the online version, at doi:10.1016/j.comnet.2019.06.010.

References

[1] W.M. Bolstad, J.M. Curran, Introduction to Bayesian Statistics, John Wiley & Sons, 2016.

[2] S. Devarakonda, P. Sevusu, H. Liu, R. Liu, L. Iftode, B. Nath, Real-time air quality monitoring through mobile sensing in metropolitan areas, in: Proceedings of the 2nd ACM SIGKDD International Workshop on Urban Computing, ACM, 2013, p. 15.

[3] R.K. Ganti, F. Ye, H. Lei, Mobile crowdsensing: current state and future challenges, IEEE Commun. Mag. 49 (11) (2011).

[4] A. Gelman, H.S. Stern, J.B. Carlin, D.B. Dunson, A. Vehtari, D.B. Rubin, Bayesian Data Analysis, Chapman and Hall/CRC, 2013.

[5] D. Hasenfratz, O. Saukh, S. Sturzenegger, L. Thiele, Participatory air pollution monitoring using smartphones, Mobile Sens. 1 (2012) 1–5.

[6] M.J. Hausknecht, P. Stone, Deep recurrent q-learning for partially observable mdps, AAAI Fall Symposium Series (2015) 29–37. abs/1507.06527

[7] F. Ingelrest, G. Barrenetxea, G. Schaefer, M. Vetterli, O. Couach, M. Parlange, Sensorscope:application-specific sensor network for environmental monitoring, ACM Trans Sens Netw 6 (2) (2010) 1–32.

[8] L. Kong, M. Xia, X.-Y. Liu, G. Chen, Y. Gu, M.-Y. Wu, X. Liu, Data loss and reconstruction in wireless sensor networks, IEEE Trans. Parallel Distrib. Syst. 25 (11) (2014) 2818–2828.

[9] G. Lample, D.S. Chaplot, G. Lample, D.S. Chaplot, Playing FPS games with deep reinforcement learning, in: AAAI Conference on Artificial Intelligence, 2016.

[10] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, M. Riedmiller, Playing Atari with deep reinforcement learning, Comput. Sci. (2013).

[11] V. Mnih, K. Kavukcuoglu, D. Silver, A.A. Rusu, J. Veness, M.G. Bellemare, A. Graves, M. Riedmiller, A.K. Fidjeland, G. Ostrovski, et al., Human-level control through deep reinforcement learning, Nature 518 (7540) (2015) 529.

[12] R.K. Rana, C.T. Chou, S.S. Kanhere, N. Bulusu, W. Hu, Ear-phone: an end-to-end participatory urban noise mapping system, in: Proceedings of the 9th ACM/IEEE International Conference on Information Processing in Sensor Networks, ACM, 2010, pp. 105–116.

[13] M. Roughan, Y. Zhang, W. Willinger, L. Qiu, Spatio-temporal compressive sensing and internet traffic matrices, IEEE/ACM Trans. Netw. (ToN) 20 (3) (2012) 662–676.
[14] J. Shang, Y. Zheng, W. Tong, E. Chang, Y. Yu, Inferring gas consumption and pollution emission of vehicles throughout a city, in: Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2014, pp. 1027–1036.
[15] D. Silver, A. Huang, C.J. Maddison, A. Guez, L. Sifre, G.V.D. Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, Mastering the game of go with deep neural networks and tree search, Nature 529 (7587) (2016) 484.
[16] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, et al., Mastering the game of go without human knowledge, Nature 550 (7676) (2017) 354.
[17] R. Sutton, A. Barto, Reinforcement Learning: An Introduction, MIT Press, 2005.
[18] E. Wang, Y. Yang, J. Wu, W. Liu, X. Wang, An efficient prediction-based user recruitment for mobile crowdsensing, IEEE Trans. Mob. Comput. 17 (1) (2018) 16–28.
[19] J. Wang, Y. Wang, D. Zhang, F. Wang, Y. He, L. Ma, PSAllocator: multi-task allocation for participatory sensing with sensing capability constraints, in: Proceedings of the 2017 ACM Conference on Computer Supported Cooperative Work and Social Computing, ACM, 2017, pp. 1139–1151.
[20] L. Wang, D. Zhang, A. Pathak, C. Chen, H. Xiong, D. Yang, Y. Wang, Ccs-ta: quality-guaranteed online task allocation in compressive crowdsensing, in: Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing, ACM, 2015, pp. 683–694.
[21] L. Wang, D. Zhang, Y. Wang, C. Chen, X. Han, A. M'hamed, Sparse mobile crowdsensing: challenges and opportunities, IEEE Commun. Mag. 54 (7) (2016) 161–167.
[22] L. Wang, D. Zhang, D. Yang, B.Y. Lim, X. Ma, Differential location privacy for sparse mobile crowdsensing, in: Data Mining (ICDM), 2016 IEEE 16th International Conference on, IEEE, 2016, pp. 1257–1262.
[23] L. Wang, D. Zhang, D. Yang, A. Pathak, C. Chen, X. Han, H. Xiong, Y. Wang, SPACE-TA: cost-effective task allocation exploiting intradata and interdata correlations in sparse crowdsensing, ACM Trans. Intell. Syst.Technol. 9 (2) (2017) 1–28.
[24] L. Xiao, T. Chen, C. Xie, H. Dai, V. Poor, Mobile crowdsensing games in vehicular networks, IEEE Trans. Veh. Technol. PP (99) (2017). 1–1
[25] L. Xiao, Y. Li, G. Han, H. Dai, H.V. Poor, A secure mobile crowdsensing game with deep reinforcement learning, IEEE Trans. Inf. ForensicsSecur. PP (99) (2017). 1–1
[26] H. Xiong, D. Zhang, L. Wang, H. Chaouchi, EMC 3: energy-efficient data transfer in mobile crowdsensing under full coverage constraint, IEEE Trans. Mob. Comput. 14 (7) (2015) 1355–1368.
[27] L. Xu, X. Hao, N.D. Lane, X. Liu, T. Moscibroda, More with less: Lowering user burden in mobile crowdsourcing through compressive sensing, in: Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing, ACM, 2015, pp. 659–670.
[28] Y. Yang, W. Liu, E. Wang, J. Wu, A prediction-based user selection framework for heterogeneous mobile crowdsensing, IEEE Trans. Mob. Comput. (2018).
[29] D. Zhang, L. Wang, H. Xiong, B. Guo, 4W1H in mobile crowd sensing, IEEE Commun. Mag. 52 (8) (2014) 42–48.
[30] Y. Zheng, F. Liu, H.P. Hsieh, U-Air: when urban air quality inference meets big data, in: ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2013, pp. 1436–1444.
[31] Y. Zhu, Z. Li, H. Zhu, M. Li, Q. Zhang, A compressive sensing approach to urban traffic estimation with probe vehicles, IEEE Trans. Mob. Comput. 12 (11) (2013) 2289–2302.

Wenbin Liu received his B.S. degree in physics from Jilin University, Changchun, China in 2012; and M.E. degree in department of software from Jilin University, Changchun in 2016. He is currently a Ph.D. candidate in the Department of Computer Science and Technology, Jilin University, Changchun. His current research focuses on the Mobile CrowdSensing.

Leye Wang is currently a research associate in the Hong Kong University of Science and Technology, working with Prof. Qiang Yang and Prof. Xiaojuan Ma. In May 2016, he obtained Ph.D. at Institut Mines-Telecom (IMT) and Universite Pierre et Marie CURIE (UPMC), Paris, under the supervision of Prof. Daqing ZHANG and Prof. Abdallah MHAMED. He received his B.S. (2009) and M.S.(2012) in computer science from Peking University, Beijing, under the supervision of Prof. Bing XIE. His research interests include mobile crowdsensing and ubiquitous computing.

En Wang is the corresponding author, email: wangen@jlu.edu.cn, received his B.E. degree in software engineering from Jilin University, Changchun in 2011, his M.E. degree in computer science and technology from Jilin University, Changchun in 2013, and his Ph.D. in computer science and technology from Jilin University, Changchun in 2016. He is currently an associate professor in the Department of Computer Science and Technology at Jilin University, Changchun. He is also a visiting scholar in the Department of Computer and Information Sciences at Temple University in Philadelphia. His current research focuses on the efficient utilization of network resources, scheduling and drop strategy in terms of buffer-management, energy-efficient communication between humancarried devices, and mobile crowdsensing.

Yongjian Yang received his B.E. degree in automatization from Jilin University of Technology, Changchun, Jilin, China in 1983; his M.E. degree in computer communication from Beijing University of Post and Telecommunications, Beijing, China in 1991; and his Ph.D. in software and theory of computer from Jilin University, Changchun, Jilin, China in 2005. He is currently a professor and a PhD supervisor at Jilin University, the Vice Dean of the Software College of Jilin University, Director of Key lab under the Ministry of Information Industry, 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.

Djamal Zeghlache graduated from SMU in Dallas, Texas in 1987 with a Ph.D. in Electrical Engineering and joined the same year Cleveland State University as an Assistant Professor. In 1990 and 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 at Telecom SudParis of Institut Telecom where he currently acts as Professor and Head of the Wireless Networks and Multimedia Services Department. Professor Zeghlache is also acting Dean of Research of Telecom SudParis. He co-authored around one hundred publications in ranked international conferences and journals and was an editor for IEEE Transactions on Wireless. His interests and research activities span a broad spectrum related to fixed and wireless networks and services. The 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.

Daqing Zhang is a professor at Peking University, China, and Telecom Sud-Paris, France. He obtained his Ph.D from the University of Rome La Sapienza, Italy, in 1996. His research interests include contextaware computing, urban computing, mobile computing, and so on. He served as the General or Program Chair for more than 10 international conferences. He is an Associate Editor for ACM Transactions on Intelligent Systems and Technology, IEEE Transactions on Big Data, and others.