

Demand-Driven Sparse Mobile Crowdsensing With Neighborhood-Aware Reconstruction

Yuhao Wang, Qihang Zhou, Yu Zhang, Guoqiang Deng^{1b}, Lingyu Liang^{2b}, and Xinglin Zhang^{3b}, *Member, IEEE*

Abstract—Sparse mobile crowdsensing (SMCS) is a cost-effective paradigm aimed at recruiting workers to complete sensing tasks and inferring the remaining unobserved data, with broad applications in large-scale, fine-grained monitoring services. In SMCS, spatial coverage of the sensing area or global completion accuracy is typically used as the performance metric. However, in many real-world service scenarios (e.g., temperature, humidity, air quality monitoring), users are generally only interested in data from their specific regions and expect the highest possible data accuracy. In such cases, relying solely on coverage or global completion error fails to adequately assess the quality of the platform's service. To address this and satisfy users' sensing demands as much as possible while maintaining low sensing costs, we propose the demand-driven framework with neighborhood-aware data reconstruction (D2-SMCS), which integrates regional population demand calculation, dynamic clustering, and data reconstruction. Unlike existing approaches, we introduce Quality of Service (QoS) as a performance metric based on regional population demand. First, we quantify the interest level of sensing tasks in different regions by considering factors, such as population demand and data fluctuation. Based on this quantification, the dynamic clustering module selects the regions most beneficial for accurate data completion. Finally, to overcome the limitation of traditional matrix completion methods in capturing short-term variations, we propose an innovative neighborhood-aware latent matrix completion (NALMC) approach to infer and complete the unobserved regions. Extensive experiments on real-world datasets demonstrate the effectiveness of our framework.

Index Terms—Matrix completion, Quality of Service (QoS), region selection, sparse mobile crowdsensing (SMCS).

I. INTRODUCTION

MOBILE crowdsensing (MCS) is a sensing paradigm that recruits numerous participants to perform sensing

tasks at designated times and locations, offering an effective solution to challenging sensing problems [1], [2], [3]. With the widespread adoption of smart devices equipped with advanced sensing and computational capabilities, MCS has been applied in large-scale environmental information monitoring, such as air quality [4], [5], noise [6], and traffic data monitoring [7]. Among the various service approaches enabled by the widespread use of MCS [2], this article focuses on the sensing as a service (S²aaS) model, where the MCS platforms collect available sensor data and deliver data services to users [8]. While MCS enables the acquisition of fine-grained, high-quality data, MCS platforms inevitably incur substantial expenses to incentivize participants. Given that platforms and organizers are highly cost-sensitive, achieving high-quality sensing data at a minimal cost has long been a critical issue in the MCS domain.

To tackle this issue, sparse mobile crowdsensing (SMCS) has emerged as a cost-effective approach that leverages inherent data correlations to infer unsensed data based on limited sensing areas [9], [10], [11]. SMCS effectively reduces costs by directly minimizing the number of sensing tasks, drawing increasing attention in the field. The core problem in SMCS is how to reduce costs while maintaining adequate sensing quality. Naturally, this raises the question: How should sensing quality be defined? Existing research generally focuses on two aspects: 1) spatial coverage of the sensing area [12] and 2) global completion accuracy [13]. Some studies have proposed additional quality metrics, such as maximum inference error [14], adding a new dimension to data completion evaluation. However, given that SMCS shares a similar S²aaS model with MCS, it is essential to maximize the service quality perceived by users in practical environments. Consequently, it becomes evident that existing evaluation metrics are inadequate for many task scenarios.

For instance, consider a sensing task aimed at providing accurate temperature data for a region that includes both urban areas and lakes. Since urban areas are densely populated, prioritizing sensing efforts in those regions would likely enhance the overall service quality. However, if metrics, such as spatial coverage of the sensing area or global completion accuracy are used to guide the SMCS platform in assigning sensing tasks and collecting data, the urban areas and lake areas would be treated with equal weight by the platform. As a result, an unnecessary number of workers may be dispatched to the vicinity of lakes to perform sensing tasks, while urban areas lose opportunities to further improve sensing accuracy and enhance the service quality provided to users by the

Received 7 February 2025; revised 2 April 2025; accepted 24 April 2025. Date of publication 2 May 2025; date of current version 9 July 2025. This work was supported in part by the National Natural Science Foundations of China under Grant 62372185, and in part by the Guangdong Basic and Applied Basic Research Foundation under Grant 2024A151010131. (Corresponding authors: Guoqiang Deng; Xinglin Zhang.)

Yuhao Wang, Qihang Zhou, Yu Zhang, and Xinglin Zhang are with the School of Computer Science and Engineering, South China University of Technology, Guangzhou 510006, China (e-mail: zhxlinsc@gmail.com).

Guoqiang Deng is with the Information and Network Engineering and Research Center, South China University of Technology, Guangzhou 510006, China (e-mail: denggq@scut.edu.cn).

Lingyu Liang is with the South China University of Technology, Guangzhou 510641, China, also with the Henan Key Laboratory of Chronic Disease Prevention and Therapy and Intelligent Health Management, Henan Institute of Interconnected Intelligent Health Management, Zhengzhou 450045, China. Digital Object Identifier 10.1109/IJOT.2025.3566314

platform. This example highlights a critical limitation: relying solely on existing metrics cannot effectively guide platforms in maximizing the quality of the sensing service experienced by users. Therefore, the first major challenge we face is to develop a new sensing quality evaluation metric that explicitly incorporates user-perceived service quality while balancing cost efficiency and data reliability.

As a critical factor in determining the distribution of sensing information, task allocation plays a fundamental role in SMCS [10], [15], [16], [17]. To minimize sensing costs, platforms must select the fewest possible sensing units, prioritizing those that are most representative within the sensing map. Data inference based on representative cells typically yields a higher completion accuracy than random selection. However, the importance of each sensing area varies and often changes over time, making task allocation increasingly complex. The second challenge, therefore, is determining how to identify the most information-rich representative cells in such dynamic conditions to enhance data inference in completion algorithms.

The raw sparse sensing information serves as the foundation for subsequent inference, with data reconstruction being the core process in SMCS to generate a fully completed sensing map. Most existing methods employ matrix completion or its variants for data reconstruction [18], [19]. As a latent factor model, matrix completion excels in extracting global information and achieving good global completion accuracy. However, empirical observations reveal that matrix completion performs poorly in responding to short-term data changes. For instance, if multiple regions exhibit sharp increases in recent sensing data, possibly due to extreme weather, matrix completion, focusing on global patterns, may infer abnormally low values for unsensed regions due to the influence of historical data. This limitation arises from the inherent characteristics of matrix completion. Given that many tasks require continuous adaption to environmental changes, the third challenge in SMCS is how to retain global information capture while enhancing responsiveness to local information changes.

To address these challenges, we propose the Demand-Driven Framework with Neighborhood-Aware Data Reconstruction (D2-SMCS) in this article. The framework consists of three main components: regional population demand calculation, dynamic clustering, and data reconstruction. Fig. 1 shows an example of the D2-SMCS process. In this example, upon the release of a task on the platform, we first calculate each region's contribution to Quality of Service (QoS) based on population demand (i.e., which regions the task prioritizes) and historical data for the task. The dynamic clustering module then determines the final sensing area selection, followed by data reconstruction using the collected data, which updates the region's QoS contribution in the next timeslot. These three components form an interconnected and cohesive whole. To accurately reflect perceived service quality, we propose a regional population-based QoS evaluation metric, which incorporates both population demand and regional data fluctuation. This metric guides the framework in allocating sensing resources more to densely populated or high-demand areas and less to sparsely populated or low-demand areas. For

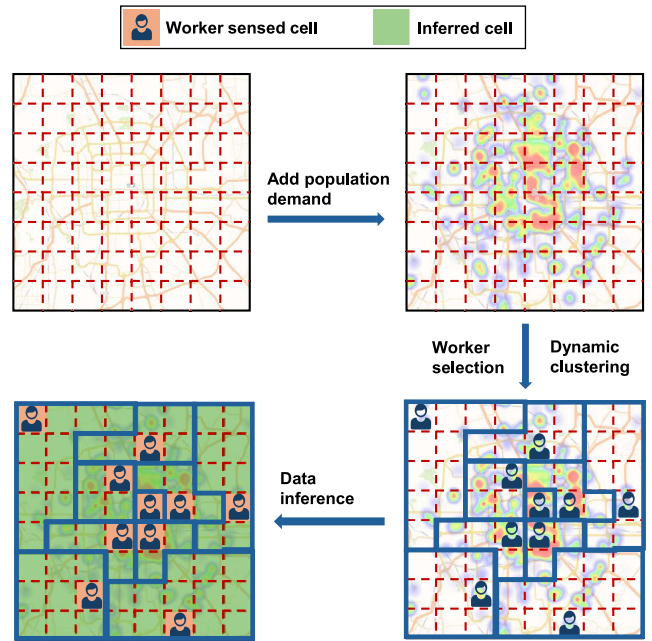


Fig. 1. Example flow of D2-SMCS.

selecting representative sensing areas, the dynamic clustering module groups areas with redundant information based on geographic and data similarity, then selects one target per cluster. Finally, observing that traditional latent factor models struggle with short-term changes, we incorporate neighborhood and spatiotemporal factors to improve matrix completion, applying alternating least squares (ALS) for offline and online settings. We also integrate K -nearest neighbors (KNN) model completion results to retain global information extraction while enhancing short-term information capture.

In summary, this article makes the following contributions.

- 1) We propose D2-SMCS, an innovative SMCS framework, which is the first to incorporate population demand and service quality factors, using them to guide task allocation and data reconstruction. This effectively improves the service quality and stability accepted by the crowd at the same cost.
- 2) We introduce a sensing area QoS evaluation metric based on task and population demand, guiding task allocation with a more realistic approach to evaluating task quality.
- 3) We present a dynamic clustering module that considers geographical location, historical data, and cost constraints, selecting representative sensing areas for task allocation.
- 4) We propose neighborhood-aware latent matrix completion (NALMC) as data reconstruction method, incorporating spatiotemporal and neighborhood factors and combining with the KNN model and ALS, enabling usage in both offline and online scenarios.
- 5) We extensively evaluate our framework on two classical environmental sensing datasets, comparing traditional sensing errors and service quality metrics. The results demonstrate the effectiveness, applicability, and stability of NALMC, showing significant service quality

improvements under the same costs with stable performance across multiple datasets.

The remainder of this article is organized as follows. Section II discusses related work, Section III provides definitions, assumptions, and an overview of the framework, Section IV elaborates on D2-SMCS's details, Section V presents performance evaluation, and Section VI concludes this article.

II. RELATED WORKS

MCS is a paradigm that harnesses collective power to accomplish tasks that are nearly impossible for individuals to complete alone [2], [3]. It is now applied in various large-scale sensing scenarios, such as monitoring traffic conditions [20], air pollution levels [21], and watershed status [22]. Organizers or MCS platforms recruit many participants to perform sensing tasks, creating high-quality sensing maps. However, allocating tasks to every sensing area within the map may lead to prohibitively high costs and inefficient use of resources, including participants' time and effort, as well as the compensation provided to them. To address this issue, a new MCS-based approach, called SMCS, has emerged [10]. This approach only collects data from a limited portion of the sensing area and then infers data for unsensed regions. The core challenges in SMCS revolve around task area selection, data reconstruction and quality evaluation.

Task area selection focuses on identifying areas most beneficial to the sensing objective. For instance, Wang et al. [13] employed a query by committee (QBC) method to select significant units for the next task. Liu et al. [23] proposed a cell selection algorithm based on reinforcement learning, using a deep Q -network to learn the Q function, assisting in selecting the optimal cell under specific conditions. Meanwhile, Zhu et al. [24] modeled the area selection problem as a solvable bi-objective optimization and introduces a novel three-step cell selection approach, achieving significant cost reduction by modeling information, estimating costs, and selecting cost-quality optimal cells for SMCS.

Data reconstruction is a central focus of SMCS, with current research exploring various methods to reconstruct complete sensing maps accurately from sparse data. Among the popular inference methods, we can categorize them into latent factor models and neighborhood models. Latent factor models primarily involve matrix completion [25] and its derivatives. For example, Liu et al. [18] apply an enhanced ALS method with spatiotemporal constraints for online matrix completion. Xie et al. [26] integrated bidirectional graph-based sensing scheduling with matrix completion to actively determine sampling locations per time segment, accurately recovering unsampled data despite sensing and communication errors. Additionally, Fan and Cheng [27] combined traditional linear matrix factorization with deep neural networks to propose a deep matrix factorization (DMF) approach, effectively capturing nonlinear spatiotemporal features from sparse matrices. The neighborhood model, represented by KNN, has limited applications in SMCS but is widely used in recommendation systems [28]. Intuitively, recommendation

systems share similarities with SMCS data inference scenarios, and neighborhood models offer distinct advantages over latent factor models. Inspired by Koren [29], which combines latent factor and neighborhood models to achieve promising results in recommendation systems, our model further integrates these two approaches, incorporating incremental updates to suit online settings and optimizing the objective function for enhanced completion accuracy and stability. Additionally, some studies approach completion from different perspectives. For instance, Wei et al. [14] employed Bayesian compressive sensing (BCS) with projection matrices capturing temporal, spatial, and historical correlations, reconstructing a complete sensing map from partial data under noisy conditions.

Several studies leverage the correlation within sensing data to improve completion accuracy [30]. Wang et al. [31] exploited spatiotemporal correlations to dynamically select a small number of subareas for sensing at each time point, significantly reducing task allocation. Similarly, Liu et al. [18] incorporated spatiotemporal terms into the matrix completion objective function, achieving improved reconstruction. In addition to correlations within the same data type, some studies explore cross-data correlations. Wang et al. [32] observed interdata correlations, leveraging them to capture outliers and enhance completion accuracy.

Lastly, evaluating task quality is also crucial in SMCS. Most existing studies use spatial coverage of the sensing area or global completion accuracy as evaluation metrics. Wang et al. [13] used global completion error as a quality measure, while Wei et al. [14] considered both global error and maximum inference error. Zhao et al. [12] focused on minimizing costs by selecting the fewest participants to meet predefined coverage requirements for each timeslot, using spatial coverage as a quality metric. Notably, current task quality definitions are technically oriented; however, with a human-centered approach, we define a new task quality evaluation metric in this article by integrating insights from the service quality domain [33].

III. SYSTEM FRAMEWORK

In this section, we first introduce the necessary definitions and assumptions. Then we formulate the SMCS problem we aim to solve and provide the overview of our solution. For easy reference, the main notations are listed in Table I.

A. Definitions and Assumptions

Definition 1 (Ground Truth Sensing Matrix): For an SMCS task involving N cells and K sensing timeslots, the *ground truth sensing matrix* is represented as $M_{N \times K}$. Each entry $M[i, j]$ indicates the true sensing data of cell i in cycle j .

Definition 2 (Selection Matrix): In the *selection matrix* $S_{N \times K}$, each entry $S[i, j]$ specifies whether the corresponding entry in the ground truth sensing matrix $M[i, j]$ is selected for sensing. If cell i is selected for sensing in timeslot j , then $S[i, j] = 1$; otherwise, $S[i, j] = 0$.

Definition 3 (Sensing Data Matrix): The *sensing data matrix* $D_{N \times K}$ records the actual collected sensing data:

$$D = M \circ S \quad (1)$$

TABLE I
LIST OF SYMBOLS AND DEFINITIONS

Symbol	Description
M, \hat{M}	Groundtruth matrix and reconstructed matrix
N, K	Number of sensing areas and time slots
D	Sensing data matrix $D \in \mathbb{R}^{N \times K}$
S	Selection matrix indicating the chosen cells for sensing $S \in \{0, 1\}^{N \times K}$
U, V	Latent feature matrices for users and items; $U \in \mathbb{R}^{N \times r}$, $V \in \mathbb{R}^{K \times r}$
R	Demand matrix representing the importance level of each area's contribution to service quality.
F	Feature matrix, used to extract the historical data distribution of each area for clustering.
W	Similarity matrix
Q	Quality of service (QoS) score
\mathcal{E}	Sensing error between groundtruth matrix and reconstructed matrix
r	The rank of latent factors
α	Sparse ratio
$\lambda, \lambda_t, \lambda_s$	Regularization for complexity, temporal, and spatial smoothness
γ_u, γ_v	Regularization for neighborhood constraints
k	Number of nearest neighbors
$\mathcal{N}_u, \mathcal{N}_i$	k nearest neighbors for item u and item i
w_{uj}, w_{ik}	Similarity weights for u - j and i - k item pairs
$\hat{M}_{ALS}, \hat{M}_{KNN}$	Reconstructed matrices from ALS and KNN models
ρ	Weighting parameter to combine ALS and KNN results
ρ^*	Optimal ρ minimizing reconstruction error
$E(\rho)$	Reconstruction error on validation set for ρ

where \circ denotes the element-wise product of matrices M and S .

Definition 4 (Reconstructed Matrix): The reconstructed matrix $\hat{M}_{N \times K}$ is generated by filling in missing data in the sensing data matrix using data reconstruction methods.

Definition 5 (Sensing Error): The sensing error quantifies the difference between elements in the reconstructed full sensing matrix \hat{M} and the true ground truth sensing matrix M . For cell i in timeslot j , the sensing error is defined as:

$$\mathcal{E}_{ij} = \text{error}(\hat{M}[i, j], M[i, j]) \quad (2)$$

where the specific error function, $\text{error}()$, depends on the specific setup.

Definition 6 (Sparse Ratio): To limit the cost, which corresponds to the number of sensing areas, we introduce a *sparse ratio* denoted by α . In each timeslot, at most $(1 - \alpha) \times N$ regions can be selected for sensing. The value range of α is $[0, 1]$.

Definition 7 (QoS Score): For an SMCS task, the QoS score Q measures the service quality represented by the task's completion. It is defined as

$$Q = \sum_{i=1}^N \text{Demand}_i \times \left(1 - \frac{1}{K} \sum_{j=1}^K \mathcal{E}_{i,j} \right) \quad (3)$$

where \mathcal{E} denotes the sensing error. The demand varies depending on the task type; for instance, in most environmental sensing tasks, population density can be used to quantify the demand.

In this article, we make the following assumptions.

Assumption 1: Each timeslot is sufficiently long to ensure that workers or sensing devices selected for the sensing area can complete the sensing task within the slot. This assumption guarantees that all sensing tasks distributed by the system can be completed as expected.

Assumption 1 ensures that, once tasks are assigned to workers, they are able to complete the tasks and submit the results to the platform in a timely manner. For most environmental sensing tasks, such as detecting temperature, humidity, and PM2.5 levels, workers can typically complete the task within a few tens of seconds using the sensors built into their devices [15]. Furthermore, since we are not currently focusing on the mobility of workers, we select workers who are located near the task site for task assignment. As a result, this assumption can be satisfied for typical time slots (e.g., 10 min).

Assumption 2: All recruited workers and sensing devices are reliable and will return the true values of the task location without maliciously providing incorrect values.

Assumption 2 ensures that we do not need to consider the issue of user trustworthiness. This assumption is consistent with the existing works [34]. The main purpose of this assumption is to simplify the complexity of evaluating workers' abilities and reputations, and to focus on how to better allocate tasks. In practical applications, if the issue of malicious users needs to be addressed, reputation mechanisms can be employed to assess workers and ensure the credibility of the collected data [35], [36]. By modeling workers' reputations, tasks can be assigned to more trustworthy workers, effectively resolving reliability issues.

B. Problem Formulation

Given N sensing regions and K timeslots, our objective is to determine the optimal selection matrix S , with the goal of maximizing the QoS score Q within the sparse ratio α :

$$\begin{aligned} & \underset{S}{\text{maximize}} && \sum_{i=1}^N \sum_{j=1}^K \mathcal{Q}_i^j \\ & \text{s.t.} && \hat{M} \circ S = M \circ S \\ & && \sum_{i=1}^N \sum_{j=1}^K S[i, j] \leq (1 - \alpha) \cdot (N \times K) \end{aligned} \quad (4)$$

where the first constraint requires that \hat{M} must match M at selected points, ensuring data consistency, and the second constraint limits the number of observations to $(1 - \alpha)$ of M , balancing quality with efficient resource use.

C. Overview of D2-SMCS

D2-SMCS is an SMCS framework comprising the SMCS platform, requesters, and participants. Requesters seek services provided by the platform (e.g., querying real-time temperature

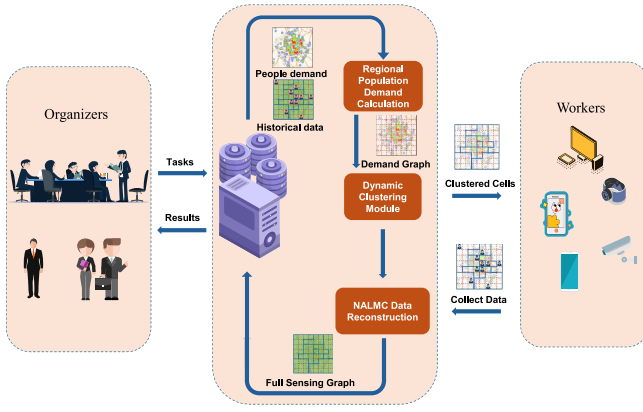


Fig. 2. Framework of D2-SMCS.

or air pollution levels in a specific region). The platform fulfills these requests by distributing sensing tasks to participants to collect the necessary data. Our framework primarily focuses on how the platform optimally assigns tasks and utilizes collected data to deliver the highest possible QoS while adhering to cost constraints (defined by sensing sparsity ratio settings). The architecture of D2-SMCS is illustrated in Fig. 2, and its workflow is described in detail below:

Regional Population Demand Calculation: First, we calculate a demand score for each region in the current timeslot, considering historical data fluctuations and population density (In Section IV-A, we provide a detailed explanation of why population density is used as an indicator of population demand). The more accurate information we can gather from high-demand regions, the greater the overall service quality. This calculation provides a basis for assigning tasks to participants.

Dynamic Clustering Module: Next, to determine the tasks to be distributed, we perform an initial clustering of available regions based on geographical proximity. Clustering continues until each cluster's population demand score reaches a defined upper limit, after which new clusters are formed. We then dynamically refine these clusters by aggregating regions with similar recent sensing data distributions. Based on the sparsity requirement, we adjust clusters with excessive or insufficient results. In the end, a random selection of one region per cluster forms the sensing tasks for the current timeslot. Given the geographical or data distribution proximity within clusters, any region in a cluster can represent a substantial amount of the cluster's information.

Sensing Data Reconstruction: The platform reconstructs global sensing data using partial data collected by participants. We enhance the conventional latent factor model for data completion by integrating temporal, spatial, and neighborhood information. The results from this improved completion process are then fused with results from a KNN model, leveraging the strengths of both models to achieve high-precision data completion.

IV. DETAILED DESIGN OF D2-SMCS

In this section, we provide a comprehensive introduction to the components of our D2-SMCS framework, including

regional population demand calculation, dynamic clustering, and data reconstruction. It should be noted that since D2-SMCS is designed to adapt to a wide range of application scenarios, it can be deployed in both offline and online modes. For offline tasks, we have more complete sensing information, while for online tasks, we need to analyze the task completion status of the previous timeslot and make timely adjustments to the task allocation strategy for the next timeslot. The differences between these two modes require the corresponding adjustments in our approach. In the following discussion, we will categorize the explanation where necessary and present the results for both offline and online modes in Section V.

A. Regional Population Demand Calculation

A core factor in the D2-SMCS framework is the QoS provided to the population. Therefore, it is essential to reasonably quantify the importance of service quality in a region. We consider two aspects: historical data and population density.

Historical Data Perspective: We first consider selecting the significant cells from the historical data, which are defined as cells that show significant differences either when compared to other cells within the same timeslot, or in relation to their own historical data. Here, a cell refers to the smallest geographic unit in the dataset, typically consistent-sized rectangles. These significant cells often indicate that a major change has occurred in the corresponding area, and thus warrant greater attention. To identify regions with significant differences from other cells in the same timeslot, we introduce the Z-score. The Z-score is a statistical measure that tells us how many standard deviations a data point is from the mean. By examining the Z-score values, we can identify the areas that are worth paying attention to.

We define $v_{i,j}$ as the actual sensing data at the j th timeslot, and n as the number of sensed regions at the i th timeslot. The average sensing data at the j th timeslot is then calculated as

$$\mu_j = \frac{1}{n} \sum_{i=1}^n v_{i,j}. \quad (5)$$

The standard deviation is

$$\sigma_j = \sqrt{\frac{1}{n} \sum_{i=1}^n (v_{i,j} - \mu_j)^2}. \quad (6)$$

Thus, the Z-score for each element is obtained by

$$z_{i,j} = \left| \frac{v_{i,j} - \mu_j}{\sigma_j} \right|. \quad (7)$$

To find regions with significant changes compared to the previous timeslot, even if data points do not deviate from the mean of other cells in the current timeslot, we define

$$\Delta v_{i,j} = |v_{i,j} - v_{i,j-1}|. \quad (8)$$

Considering both absolute extreme values and significant changes, we define the t -score based on historical data:

$$t_{i,j} = a_t \cdot |z_{i,j}| + b_t \cdot |\Delta v_{i,j}| \quad (9)$$

where a_t and b_t are used to tradeoff Z-score and Δv_{ij}

To maintain numerical stability in subsequent calculations, we normalize $t_{i,j}$. Let t_{\min} and t_{\max} be the minimum and maximum t-scores within the timeslot, respectively

$$t_{i,j}^* = \frac{t_{i,j} - t_{\min}}{t_{\max} - t_{\min}}. \quad (10)$$

Population Demand Perspective: In SMCS, the significance of different sensing areas varies across tasks. For instance, in tasks, such as traffic flow monitoring, greater attention is typically given to busy urban centers, while tasks like wildfire detection prioritize forested regions. Deploying workers to these critical areas allows for more accurate data collection, thereby improving service provision. The variation in the importance of different regions is referred to as “population demand” in our work.

In many tasks (e.g., monitoring PM2.5, temperature, and humidity), population density can serve as an indicator of population demand. Regions with higher population density should be given higher weights. This is because providing better services in high-density areas benefits more people compared to low-density areas, thereby improving the overall impact and efficiency of the service. In this article, to make the discussion more general, we use population density as population demand. For each cell, $h_{i,j}$ represents the proportion of the population in that cell relative to the total population size, known as the H-score. Similarly, considering numerical stability, we normalize $h_{i,j}$ to obtain $h_{i,j}^*$:

$$h_{i,j}^* = \frac{h_{i,j} - h_{\min}}{h_{\max} - h_{\min}} \quad (11)$$

where h_{\min} and h_{\max} represent the minimum and maximum H-scores within the timeslot.

It is noteworthy that this article primarily employs population density as the principal indicator of population demand. This choice is motivated by the intent to encompass a wider range of common sensing tasks and to enhance the generalizability of the discussion. Our framework can be readily extended to other sensing scenarios, such as forest fire monitoring or wildlife tracking. In these cases, the indicator reflecting the degree of concern can be constructed to represent population demand. For instance, in forest fire monitoring, historical data could be utilized to assign higher population demand values to areas with a documented history of frequent fire outbreaks. This adaptation enables the algorithm to achieve its objectives effectively, ensuring high-quality sensing (e.g., timely detection of fire incidents) at a reduced cost.

Combined Consideration: By integrating the above two factors, we define regional population demand $S_{i,j}$ to comprehensively consider the influence of historical data and population density. The specific definition is as follows, where w_1 and w_2 are the weights corresponding to each factor

$$S_{i,j} = w_1 \cdot h_{i,j}^* + w_2 \cdot t_{i,j}^*. \quad (12)$$

Similarly, we need to perform normalization to ensure the stability of the values

$$S_{i,j}^* = \frac{S_{i,j} - S_{\min}}{S_{\max} - S_{\min}}. \quad (13)$$

Although $S_{i,j}^*$ effectively measures the importance among different regions, the absolute score differences are not substantial enough to reflect significant disparities in calculations. Therefore, we employ the Gompertz function to amplify the score differences among regions

$$R_{i,j}^* = a \cdot \exp\left(-b \cdot \exp\left(-c \cdot S_{i,j}^*\right)\right). \quad (14)$$

Due to the sparsity of sensing data, the computed $R_{i,j}^*$ inevitably contains missing values, denoted as $\tilde{R}_{i,j}^*$. To reasonably fill these missing values, our imputation strategy relies on two factors: temporal-spatial continuity and window length.

- 1) **Temporal-Spatial Continuity:** For a given task, Regional Population Demand exhibits continuity across time and space. Temporally, demand variations transition gradually rather than abruptly within individual regions, ensuring local smoothness along the temporal dimension. Spatially, geographically adjacent regions show demand interdependence owing to shared urban functions and population mobility patterns. From this, it can be concluded that temporally and spatially adjacent values each contain partial information about the missing value. Therefore, we integrate temporal (T) and spatial (S) data from adjacent timeslots and regions to facilitate imputation.
- 2) **Window Length:** The window length specifies the extent of surrounding nonmissing values used for imputation. Here, parameter k represents the number of timeslots considered in the temporal dimension, while parameter h represents the number of neighboring regions in the spatial dimension. A large window length may overly broaden the averaging process, driving imputed values toward the global mean and reducing precision. Conversely, a small window length can make imputed values too sensitive to local fluctuations.

To account for the distinctions between *offline* and *online* scenarios, the methods and parameters for computing missing values are adapted accordingly. In the following, we elaborate on the specific imputation approaches for each scenario separately.

Offline Update Phase: We compute the fill-in value $\tilde{R}_{i,j}^*$ for region i at timeslot j using the following formula:

$$\tilde{R}_{i,j}^* = \frac{1}{k+h} \left(\sum_{\substack{m=-\frac{k}{2} \\ m \neq 0}}^{\frac{k}{2}} T_{i,j+m} + \sum_{p=1}^h S_{ip,j} \right). \quad (15)$$

Here, $T_{i,j+m}$ denotes the m th nearest nonmissing value in the temporal dimension for region i offline, where $m > 0$ indicates future and $m < 0$ indicates past. For simplicity, we select $k/2$ nonmissing values before and after j as temporal approximations once k is fixed. Meanwhile, $S_{ip,j}$ represents the p th nearest spatial neighbor of region i , with larger p indicating greater distance.

As mentioned earlier, we comprehensively utilize both temporal and spatial information by averaging the approximations to impute missing values. The precision of the imputed data remains consistent with the original data. Through analysis

and testing on the dataset, we found that $k = 4$ and $h = 4$ generally yield good imputation results. Therefore, this parameter combination is adopted for offline scenario imputation in this article.

Online Update Phase: Compared to the offline scenario, since future data is unavailable, we adjust the fill-in method to rely solely on past data

$$\tilde{R}_{i,j}^* = \frac{1}{k+h} \left(\sum_{m=-k}^{-1} T_{i,j+m} + \sum_{p=1}^h S_{i,p,j} \right). \quad (16)$$

In the online scenario, the definitions of $T_{i,j+m}$ and $S_{i,p,j}$ remain the same as in the offline setting. However, since future data relative to timeslot j is unavailable, m can only take values up to -1 , where $m = -1$ represents the most recent nonmissing value in the past for region i .

Unlike the offline scenario, the online scenario requires a more sensitive response to data changes to promptly adapt to sudden events. This is reflected in the parameter selection, where a smaller k is chosen. Through analysis and testing on the dataset, we found that $k = 1$ and $h = 4$ generally yield good performance. Hence, we adopt this parameter set for online scenario imputation in this article.

By following the above procedure, we obtain the importance matrix for each region regarding service quality, also known as the demand matrix R

$$R_{i,j} = \begin{cases} R_{i,j}^*, & \text{if } S[i,j] = 1 \\ \tilde{R}_{i,j}^*, & \text{if } S[i,j] = 0 \end{cases}. \quad (17)$$

B. Dynamic Clustering Module

The dynamic clustering module is an essential component of the D2-SMCS framework. It addresses the problem of how to most effectively select sensing regions to maximize completion accuracy after obtaining the demand matrix R . As stipulated by Assumption 1, which posits that workers or sensing devices can reliably complete their assigned tasks within each timeslot, the identification of sensing regions directly dictates how the platform allocates and publishes sensing tasks. Traditional random methods may result in selecting regions that are overly concentrated, leading to insufficient information in other areas and causing data redundancy. The dynamic clustering module divides different clusters by combining geographical proximity and numerical features. The basic idea is to group together regions that are geographically close in real space and exhibit similar characteristics in historical data. Clustering stops when the sum of demand $R_{i,j}^*$ within the clustered region reaches the threshold T which is set to limit the number of cells in a cluster and prevent excessive cells from affecting sensing. Only one cell is selected for sensing in each cluster region within each timeslot.

To measure the geographical distance between different regions, we define the distance between region x and region y as $\text{dist}_{x,y}$

$$\text{dist}_{x,y} = \text{Haversine}(\text{lat}_x, \text{lon}_x, \text{lat}_y, \text{lon}_y). \quad (18)$$

The Haversine formula is a commonly used method for calculating geographical distances [37], which computes the

shortest path distance between two points on the Earth's surface based on their latitudes and longitudes. Next, to capture the distance between different regions in the feature space, we introduce an feature matrix \mathbf{F}

$$\mathbf{F} = \mathbf{D}_{[:, t-d+1:t]}. \quad (19)$$

Here, $\mathbf{D} \in \mathbb{R}^{n \times m}$ represents the sensing data matrix, containing observations of n regions over m time slots. $\mathbf{F} \in \mathbb{R}^{n \times d}$ is the feature matrix, containing the most recent d columns (time slots) of observation data for each region, used to capture the similarity between regions in the feature space. t is the column index of the current timeslot, i.e., the position of the latest timeslot. The introduction of the feature matrix \mathbf{F} provides dynamic change information of each region in recent time slots for the clustering process, so that the distance between different regions in the feature space can reflect the recent characteristics of their sensing data.

The dynamic clustering module is based on intuitive geographical distance clustering and modifies the results through feature space distances to ultimately obtain the clustering results and the total number of clusters. Next, we explain the details of the clustering algorithm.

As shown in Algorithm 1, we initialize the clustering by setting up a processing flag array \mathbf{P} to track cell assignments and creating an empty cluster set \mathcal{C} . A nearest neighbor model is trained on the geographical coordinate matrix \mathbf{C} for efficient neighbor identification. During cluster construction, each unprocessed cell i in the demand matrix \mathbf{R} initializes a new cluster C_i and is marked as processed. Neighboring unprocessed cells j are iteratively added to C_i , accumulating their demand values into the total demand sum S until S reaches the threshold T or no further cells can be added. The resulting cluster is then added to \mathcal{C} .

Subsequently, outlier detection and reallocation utilize the feature matrix \mathbf{F} by computing each cluster's centroid μ_i and determining the feature space distance d_k of cell k to its centroid (lines 10–11). An outlier threshold τ_i is set based on the mean and standard deviation of the cell-to-centroid distances, and cells with distances exceeding τ_i are reassigned to their closest centroid's cluster to enhance compactness (lines 12–14). If the sparsity ratio α is provided, sparsity control part adjusts the number of clusters to the target $t = (1 - \alpha) \cdot \text{num}$ by splitting the largest clusters or merging the smallest ones as necessary (lines 16–26). The algorithm concludes by outputting the final cluster set \mathcal{C} and the total number of clusters $V = |\mathcal{C}|$, effectively combining geographical distance and feature space similarity to define distinct cluster regions.

C. NALMC Data Reconstruction

In this section, we will first introduce the basic concepts of latent factor model completion and neighborhood completion, and then elaborate on the improvements we have made on the basic matrix completion to enhance the QoS.

1) *Latent Factor Model Data Completion:* In real-world scenarios, there is often inherent correlation among sensing data, which allows the entire data matrix to be effectively approximated by a low-rank matrix. Given an incomplete

Algorithm 1: Dynamic Clustering Algorithm

Input : Demand Matrix $\mathbf{R} \in \mathbb{R}^{n \times 1}$, Latitude-longitude coordinates $\mathbf{C} \in \mathbb{R}^{n \times 2}$, Max total demand threshold per cluster T , Feature matrix $\mathbf{F} \in \mathbb{R}^{n \times d}$, Sparsity ratio α , Nonzero cell count num

Output: Clusters $\mathcal{C} = \{C_1, C_2, \dots, C_K\}$, Number of clusters V

- 1 **Initialize** processed flags $\mathbf{P} \leftarrow \mathbf{0}_n$, empty clusters list \mathcal{C}
- 2 **Train** nearest neighbor model NN on \mathbf{C} using Haversine distance
- 3 **foreach** *unprocessed cell* i in \mathbf{R} **do**
- 4 Initialize new cluster $C_i = \{i\}$ and sum $S \leftarrow \mathbf{R}[i]$
- 5 **while** $S < T$ and *unprocessed neighbors exist* **do**
- 6 Add nearest unprocessed neighbor j to C_i , update $S \leftarrow S + \mathbf{R}[j]$
- 7 $\mathbf{P}[j] \leftarrow 1$ # Mark j as processed
- 8 add C_i to \mathcal{C}
- 9 **foreach** cluster $C_i \in \mathcal{C}$ **do**
- 10 Compute cluster centroid in feature space
- 11 $\mu_i = \frac{1}{|C_i|} \sum_{k \in C_i} \mathbf{F}[k, :]$
- 12 Compute mean and std of cell-centroid distances
- 13 $\bar{d}_i \leftarrow \frac{1}{|C_i|} \sum_{k \in C_i} \|\mathbf{F}[k, :] - \mu_i\|_2$,
- 14 $\sigma_i \leftarrow \sqrt{\frac{1}{|C_i|} \sum_{k \in C_i} (d_k - \bar{d}_i)^2}$
- 15 Set outlier threshold $\tau_i \leftarrow \bar{d}_i + 1.5 \cdot \sigma_i$
- 16 **foreach** outlier $k \in C_i$ with $d_k > \tau_i$ **do**
- 17 Find closest cluster j with centroid μ_j using $j = \arg \min_{j \neq i} \|\mathbf{F}[k, :] - \mu_j\|_2$
- 18 Transfer k to C_j
- 19 Set target cluster count $t \leftarrow (1 - \alpha) \cdot num$;
- 20 **while** $|\mathcal{C}| \neq t$ **do**
- 21 **if** $|\mathcal{C}| < t$ **then**
- 22 Sort clusters \mathcal{C} in descending order by size;
- 23 Select largest cluster C_{\max} in \mathcal{C} ;
- 24 Split C_{\max} into smaller clusters;
- 25 Update \mathcal{C} and continue until $|\mathcal{C}| \geq t$;
- 26 **else if** $|\mathcal{C}| > t$ **then**
- 27 Sort clusters \mathcal{C} in ascending order by size;
- 28 Select smallest clusters $C_{\min 1}$ and $C_{\min 2}$;
- 29 Merge $C_{\min 1}$ and $C_{\min 2}$;
- 30 Update \mathcal{C} and continue until $|\mathcal{C}| \leq t$;
- 31 **return** \mathcal{C} , $V = |\mathcal{C}|$

sensing data matrix D , we can utilize its low-rank property to reconstruct the complete sensing matrix \hat{M} . This reconstruction problem can be formulated as the following optimization objective:

$$\min \text{rank}(\hat{M}), \quad \text{s.t.}, \quad \hat{M} \circ S = D \quad (20)$$

where \circ denotes element-wise multiplication of matrices, \hat{M} is the complete matrix to be reconstructed, and S is the selection matrix indicating the observed entries in the sensing data matrix. However, this optimization problem is nonconvex, making direct solving difficult. Therefore, we decompose the

reconstruction matrix $\hat{M}_{N \times K}$ into the product of two smaller matrices, i.e., $\hat{M} = U_{N \times r} V_{K \times r}^T$, where U and V are the latent feature matrices in space and time, respectively, and r is the latent rank.

In the case where the matrix has low-rank characteristics [38], minimizing the rank of \hat{M} is equivalent to minimizing $\|U\|_F^2 + \|V\|_F^2$. Due to the temporal and spatial continuity of SMCS tasks, their sensing matrices are often low-rank. Therefore, the above optimization objective can be reformulated as

$$\min \|(D - UV^T) \circ S\|_F^2 + \lambda (\|U\|_F^2 + \|V\|_F^2) \quad (21)$$

where $\|(D - UV^T) \circ S\|_F^2$ represents the reconstruction error based on the Frobenius norm, and λ is a balancing parameter used to tradeoff between rank minimization and reconstruction accuracy.

2) *Neighborhood Model Data Completion*: Unlike the latent factor model, the neighborhood model for data completion focuses on filling missing data based on the similarity between regions. Specifically, the neighborhood model assumes that regions that are spatially adjacent or have similar features are more likely to exhibit similar data patterns. Therefore, when completing missing data, we first define the “neighborhood” of each region through distance or feature similarity measures, and then use the observed data in the neighborhood to infer the missing values of the target region.

Let \mathcal{N}_i be the neighborhood set of region i . For a missing value $D[i, j]$ in matrix D , the neighborhood model completes it based on the observed data in \mathcal{N}_i , which can be expressed as a weighted average

$$\hat{D}[i, j] = \frac{\sum_{k \in \mathcal{N}_i} w_{ik} D[k, j]}{\sum_{k \in \mathcal{N}_i} w_{ik}} \quad (22)$$

where w_{ik} denotes the similarity weight between region i and its neighbor k . The weights w_{ik} can be computed based on geographical distance, feature similarity, or other correlation measures. In this way, the neighborhood model leverages spatial proximity information, enabling the completion results to better reflect the characteristics and trends of local data.

3) *Latent Factor Model With Spatiotemporal and Neighborhood Constraints*: The formula (21) is merely the basic model for latent factor model, utilizing only the most basic global low-rank information for completion. However, the environmental data we use often have strong spatiotemporal correlations. By leveraging this property, we can better mine the implicit information in the data. After adding spatiotemporal constraints and neighborhood constraints, the objective function becomes

$$\min_{U, V} \|D - UV^T\|_F^2 + \lambda (\|U\|_F^2 + \|V\|_F^2) + \lambda_t \cdot \mathcal{T}(V) + \lambda_s \cdot \mathcal{S}(U) \quad (23)$$

where $\mathcal{T}(V)$ and $\mathcal{S}(U)$ are the temporal constraint function and spatial constraint function, respectively, with λ_t and λ_s as the corresponding regularization parameters controlling the magnitude of the regularization terms.

a) *Temporal smoothness constraint term $\mathcal{T}(V)$* : The temporal smoothness constraint term $\mathcal{T}(V)$ is used to capture the continuity between adjacent time points. We achieve this by multiplying the transpose of the item feature matrix V with a Toeplitz matrix T . The Toeplitz matrix $T \in \mathbb{R}^{K \times K}$ is a special matrix designed to impose constraints on adjacent points in the time dimension, making the feature changes between adjacent time points tend to be smooth. This term is specifically defined as

$$\mathcal{T}(V) = \|V^T T\|_F^2. \quad (24)$$

Minimizing $\|V^T T\|_F^2$ encourages smaller differences between feature vectors at adjacent time points, ensuring smoothness in the time dimension.

b) *Spatial smoothness constraint term $\mathcal{S}(U)$* : The spatial smoothness constraint term $\mathcal{S}(U)$ is achieved by utilizing the spatial distances between regions. There exists a spatial matrix P between users, which reflects the geographical adjacency relations between regions. It is constructed based on the geographical locations (latitude and longitude) of the center points of each region, using the distances between these locations to measure interregional similarity. Specifically, for N regions, we define a spatial similarity matrix $P \in \mathbb{R}^{N \times N}$, where each element $P[i, j]$ represents the similarity weight between region i and region j .

Each element of matrix P is calculated through the following steps. First, the geographical distance d_{ij} between region i and region j is computed using the Haversine formula [37]. Then, for each pair of regions i and j , the distance d_{ij} is mapped to a similarity weight $P[i, j]$ through an exponential transformation, with the calculation formula:

$$P[i, j] = \begin{cases} e^{-\frac{d_{ij}}{\sigma_c}}, & \text{if } i \neq j \\ 0, & \text{if } i = j \end{cases} \quad (25)$$

where σ_c is a tuning parameter that controls the rate at which similarity weights decay with increasing geographical distance. A larger σ_c reduces the decay rate, enhancing connections between distant regions, whereas a smaller one emphasizes local similarities, impacting the performance of data reconstruction. Therefore, in practical sensing scenarios, the value of σ_c must align with the characteristics of the dataset used. For applications where distant regions still exhibit strong correlations, a larger σ_c is preferred; conversely, a smaller one is suitable. In this study, we employ multiple datasets with significant variations in interregion distances. To control variables and ensure the order of magnitude of d_{ij} matches σ_c^2 , we uniformly adopt $\sigma_c = 5$ across all experiments in this article. The exponential transformation causes the similarity weight to gradually decrease as the geographical distance between regions increases. The diagonal elements (i.e., when $i = j$) are set to zero to avoid the impact of self-similarity on matrix calculations. The spatial similarity matrix P maps geographical distances to similarity weights, making adjacent regions have higher weights.

The spatial smoothness constraint term is introduced to make cells that are closer in distance tend to have more similar

values, while the constraint on cells that are farther apart becomes weaker. Therefore, it is defined as follows:

$$\mathcal{S}(U) = \|PU\|_F^2. \quad (26)$$

c) *Neighborhood constraint terms*: We introduce neighborhood constraint terms to encourage similar feature vectors in the latent matrices U and V to be closer. The neighborhood constraint terms exploit the local correlations of latent features, enabling the model to better capture local patterns in the data. Specifically, the neighborhood constraint term comprises two parts, acting on similar feature pairs in latent matrices U and V , respectively.

The neighborhood constraint on latent matrix U is defined as

$$\gamma_u \sum_{u=1}^N \sum_{j \in \mathcal{N}_u} w_{uj} \|U_u - U_j\|_F^2 \quad (27)$$

where \mathcal{N}_u denotes the set of the k most similar neighbors to u in latent matrix U , and w_{uj} represents the similarity weight between the u th and j th rows of latent matrix U . By minimizing the differences between adjacent feature vectors, the model constrains the feature differences among neighboring units in latent matrix U , thereby improving smoothness in the spatial dimension.

Similarly, the neighborhood constraint on latent matrix V is defined as

$$\gamma_v \sum_{i=1}^K \sum_{k \in \mathcal{N}_i} w_{ik} \|V_i - V_k\|_F^2 \quad (28)$$

where \mathcal{N}_i is the set of the k most similar neighbors to i in latent matrix V . This term ensures continuity and consistency in the time dimension by minimizing the differences between adjacent feature vectors in latent matrix V .

To compute the aforementioned similarity weights w , we construct a similarity matrix W using the column vectors of the latent matrix X , where X can be the spatial latent matrix $U \in \mathbb{R}^{N \times r}$ or the temporal latent matrix $V \in \mathbb{R}^{K \times r}$. Here, the column vectors represent the feature values of each spatial region or time point in different latent dimensions. Therefore, the elements $W[u, j]$ of the similarity matrix $W \in \mathbb{R}^{r \times r}$ represent the similarity between the u th and j th columns of the latent matrix X .

To compute the similarity matrix W , we use cosine similarity as the similarity measure. Cosine similarity measures the directional similarity between two vectors. The specific calculation process is as follows. First, we perform mean centering on the columns of the latent matrix X to eliminate biases between different features. The centered column vectors are expressed as

$$\tilde{X}[:, u] = X[:, u] - \frac{1}{L} \sum_{j=1}^L X[j, u] \quad (29)$$

where $\tilde{X}[:, u]$ represents the u th column of matrix X after centering, and L is the number of rows of the latent matrix, i.e., $L = N$ when $X = U$, or $L = K$ when $X = V$. Next, the

elements $W[u, j]$ in the similarity matrix W are calculated by the following formula:

$$W[u, j] = \frac{\tilde{X}[:, u]^\top \tilde{X}[:, j]}{\|\tilde{X}[:, u]\|_F \cdot \|\tilde{X}[:, j]\|_F}. \quad (30)$$

The values $W[u, j]$ obtained through cosine similarity computation range between $[-1, 1]$, where values closer to 1 indicate that the two vectors have more similar directions and higher similarity.

After adding the neighborhood constraint terms, the final objective function becomes

$$\begin{aligned} \min_{U, V} & \|R - UV^\top\|_F^2 + \lambda (\|U\|_F^2 + \|V\|_F^2) + \lambda_t \cdot \mathcal{T}(V) + \lambda_s \cdot \mathcal{S}(U) \\ & + \gamma_u \sum_{u=1}^N \sum_{j \in \mathcal{N}_u} w_{uj} \|U_u - U_j\|_F^2 + \gamma_v \sum_{i=1}^K \sum_{k \in \mathcal{N}_i} w_{ik} \|V_i - V_k\|_F^2. \end{aligned} \quad (31)$$

This objective function adds four regularization constraints to the basic low-rank decomposition term, namely, the temporal smoothness term, the spatial smoothness term, and the neighborhood constraint terms for the latent matrices. The temporal smoothness constraint term $\lambda_t \cdot \mathcal{T}(V)$ ensures that feature changes at adjacent time points remain smooth, thereby capturing similarity in the time dimension. The spatial smoothness constraint term $\lambda_s \cdot \mathcal{S}(U)$ ensures consistency in the spatial dimension through geographical similarity between regions. In addition, the neighborhood constraint terms $\gamma_u \sum_{u=1}^N \sum_{j \in \mathcal{N}_u} w_{uj} \|U_u - U_j\|_F^2$ and $\gamma_v \sum_{i=1}^K \sum_{k \in \mathcal{N}_i} w_{ik} \|V_i - V_k\|_F^2$ further ensure the closeness of similar features in the latent matrices U and V . By leveraging the relationships between similar feature vectors, the model becomes more robust at the local level.

Neighborhood-Aware Latent Matrix Completion: To effectively solve the above optimization problem that includes spatiotemporal constraints and neighborhood constraints, and to enable both offline updates and online incremental updates, we adopt the ALS algorithm. The basic principle of ALS is to decompose a complex bivariate optimization problem into two univariate optimization subproblems. In each iteration, we fix one latent matrix (e.g., V) and treat the objective function as a function of the other latent matrix (e.g., U), solving the minimization problem with respect to U . Then, we fix the updated U and solve the minimization problem with respect to V . Through this alternating update method, the algorithm gradually approaches a global or local optimal solution. Moreover, when new data arrives, only two vectors in the latent matrices U and V need to be updated, without recomputing the entire matrix factorization. This localized update significantly improves computational efficiency, allowing the model to quickly handle dynamically changing data and enabling online updates.

Next, to update the latent matrices U and V , we provide the corresponding update formulas by solving the closed-form solutions.

Updating U While Fixing V : We take the derivative of the objective function L with respect to U

$$\begin{aligned} \frac{\partial L}{\partial U} &= \frac{\partial}{\partial U} \left(\|D - UV^\top\|_F^2 + \lambda \|U\|_F^2 \right. \\ &\quad \left. + \lambda_s \mathcal{S}(U) + \gamma_u \sum_{u=1}^N \sum_{j \in \mathcal{N}_u} w_{uj} \|U_u - U_j\|_F^2 \right). \end{aligned} \quad (32)$$

Setting the derivative to zero, we obtain

$$\begin{aligned} U_u &= \left(\sum_{i \in \mathcal{S}_u} V_i V_i^\top + \left(\lambda + \gamma_u \sum_{j \in \mathcal{N}_u} w_{uj} \right) I + \lambda_s (S^\top S)_{uu} \right)^{-1} \\ &\quad \times \left(\sum_{i \in \mathcal{S}_u} V_i D_{ui} + \gamma_u \sum_{j \in \mathcal{N}_u} w_{uj} U_j - \lambda_s (S^\top S U)_u \right). \end{aligned} \quad (33)$$

Here, U_u is the u -th column of the latent matrix U .

Updating V While Fixing U : Similarly, we take the derivative of the objective function L with respect to V

$$\begin{aligned} \frac{\partial L}{\partial V} &= \frac{\partial}{\partial V} \left(\|D - UV^\top\|_F^2 + \lambda \|V\|_F^2 \right. \\ &\quad \left. + \lambda_t \mathcal{T}(V) + \gamma_v \sum_{i=1}^K \sum_{k \in \mathcal{N}_i} w_{ik} \|V_i - V_k\|_F^2 \right). \end{aligned} \quad (34)$$

Setting the derivative to zero, we obtain

$$\begin{aligned} V_i &= \left(\sum_{u \in \mathcal{S}_i} U_u U_u^\top + \left(\lambda + \gamma_v \sum_{k \in \mathcal{N}_i} w_{ik} \right) I + \lambda_t T T^\top \right)^{-1} \\ &\quad \times \left(\sum_{u \in \mathcal{S}_i} U_u D_{ui} + \gamma_v \sum_{k \in \mathcal{N}_i} w_{ik} V_k - \lambda_t (V T T^\top)_i \right). \end{aligned} \quad (35)$$

Thus, we have derived how to update the latent matrices U and V based on new incoming information. The above update equations initially incorporate neighborhood factors into the latent factor model completion. However, in practical experiments, constrained by the global low-rank completion principle of the latent factor matrix itself, there is still room for improvement in incorporating neighborhood factors, even after adding neighborhood terms. Therefore, compared to other matrix completion works that rely solely on matrix completion methods for data inference, we combine the improved ALS matrix completion with the KNN method to form NALMC. The principle is to extract a ratio ϕ of the sensing data in the current timeslot as training data to update the latent matrices U and V and use the model to complete the remaining data. Simultaneously, we use the KNN method to perform completion under the same information conditions. The final completion result is a combination of the improved ALS model completion result and the KNN model completion result, weighted by $(1 - \rho)$ and ρ , respectively. The remaining $(1 - \phi)$ of the sensing information is used as test data to determine an optimal value of ρ . Using the resulting timeslot-specific ρ^* , we retrain the improved ALS model with the complete sparse sensing data and generate the final completion. The detailed process is summarized in pseudocode of Algorithm 2.

Algorithm 2: NALMC

Input : Sparse data matrix $\mathbf{D} \in \mathbb{R}^{N \times K}$; Selection matrix $\mathbf{S} \in \{0, 1\}^{N \times K}$; training-validation split ratio ϕ ; regularization parameters $\lambda, \lambda_t, \lambda_s$; neighborhood constraint regularization parameter γ_u, γ_v ; neighborhood size $k \in \mathbb{Z}^+$ (number of nearest neighbors)

Output: Reconstructed complete matrix $\hat{\mathbf{M}} \in \mathbb{R}^{N \times K}$

- 1 **Define** observed indices: $\Omega = \{(u, i) \mid \mathbf{S}_{ui} = 1\}$
- 2 **Split** Ω into training set Ω_{train} and validation set Ω_{val} , where $|\Omega_{\text{train}}| = \phi|\Omega|$
- 3 **Initialize** latent factors: $\mathbf{U} \in \mathbb{R}^{N \times r}$ and $\mathbf{V} \in \mathbb{R}^{K \times r}$
- 4 **repeat**
- 5 # Alternating Least Squares (ALS) with Neighborhood Constraints
- 6 **Fix** \mathbf{V} , update \mathbf{U} :
- 7 **for** $u = 1$ **to** N **do**
- 8 Find k nearest neighbors \mathcal{N}_u of user u
- 9 Update \mathbf{U}_u using $\mathcal{N}_u, \lambda, \lambda_s$ and γ_u
- 10 **Fix** \mathbf{U} , update \mathbf{V} :
- 11 **for** $i = 1$ **to** K **do**
- 12 Find k nearest neighbors \mathcal{N}_i of item i
- 13 Update \mathbf{V}_i using $\mathcal{N}_i, \lambda, \lambda_t$ and γ_v
- 14 **until** convergence;
- 15 Compute $\hat{\mathbf{M}}_{\text{ALS}} = \mathbf{U}\mathbf{V}^\top$
- 16 Compute $\hat{\mathbf{M}}_{\text{KNN}}$ using Ω_{train}
- 17 **Find** optimal blending parameter ρ^* :
- 18 **begin**
- 19 **Initialize** ρ with an initial value
- 20 **repeat**
- 21 Compute blended reconstruction:
 $\hat{\mathbf{M}}(\rho) = (1 - \rho)\hat{\mathbf{M}}_{\text{ALS}} + \rho\hat{\mathbf{M}}_{\text{KNN}}$
- 22 Calculate error $E(\rho)$ on validation set Ω_{val}
- 23 Update ρ to reduce $E(\rho)$
- 24 **until** convergence;
- 25 Set ρ^* to the value of ρ that minimizes $E(\rho)$
- 26 **Retrain** \mathbf{U} and \mathbf{V} using all data Ω , repeat ALS updates until convergence
- 27 Recompute $\hat{\mathbf{M}}_{\text{ALS}}$ and $\hat{\mathbf{M}}_{\text{KNN}}$ using all data
- 28 **Set** final reconstruction:
- 29 $\hat{\mathbf{M}} = (1 - \rho^*)\hat{\mathbf{M}}_{\text{ALS}} + \rho^*\hat{\mathbf{M}}_{\text{KNN}}$
- 30 **return** $\hat{\mathbf{M}}$

V. PERFORMANCE EVALUATION

In this section, we provide an overview of two typical datasets and the latest prominent or popular baseline methods in sparse data inference. Subsequently, we present a detailed performance evaluation of our approach on each dataset.

A. Datasets

To evaluate the data completion problem of crowd service quality that we proposed, we applied famous and popular

urban crowdsensing datasets, including Sensor-Scope [39] and U-Air [40]. The Sensor-Scope dataset comprises various typical sensing data pertinent to urban environments, including measurements of humidity and temperature. In contrast, U-Air gathers sensing readings from urban air quality monitoring systems. These datasets, although gathered through static sensors, could just as effectively be collected and processed using mobile devices. The chosen tasks, such as temperature, PM2.5 and CO monitoring, exemplify typical urban crowdsensing applications, making these datasets ideal for evaluating the effectiveness of our proposed questions and solution. It is important to clarify that, per Assumption 2, we assume all data within these datasets to be reliable and free from distortions, such as sensor noise or interference from malicious users.

Additionally, to assess crowd service quality, we constructed a population density dataset for Beijing, segmented by time intervals, based on real-world, grid-based population flow data provided by Datafountain [41] and aligned with the U-Air monitored regions. Using the distribution characteristics of this dataset, we further generated a synthetic population density dataset for Sensor-Scope, also segmented by time intervals. The detailed description of the data portions used from datasets is presented in Table II.

B. Baselines and Measures

- 1) *Baselines*: To fully leverage sparse sensing data for data inference and enhance QoS in crowd applications, we introduce the NALMC data inference algorithm. This approach is evaluated against various data inference algorithms.
 - a) *KNN* [42] is a nonparametric algorithm that fits data by predicting outcomes based on the average value of its nearest neighbors, effectively capturing local patterns in the feature space.
 - b) *GP* [43] leverages covariance functions to model complex data relationships, capturing spatial and temporal dependencies for interpolation.
 - c) *KDE* [44] is a technique for estimating the probability density function of a variable by applying a smoothing function to data points, resulting in a continuous density curve that captures the underlying distribution.
 - d) *ALS-ST* [18] is an enhanced version of ALS that incorporates spatiotemporal factors, allowing for more effective modeling of data that varies across space and time, making it suitable for applications like collaborative filtering and data completion.
 - e) *SAITS* [45] applies pure self-attention for time series imputation, effectively capturing temporal dependencies to achieve accurate data completion without recursive structures.
 - f) *EADNMF* [46] leverages adversarial training with an adaptive elastic loss that interpolates between Frobenius and other norms to handle complex data structures in matrix completion.

Key parameters for baseline methods were selected as follows. KNN (neighbors = 3), GP (RBF kernel, length scale = 1.0, noise = 0.01), and KDE (bandwidth = 0.5)

TABLE II
STATISTICS OF EVALUATION DATASETS

Data City Subarea Cycle & Duration Mean Std. Unit Data Source	Environmental Monitoring					Population Data
	Humidity	PM2.5	PM10	CO	O ₃	Crowd Density
	Lausanne (Switzerland)	Beijing (China)				Lausanne & Beijing
	51 subareas each with $50 \times 30 \text{ m}^2$	36 subareas each with $1000 \times 1000 \text{ m}^2$				Matches environmental data
	10 min & 16 d	1 h & 94 d				1 h & 24 h
	81.99	64.44	108.42	0.81	90.77	1.96 & 2.77
Std.	8.26	52.90	73.01	0.53	64.46	3.64 & 2.96
	%	$\mu\text{g}/\text{m}^3$	$\mu\text{g}/\text{m}^3$	mg/m^3	$\mu\text{g}/\text{m}^3$	%
	SensorScope	U-Air				DataFountain

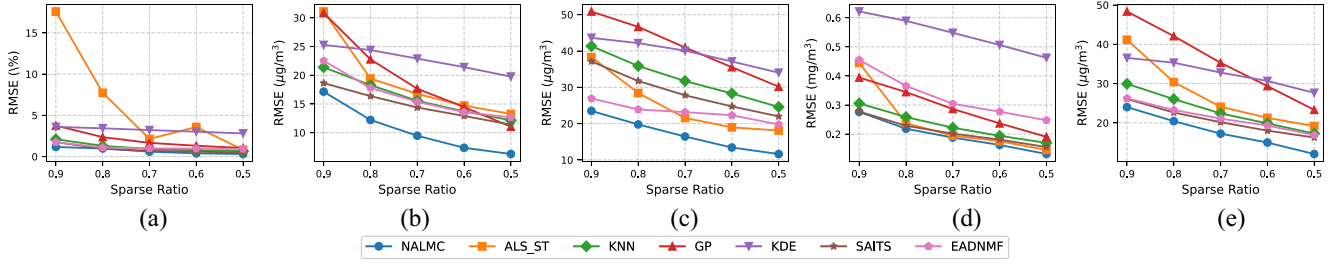


Fig. 3. RMSE of completion results with different sparse ratios (offline). (a) Humidity. (b) PM2.5. (c) PM10. (d) CO. (e) O₃.

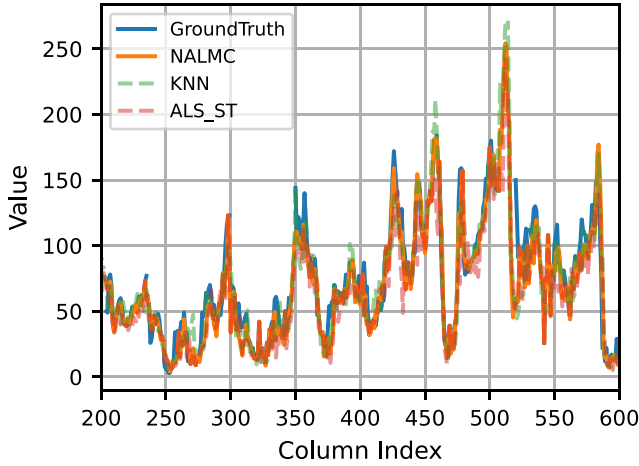


Fig. 4. PM2.5 at sparse ratio 0.2 versus ground truth.

were tuned via grid search; ALS-ST's latent rank aligns with NALMC's r in this article; SAITS and EADNMF use optimal values from related works.

2) *Measures*: In the following experiments, we mainly evaluate our work from the following two perspectives.

- Root mean square error (RMSE)* measures prediction accuracy by calculating the square root of the average squared differences between predicted and actual values
- QoS (Q)* is evaluated by calculating the Epsilon for each completed region, defined as the ratio of completion error to the original value. This is then weighted by the population density index to produce the final QoS measure.

C. Evaluation Results

1) *Offline Data Completion*: In offline data completion experiments, we extracted 2000 timeslots from each

dataset for evaluation. The proportion of sensed data within each timeslot varies depending on the sparse ratio applied. We evaluated completion accuracy across five dataset tasks using the RMSE metric. As illustrated in Figs. 3 and 5, the performance of each method fluctuates across datasets, with these variations strongly linked to the distinct characteristics of each dataset. For example, the ALS-ST method, a latent matrix completion technique, captures global information effectively but is less adept at capturing local features. This makes it particularly suitable for datasets with lower variability, such as the CO dataset in Fig. 3(d), where ALS-ST demonstrates low RMSE at lower sparsity. However, as sparsity increases, the method struggles to capture global trends accurately, leading to a sharp rise in RMSE. At lower sparsity, ALS-ST consistently exhibits relatively high RMSE, especially in datasets with significant fluctuations, as seen in Fig. 3(b). In contrast, KNN—a neighborhood-based completion method—focuses on capturing local information. This ensures a baseline level of completion accuracy regardless of dataset variability. Even in datasets with pronounced peaks, KNN can swiftly adapt to trend changes by leveraging similar values within the feature space, achieving robust performance across all datasets. However, its exclusive reliance on local information imposes a ceiling on performance, as it lacks the capability to capture global trends comprehensively. NALMC combines the strengths of both latent matrix completion and neighborhood-based completion, enhancing completion accuracy while achieving exceptional stability. Across the 25 test points in Q (five datasets with five sparsity levels each), NALMC achieved optimal results in 24 cases, significantly boosting completion stability compared to other methods. To illustrate this, we selected

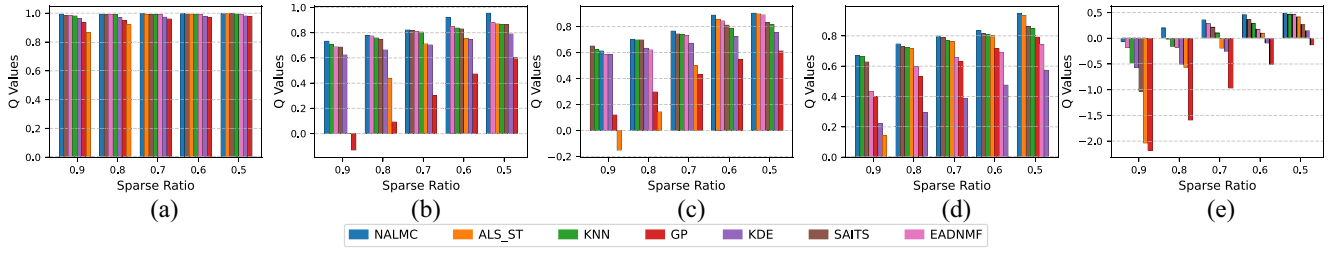


Fig. 5. Q of completion results with different sparse ratios (offline). (a) Humidity. (b) PM2.5. (c) PM10. (d) CO. (e) O₃.

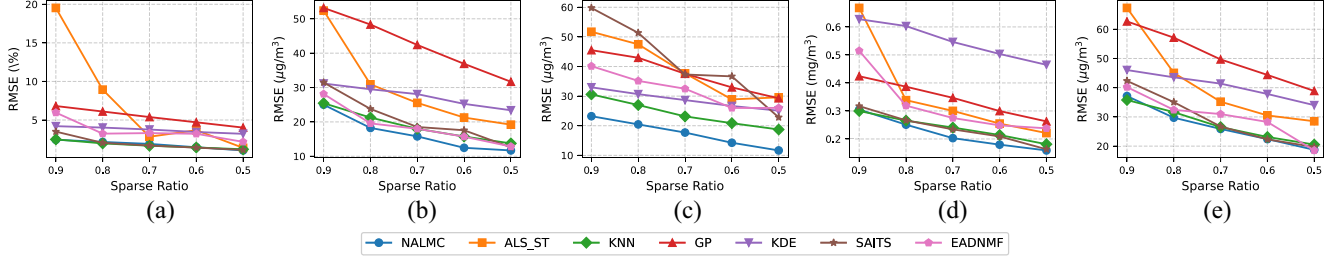


Fig. 6. RMSE of completion results with different sparse ratios (online). (a) Humidity. (b) PM2.5. (c) PM10. (d) CO. (e) O₃.

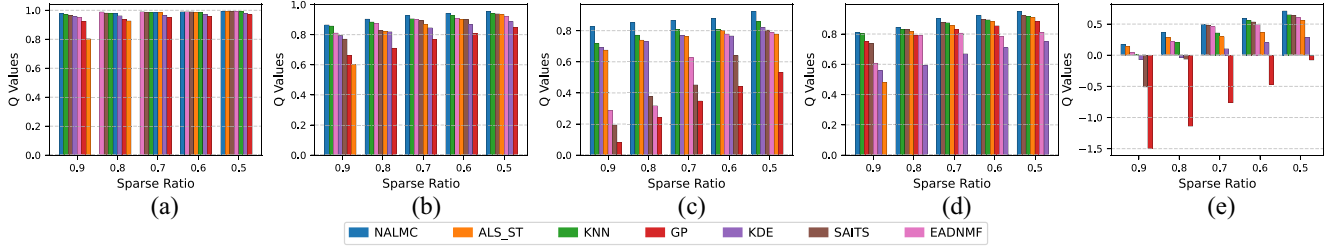


Fig. 7. Q of completion results with different sparse ratios (online). (a) Humidity. (b) PM2.5. (c) PM10. (d) CO. (e) O₃.

a highly variable segment from the PM2.5 dataset at a sparse ratio of 0.2, performing completion and comparing it against the ground truth in Fig. 4. NALMC not only captures short-term fluctuations better than ALS-ST but also improves accuracy over KNN, producing predictions closer to actual values. Other methods exhibit unique advantages and limitations. Notably, SAITS and EADNMF outperform traditional methods like GP and KDE, with slower error growth as sparsity increases.

When assessing QoS, we observe negative Q values in the PM2.5, PM10, and O₃ datasets, indicating that the average completion error exceeds the actual data values, rendering the service unusable. NALMC achieves the highest Q values across all datasets in Fig. 5, due to both its enhanced data completion accuracy and the dynamic clustering module, which favors higher density regions and further improves service quality.

- 2) *Online Data Completion:* For online data completion, only data from the current and preceding timeslots can be used, leading to lower completion accuracy than offline approaches. The detailed results are shown in Figs. 6 and 7. As can be seen, in the online case of PM2.5, RMSE increases by an average of 148% across all sparse ratios, which poses greater demands

on completion methods to capture information effectively. As in offline completion, accuracy remains highly dependent on dataset characteristics, but KNN performs particularly well, showing consistent success across datasets. This is due to KNN's local information capture ability, which allows it to adapt closely to previously observed data as new data arrives, while the global method provides limited assistance. Consequently, ALS-ST exhibits poor performance. However, NALMC's integration of neighborhood and latent matrix features enables it to leverage KNN's strengths while avoiding ALS-ST's drawbacks, achieving optimal results in four of the five tasks.

- 3) *Dynamic Clustering:* In our framework, we employed a dynamic clustering module to identify regions requiring sensing. To evaluate the effectiveness of this module in improving service quality, we compared the dynamic clustering strategy with a random selection approach. For clearer presentation, the results at different sparsity levels for each dataset were averaged, as shown in the Table III. The dynamic clustering module demonstrated positive improvements in four of the five tasks. Next, we delve into the experimental results presented in Table III to investigate the factors influencing the effectiveness of the clustering module. As observed from the

TABLE III
COMPARISON OF MEAN RMSE AND Q VALUES BETWEEN DYNAMIC AND RANDOM MODULES WITH IMPROVEMENT PERCENTAGE

Dataset	RMSE			Q		
	Dynamic	Random	Improvement (%)	Dynamic	Random	Improvement (%)
PM2.5	16.6435	17.8401	6.71	0.9178	0.9017	1.78
O3	26.7774	28.4853	6.00	0.4688	0.3791	23.65
Humidity	1.8030	1.7834	-1.09	0.9845	0.9847	-0.02
PM10	17.4548	20.3687	14.31	0.8698	0.8270	5.17
CO	0.2835	0.3482	18.57	0.8739	0.8206	6.48

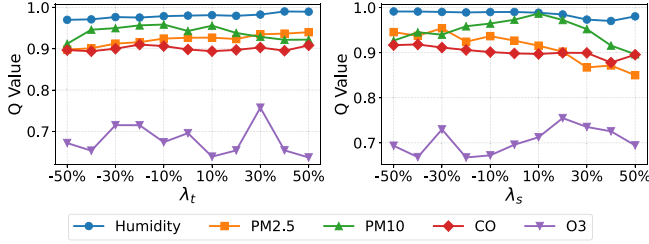


Fig. 8. Adjusting λ_t and λ_s by percentage.

table, the performance improvements of the clustering module vary across different datasets. Taking the Q value as an example, the highest improvement is seen with O_3 , which achieves a 23.65% enhancement compared to the random method, while the least effective case is Humidity, where performance decreases by 0.02%. The reason for this disparity lies in the design of the dynamic clustering module, which, to enhance service quality, tends to prioritize high population demand areas for sensing. Consequently, the selected sensing regions are somewhat more concentrated than those chosen by the random method. In cases where the dataset pertains to large-scale urban sensing tasks (e.g., the U -air dataset used in this article), the sensing range is broader and more aligned with real-world application scenarios, with data correlations between different sensing regions at a moderate level. Under such conditions, the dynamic clustering module significantly improves sensing service quality. In contrast, the Humidity data corresponds to the Sensor Scope dataset, sampled from the EPFL campus, where the sensing range is small, and the correlations between different sensing regions are extremely high. In this scenario, the dynamic clustering module has limited impact, while the random method, due to its broader coverage, enables NALMC to reconstruct the sensing data with a slight advantage.

The effectiveness of the dynamic clustering module is indeed influenced by the inherent characteristics of the dataset. However, we believe that in the vast majority of sensing scenarios—where tasks require large-scale monitoring of meteorological data, air quality, traffic conditions, and similar applications—the dynamic clustering module can effectively enhance service quality. This is evident in tasks, such as PM2.5, PM10, CO, and O_3 sensing. Moreover, for the Humidity dataset,

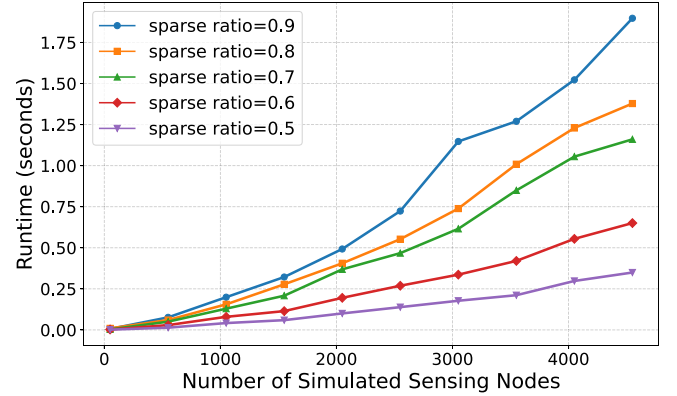


Fig. 9. Running time of dynamic cluster module.

the dynamic clustering module only marginally underperforms the random method, with negligible impact on overall sensing quality. Therefore, we conclude that dynamic clustering is a more suitable approach for real-world sensing scenarios.

- 4) *Spatial-Temporal Regularization Weights*: In the process of sensing data completion, we leverage the temporal and spatial structures inherent in the data to reconstruct the sensing data. Therefore, we adjusted the regularization weights for temporal and spatial smoothness, λ_t and λ_s , to evaluate their impact on the data completion results. Note that the baseline parameters used in this study are $\lambda_s = 0.1$ and $\lambda_t = 50$, a combination that generally yields favorable results across most scenarios. Improper values of λ_t and λ_s can prevent NALMC from converging during the update process. Here, we adjusted these two parameters proportionally, and the experimental results are presented in Fig. 8. We found that, although λ_t and λ_s exhibit a certain degree of robustness to adjustments, appropriate values of λ_t and λ_s can still significantly enhance the quality of data completion.
- 5) *Scalability Analysis*: As a framework designed for large-scale SMCS scenarios, this article analyzes the scalability of D2-SMCS from two perspectives to validate its usability on large-scale crowdsensing platforms. All experiments were conducted on a PC equipped with an AMD Ryzen 7 6800H CPU @ 3.20 GHz, 16 GB RAM, and Windows 11, using Python 3.11. First, the Dynamic Clustering module, which involves searching and adjusting clustering regions, is expected to incur increased computational overhead as the number of perception regions to be processed grows. To ensure

TABLE IV
RUNNING TIME OF DIFFERENT METHODS

	NALMC	KNN	GP	KDE	ALS-ST	SAITS	EADNMF
Running Time (s)	0.13	0.85	7.26	0.97	0.11	15.83	32.11

that the clustering module of D2-SMCS can still complete processing within a short time even when the number of perception nodes increases significantly, we constructed an artificial dataset with a larger number of perception nodes and applied the Dynamic Clustering module to it, measuring the runtime required under different sparsity levels. The dataset included perception nodes ranging from 50 to 5000, with increments of 500. The experimental results are presented in Fig. 9. As shown in Fig. 9, the runtime exhibits a nonlinear growth trend as the number of nodes increases, with the increase in computation time primarily attributed to the need to search for neighbors during clustering. Regarding the impact of sparsity, runtime increases more quickly at a lower sparsity level than at a higher one. This shows that sparse ratio setting significantly affects computational cost: lower sparse ratio requires merging more nodes, which increases the number of searches, while higher sparse ratio reduces this need. To ensure stability, we ran each configuration ten times and calculated the average, with results showing minimal performance fluctuations, indicating the algorithm's stability. We believe that the number of perception nodes in most sensing scenarios will not exceed the scale of this experimental dataset. Under such conditions, the runtime of the Dynamic Clustering module remains relatively short. Furthermore, since D2-SMCS operates in real-world scenarios on sensing platforms with abundant computational resources, the computation time would be further reduced, confirming the usability of the Dynamic Clustering module in large-scale sensing contexts.

On the other hand, we compared the runtime of NALMC with other baseline methods. It should be noted that the completion time in the offline phase is strongly positively correlated with the length of the offline time period, making it less meaningful for comparison. Therefore, we compared all methods in the online scenario. The average runtime for the online phase across five datasets is presented in Table IV. Considering that online completion relies on historical information, we provided 2000 timeslots of historical data for each method. For the NALMC and ALS-ST methods, updates involve only a small portion of the entire matrix, giving them an advantage in runtime efficiency. However, NALMC, which builds on ALS-ST with additional computations, requires slightly longer runtime. Traditional methods like KNN, which typically have shorter runtimes, face increased search and computation demands as historical information grows, resulting in longer runtimes. Meanwhile, methods involving deep learning, such as SAITS and EADNMF, require training on the dataset prior to completion, making their runtimes the longest.

VI. CONCLUSION

In this article, we consider the impact of selecting sensing regions on service quality under uneven population demand distribution, and seek to maintain high service quality while reducing costs by prioritizing more representative regions for sensing. To this end, we propose D2-SMCS, a novel SMCS framework focused on improving the quality of crowd services. D2-SMCS integrates three main modules: regional population demand calculation, dynamic clustering, and data reconstruction. It effectively balances sensing cost and reconstruction accuracy, while maintaining strong stability in reconstruction performance. The effectiveness and capability of the D2-SMCS framework in enhancing crowd service quality scores are demonstrated across two datasets containing five kinds of classic environmental data.

In the future, we plan to optimize this work in the following aspects. While this article focuses on the selection of sensing regions, we have simplified the problem by neglecting worker scheduling. To more closely align with real-world scenarios, we will consider how to more reasonably allocate tasks to workers. Additionally, we will address data reliability concerns. In practical applications, sensor errors and malicious data submissions from users can degrade completion performance. Handling unreliable data and evaluating worker trustworthiness are critical directions for our future efforts.

REFERENCES

- [1] W. Z. Khan, Y. Xiang, M. Y. Aalsalem, and Q. Arshad, "Mobile phone sensing systems: A survey," *IEEE Commun. Surveys Tuts.*, vol. 15, no. 1, pp. 402–427, 1st Quart., 2013.
- [2] A. Capponi, C. Fiandrino, B. Kantarci, L. Foschini, D. Kliazovich, and P. Bouvry, "A survey on mobile crowdsensing systems: Challenges, solutions, and opportunities," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 3, pp. 2419–2465, 3rd Quart., 2019.
- [3] D. E. Boubiche, M. Imran, A. Maqsood, and M. Shoaib, "Mobile crowd sensing—Taxonomy, applications, challenges, and solutions," *Comput. Human Behav.*, vol. 101, pp. 352–370, Dec. 2019.
- [4] S. Devarakonda, P. Sevusu, H. Liu, R. Liu, L. Iftode, and B. Nath, "Real-time air quality monitoring through mobile sensing in metropolitan areas," in *Proc. 2nd ACM SIGKDD Int. Workshop Urban Comput.*, 2013, pp. 1–8.
- [5] 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)*, 2016, pp. 1–6.
- [6] Y. Xu, Y. Zhu, and Z. Qin, "Urban noise mapping with a crowd sensing system," *Wireless Netw.*, vol. 25, pp. 2351–2364, Jul. 2019.
- [7] J. Wan, J. Liu, Z. Shao, A. V. Vasilakos, M. Imran, and K. Zhou, "Mobile crowd sensing for traffic prediction in Internet of Vehicles," *Sensors*, vol. 16, no. 1, p. 88, 2016. [Online]. Available: <https://www.mdpi.com/1424-8220/16/1/88>
- [8] X. Sheng, J. Tang, X. Xiao, and G. Xue, "Sensing as a service: Challenges, solutions and future directions," *IEEE Sensors J.*, vol. 13, no. 10, pp. 3733–3741, Oct. 2013.
- [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] S. Zhao, G. Qi, T. He, J. Chen, Z. Liu, and K. Wei, "A survey of sparse mobile crowdsensing: Developments and opportunities," *IEEE Open J. Comput. Soc.*, vol. 3, pp. 73–85, 2022.
- [11] F. Montori, P. P. Jayaraman, A. Yavari, A. Hassani, and D. Georgakopoulos, "The curse of sensing: Survey of techniques and challenges to cope with sparse and dense data in mobile crowd sensing for Internet of Things," *Pervasive Mobile Comput.*, vol. 49, pp. 111–125, Sep. 2018.
- [12] D. Zhao, H. Ma, and L. Liu, "Energy-efficient opportunistic coverage for people-centric urban sensing," *Wireless Netw.*, vol. 20, pp. 1461–1476, Aug. 2014.
- [13] L. Wang et al., "CCS-TA: Quality-guaranteed online task allocation in compressive crowdsensing," in *Proc. ACM Int. Joint Conf. Pervasive Ubiquitous Comput.*, Sep. 2015, pp. 683–694.
- [14] X. Wei, Z. Li, Y. Liu, S. Gao, and H. Yue, "SDLSC-TA: Subarea division learning based task allocation in sparse mobile crowdsensing," *IEEE Trans. Emerg. Topics Comput.*, vol. 9, no. 3, pp. 1344–1358, Jul.–Sep. 2021.
- [15] J. Wang, L. Wang, Y. Wang, D. Zhang, and L. Kong, "Task allocation in mobile crowd sensing: State-of-the-art and future opportunities," *IEEE Internet Things J.*, vol. 5, no. 5, pp. 3747–3757, Oct. 2018.
- [16] S. Song, Z. Liu, Z. Li, T. Xing, and D. Fang, "Coverage-oriented task assignment for mobile crowdsensing," *IEEE Internet Things J.*, vol. 7, no. 8, pp. 7407–7418, Aug. 2020.
- [17] Z. Liu, Z. Li, and K. Wu, "UniTask: A unified task assignment design for mobile crowdsourcing-based urban sensing," *IEEE Internet Things J.*, vol. 6, no. 4, pp. 6629–6641, Aug. 2019.
- [18] W. Liu, E. Wang, Y. Yang, and J. Wu, "Worker selection towards data completion for online sparse crowdsensing," in *Proc. IEEE Conf. Comput. Commun.*, May 2022, pp. 1509–1518.
- [19] E. Wang, M. Zhang, Y. Xu, H. Xiong, and Y. Yang, "Spatiotemporal fracture data inference in sparse urban crowdsensing," in *Proc. IEEE Conf. Comput. Commun.*, 2022, pp. 1499–1508.
- [20] P. Mohan, V. N. Padmanabhan, and R. Ramjee, "Nericell: Rich monitoring of road and traffic conditions using mobile smartphones," in *Proc. 6th ACM Conf. Embed. Netw. Sens. Syst.*, 2008, pp. 323–336. [Online]. Available: <https://doi.org/10.1145/1460412.1460444>
- [21] D. Hasenfratz, O. Saukh, S. Sturzenegger, and L. Thiele, "Participatory air pollution monitoring using smartphones," *Mobile Sens.*, vol. 1, pp. 1–5, Jan. 2012.
- [22] S. Kim, C. Robson, T. Zimmerman, J. Pierce, and E. M. Haber, "Creek watch: Pairing usefulness and usability for successful citizen science," in *Proc. SIGCHI Conf. Human Factors Comput. Syst.*, 2011, pp. 2125–2134. [Online]. Available: <https://doi.org/10.1145/1978942.1979251>
- [23] W. Liu, L. Wang, E. Wang, Y. Yang, D. Zeghlache, and D. Zhang, "Reinforcement learning-based cell selection in sparse mobile crowdsensing," *Comput. Netw.*, vol. 161, pp. 102–114, Oct. 2019. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S1389128619303706>
- [24] Z. Zhu, B. Chen, W. Liu, Y. Zhao, Z. Liu, and Z. Zhao, "A cost-quality beneficial cell selection approach for sparse mobile crowdsensing with diverse sensing costs," *IEEE Internet Things J.*, vol. 8, no. 5, pp. 3831–3850, Mar. 2021.
- [25] E. J. Candes and Y. Plan, "Matrix completion with noise," *Proc. IEEE*, vol. 98, no. 6, pp. 925–936, Jun. 2010.
- [26] K. Xie, X. Li, X. Wang, G. Xie, J. Wen, and D. Zhang, "Active sparse mobile crowd sensing based on matrix completion," in *Proc. Int. Conf. Manage. Data*, 2019, pp. 195–210. [Online]. Available: <https://doi.org/10.1145/3299869.3319856>
- [27] J. Fan and J. Cheng, "Matrix completion by deep matrix factorization," *Neural Netw.*, vol. 98, pp. 34–41, Feb. 2018. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0893608017302502>
- [28] J. L. Herlocker, J. A. Konstan, A. Borchers, and J. Riedl, "An algorithmic framework for performing collaborative filtering," in *Proc. 22nd Annu. Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 1999, pp. 230–237. [Online]. Available: <https://doi.org/10.1145/312624.312682>
- [29] Y. Koren, "Factorization meets the neighborhood: A multifaceted collaborative filtering model," in *Proc. 14th ACM SIGKDD Int. Conf. Knowl. Discov. Data Min.*, 2008, pp. 426–434.
- [30] J. Wang et al., "Learning-assisted optimization in mobile crowd sensing: A survey," *IEEE Trans. Ind. Informat.*, vol. 15, no. 1, pp. 15–22, Jan. 2019.
- [31] L. Wang et al., "SPACE-TA: Cost-effective task allocation exploiting intradata and interdata correlations in sparse crowdsensing," *ACM Trans. Intell. Syst. Technol.*, vol. 9, no. 2, pp. 1–28, Oct. 2017. [Online]. Available: <https://doi.org/10.1145/3131671>
- [32] E. Wang et al., "Outlier-concerned data completion exploiting intra- and inter-data correlations in sparse crowdsensing," *IEEE/ACM Trans. Netw.*, vol. 31, no. 2, pp. 648–663, Apr. 2023.
- [33] A. R. Ghotbabadi, S. Feiz, and R. Baharun, "Service quality measurements: A review," *Int. J. Acad. Res. Bus. Soc. Sci.*, vol. 5, no. 2, pp. 267–286, Feb. 2015.
- [34] 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. [Online]. Available: <https://doi.ieeecomputersociety.org/10.1109/TMC.2012.205>
- [35] K. L. Huang, S. S. Kanhere, and W. Hu, "On the need for a reputation system in mobile phone based sensing," *Ad Hoc Netw.*, vol. 12, pp. 130–149, Jan. 2014. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S1570870511002174>
- [36] X. Oscar Wang, W. Cheng, P. Mohapatra, and T. Abdelzaher, "ARTSense: Anonymous reputation and trust in participatory sensing," in *Proc. IEEE INFOCOM*, 2013, pp. 2517–2525.
- [37] J. Star and J. Estes, "Geographic information systems: An introduction," *Geocarto Int.*, vol. 6, no. 1, pp. 46–46, 1991.
- [38] E. J. Candes and T. Tao, "The power of convex relaxation: Near-optimal matrix completion," *IEEE Trans. Inf. Theory*, vol. 56, no. 5, pp. 2053–2080, May 2010.
- [39] F. Ingelrest, G. Barrenetxea, G. Schaefer, M. Vetterli, O. Couach, and M. Parlange, "SensorScope: Application-specific sensor network for environmental monitoring," *ACM Trans. Sens. Netw.*, vol. 6, no. 2, pp. 1–32, Feb. 2010.
- [40] 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. Discov. Data Min.*, Aug. 2013, pp. 1436–1444.
- [41] (DataFountain, Amsterdam, Noord-Holland). *Key Area Population Density Prediction*. 2020. [Online]. Available: <https://www.datafountain.cn/competitions/428/datasets>
- [42] T. Cover and P. Hart, "Nearest neighbor pattern classification," *IEEE Trans. Inf. Theory*, vol. 13, no. 1, pp. 21–27, Jan. 1967.
- [43] F. Yin and F. Gunnarsson, "Distributed recursive gaussian processes for RSS map applied to target tracking," *IEEE J. Sel. Topics Signal Process.*, vol. 11, no. 3, pp. 492–503, Apr. 2017.
- [44] E. Parzen, "On estimation of a probability density function and mode," *Ann. Math. Statist.*, vol. 33, no. 3, pp. 1065–1076, 1962. [Online]. Available: <https://doi.org/10.1214/aoms/1177704472>
- [45] W. Du, D. Côté, and Y. Liu, "SAITS: Self-attention-based imputation for time series," *Expert Syst. Appl.*, vol. 219, Jun. 2023, Art. no. 119619. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0957417423001203>
- [46] S. A. Seyed, F. Akhlaghian Tab, A. Lotfi, N. Salahian, and J. Chavoshinejad, "Elastic adversarial deep nonnegative matrix factorization for matrix completion," *Inf. Sci.*, vol. 621, pp. 562–579, Apr. 2023. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0020025522014256>