$\mathcal{AFCS}$: Aggregation-Free Spatial-Temporal Mobile Community Sensing

Jiang Bian, Haoyi Xiong, *Senior Member, IEEE*, Zhiyuan Wang, Jingbo Zhou, *Member, IEEE*, Shilei Ji, Hongyang Chen, *Senior Member, IEEE*, Daqing Zhang, *Fellow, IEEE*, Dejing Dou, *Senior Member, IEEE*

Abstract—While spatial-temporal environment monitoring has become an indispensable way to collect data for enabling smart cities and intelligent transportation applications, the cost to deploy, operate and maintain a sensor network with sensors and massive communication infrastructure is too high to bear. Compared to the infrastructure-based sensing approach, community sensing, or namely mobile crowdsensing, that leverage community members' mobile devices to collect data becomes a feasible way to scale up the spatial-temporal coverage of the sensing system. However, a community sensing system would need to aggregate sensors and location data from community members and thus would raise concerns on privacy and data security. In this paper, we present a novel community sensing paradigm $\mathcal{AFCS}$ –*Sensor and Location Data Aggregation-Free Community Sensing*, which is designed to obtain the environment information (e.g., spatial-temporal distributions of air pollution, temperature, and bike-shares) in each subarea of the target area, without aggregating sensor and location data collected by community members. $\mathcal{AFCS}$ proposes to orchestrate with the Trusted Execution Environments (TEEs) of every community member's mobile device to cover the communication, computation and storage with spatial-temporal data. Further, $\mathcal{AFCS}$ proposes a novel *Decentralized Spatial-Temporal Compressive Sensing* framework based on *Parallelized Stochastic Gradient Descent*. Through learning the *latent structure* of the spatial-temporal data via decentralized optimization, $\mathcal{AFCS}$ approximates the value of the sensor data in each subarea (both covered and uncovered) for each sensing cycle using the sensor data locally stored in every member's TEE instance. Experiments based on real-world datasets and the Virtual Mobile Infrastructure (VMI) with TEE emulations demonstrate that $\mathcal{AFCS}$ exhibits low approximation error (i.e., less than 0.2°C in city-wide temperature sensing, 10 units of PM2.5 index in urban air pollution sensing, and 2 bikes in city-wide bike sharing prediction) and performs comparably to (sometimes better than) state-of-the-art algorithms based on the data aggregation and centralized computation.

1 Introduction

Spatial-temporal community sensing is well-known as an efficient paradigm to monitor the time and space phenomena in the environment, such as air pollution, temperature, or some traffic statuses, by incorporating the mobile sensors of community members. According to [1], there are two significant roles in community sensing, which is the *organizer*, and the *participants*. The former is the individual or organization that creates the sensing task, recruits participants and collects the sensor data. In contrast, the latter (i.e., participants) involve in the sensing task and provide the sensing data. Frequently, the organizer pursues a high (or even complete) spatial-temporal coverage of the collected sensor data. However, incentives (e.g., monetary rewards) and privacy threats (e.g., exposing real-time locations) are two major concerns that may affect the willingness of participants to join community sensing.

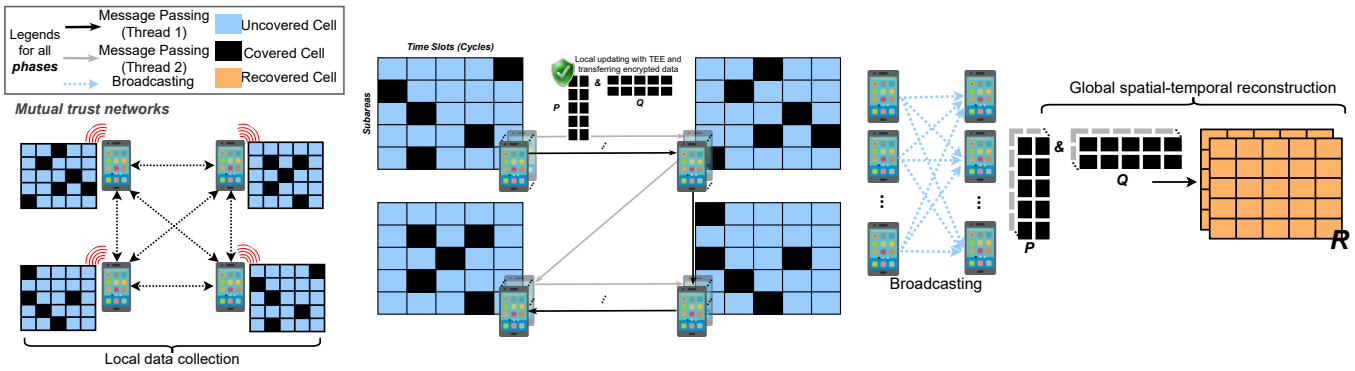
In addition to the community sensing paradigm, a wide spectrum of applications, ranging from vehicle traffic monitoring [2], [3], [4], [5], [6] to air quality sensing [7] and urban noise monitoring [8], have been proposed to efficiently monitor the environment of a large area through aggregating the real-time sensor and location data from the participants. Such applications use spatial-temporal coverage as the metric for overall task performance. Precisely, to characterize spatial-temporal coverage, the target area is split into subareas, and the sensing duration consists of a sequence of equal-length sensing cycles. In this way, the fraction of subareas covered by at least one sensor reading in each cycle offers sufficient help to measure the spatial-temporal coverage.

For example, [9], [10] proposed to use the *full spatial-temporal coverage* as the criterion of the participants selection for community sensing, while [11], [5] studied the *partial spatial-temporal coverage* as the objectives of the optimization for budget-constrained participant selection. With the sensor data that partially covers the target area, [12], [13], [14] proposed *compressive community sensing*, which is capable of recovering the missing sensor data of the uncovered subareas from the data collected. Compressive community sensing makes it possible to accurately monitor the target area with even lower spatial-temporal coverage, thus resulting in reduced incentive consumption and fewer participants involved.

Although compressive community sensing can effectively reduce the required incentives and participants, it still aggregates the real-time location and sensor data from each participant to identify the covered subareas, fill with collected data, and then recover the missing data for the rest. To protect the location privacy of participants, the same group of researchers [15], [16] proposed to leverage the *Differential Geo-Obfuscation* to replace each participants' real-time location with a "mock" location while ensuring the

The first two author contributed equally to this work.
Jiang Bian, Haoyi Xiong, Zhiyuan Wang, Jingbo Zhou, Shilei Ji, and Dejing Dou are with Baidu Inc., Beijing, China.
Zhiyuan Wang is also with Department of Systems and Information Engineering, University of Virginia, Charlottesville, VA, USA
Hongyang Chen is with the Research Center for Intelligent Network, Zhejiang Lab, Hangzhou, Zhejiang, China.
Daqing Zhang is with the Department of Computer Science, Peking University, Haidian, Beijing, China.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMC.2022.3178885, IEEE Transactions on Mobile Computing



(a) **Phase I:** Local Data Collection and Mutual Trust Network Establishment via Secure P2P Connections

(b) **Phase II:** Message Passing over Mutual Trust Networks for Decentralized Spatial-Temporal Compressive Sensing

(c) **Phase III:** Global Spatial-Temporal Data Reconstruction via Broadcasting

Fig. 1: Overall Design of $\mathcal{AFCS}$.

recovery accuracy. With the Differential Geo-Obfuscation, we can entirely obfuscate the participants' locations; however, it is still possible to attack the their location when leaking specific prior knowledge.

Thus, in our research, we propose $\mathcal{AFCS}$— a novel *Aggregation-Free Spatial-Temporal Mobile Community Sensing framework*, with the following principles on systems design:

- *Local-Only Storage and Computing* : *While each participant is supposed to cover one or multiple subareas in each sensing cycle with his/her mobility, all these locations and sensor data points are locally stored in the Trusted Execution Environment (TEE), e.g., ARM TrustZone [17], of his/her mobile device. Note that, as all local data are stored in the TEE in an encrypted form, the $\mathcal{AFCS}$ app should be certified as a Trusted Application (TA) [18] in the rich Operating System (RichOS) interfaced by the TEE instance, so as to access the local data for computation and communicate with other devices.*
- *Decentralized Spatial-Temporal Data Recovery*: *While spatial-temporal data collected by every participant is locally stored in the TEE instance of his/her mobile device and raw data sharing is prohibited, certain decentralized algorithms, such as Gossip [19], have been proved to achieve the global status using local information through message passing. Thus, it is reasonable to replace the spatial-temporal data aggregation procedure with decentralized algorithms.*
- *Low-Cost Missing Data Inference*: *While Gossip-based algorithms could approximate the global status of spatial-temporal information without aggregating the raw sensor/location data (when participants fully covered all subareas), it is still difficult to obtain the sensing data in the subareas that have not been covered by participants' mobility. Thus, there needs to infer the missing data for such uncovered subareas, in above mentioned decentralized algorithms, with low computation and communication costs that are tolerable to mobile devices.*
- *Mutual Trust Networks Connecting TEEs*: *While local data is assumed to be stored in TEE and computed by TAs (i.e., certified community sensing apps), we further assume that $\mathcal{AFCS}$ apps use inter-connections between TEE instances to communicate with each other. Note that in a sensing task, participants are community members, they thus could establish secure peer-to-peer (P2P) connections with encrypted messaging and public/private key exchanges to protect the message passing between TEEs/TAs.* efrv4

Thus, to achieve the above design goals, we design $\mathcal{AFCS}$ as Fig. 1. Specifically, $\mathcal{AFCS}$ first runs community sensing apps as Trusted Applications (TAs) in the Rich OS of the mobile devices for every participant, while the networking and storage operations of every TA are encapsulated by the TEE instance. Then, $\mathcal{AFCS}$ follows **three phases** to enable spatial-temporal community sensing sequentially. In Phase I, $\mathcal{AFCS}$ establishes a mutual trust network to connect TEEs of every two participants, using the secure P2P connections among the community members. Further in Phase II, $\mathcal{AFCS}$ proposes a decentralized message passing procedure based on *Parallelized Stochastic Gradient Descent* algorithms for spatial-temporal missing data inference. Through learning the latent structures of spatial-temporal data via decentralized algorithms, $\mathcal{AFCS}$ recovers the sensor data in each subarea (both covered and uncovered) for each sensing cycle using the sensor data that are locally stored in each participant's mobile device without data aggregation. Finally in Phase III, $\mathcal{AFCS}$ will broadcast the sensing results and finish the global spatial-temporal data reconstruction. The contributions of this paper are as follows:

- To the best of our knowledge, this paper is the first work that studies the problem of aggregation-free community sensing by addressing location privacy, decentralized message passing, Trusted Execution Environment (TEE), and Trusted Application (TA) issues. We propose a novel community sensing framework $\mathcal{AFCS}$, which aims at recovering the environmental information in subareas without aggregating sensor and location data from the community members who partially cover the target area. The basic idea here is to combine the trusty computation techniques, e.g., TEEs and TAs, with decentralized message-passing algorithms to pursue the global spatial-temporal information using the data locally stored without raw data exchange.
- To enable community sensing without location/sensor data aggregation, $\mathcal{AFCS}$ proposes a

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMC.2022.3178885, IEEE Transactions on Mobile Computing

novel spatial-temporal missing data inference (STMI) framework that recovers the spatial-temporal information through decentralized message passing. The proposed solution operates on top of the *parallelized stochastic gradient descent (SGD)*, which minimizes the reconstruction loss of STMI through passing intermediate results over community members. The algorithm analysis shows that the proposed solution efficiently approximates the centralized STMI with the tolerable worst-case communication complexity.
- We evaluate $\mathcal{AFCS}$ using three large real-world datasets (i.e., temperature, air pollution, and bike-sharing). The experimental results demonstrate that $\mathcal{AFCS}$ tightly approximates to the state-of-the-art algorithms based on the data aggregation with centralized computation, and it outperforms the rest baselines with a significant margin. Furthermore, $\mathcal{AFCS}$ also outperforms a wide range of algorithms including STCS [12], RPCA [20], TSVD [21], FISTA [22], DRS [23], and our previous work CSWA [24].

We organize the rest of the paper as follow. The Preliminaries and Problem formulation section reviews the compressive community sensing and the matrix factorization approach. Then we introduce the parallelized stochastic gradient descent and present the problem formulation. In the Frameworks and Algorithms section, we propose a framework of $\mathcal{AFCS}$ and present the algorithms in detail. In the Experiments section, we evaluate $\mathcal{AFCS}$ on real-world datasets and compare it with baseline algorithms. Finally, in the Conclusion section, we summarize the work.

2 Preliminaries, Definitions, and System Models

In this section, we first briefly introduce the previous work on compressive community sensing. Then, we formulate the problem of our research.

2.1 Compressive Community Sensing

To derive the target full sensing matrix from partially collected sensing readings, the compressive community sensing [12] commonly works as an effective approach, which consists of two parts:

2.1.1 Sensing Data Aggregation

Given the target region splitting into a set of subareas (denoted as S) and a set of m participants, to obtain the full picture of the target region for each sensing cycle (e.g., the t^{th} cycle), the Compressive Community Sensing system first collects the sensing data from all participants. Specifically, the subareas covered by the j^{th} participant in the t^{th} sensing cycle ($t \in T$) is denoted as $S_j^t \subset S$. Thus, the overall coverage in the sensing cycle t can be denoted as $S^t = S_1^t \cup S_2^t \cup ... \cup S_m^t$. Due to the limited mobility of each participant and a limited number of participants involved, the overall coverage can usually include a subset of subareas, i.e., $S^t \subseteq S$. Given the collected sensing data, the compressive community sensing system aggregates the data and assigns each covered subarea a unique sensor data value. For example, if multiple sensor data values collected (from multiple participants) cover the same subarea in a sensing cycle, the *averaged* value would represent as the value of such subarea in the sensing cycle. In this way, each subarea $s \in S^t$ has been covered with one sensor data value through data aggregation. The compressive community sensing system needs to infer missing sensor data of subareas in $S \backslash S^t$ to obtain sensor data of the whole target area.

2.1.2 Missing Data Inference

Given the aggregated sensor data of the covered subareas (S^t), there exists a wide range of inferring techniques to infer the missing data of the uncovered subareas, such as expectation maximization [25] and singular spectrum analysis [26]. One of the powerful approaches is spatial-temporal compressive sensing [27]. The essential idea of this approach derives from the spatial-temporal missing data inference (STMI) [28]. Given the aggregated sensor data of recent sensing cycles (the number of recent sensing cycles used for STMI is denoted as w), this approach first sorts the subareas using their indices from $1 \ldots$ to $|S|$, then maps the data into a $|S| \times w$ matrix denote as R, where the element $R_{a,t}$ ($1 \leq a \leq |S|$ and $1 \leq t \leq w$) refers to the aggregated sensing value of the a^{th} subarea and t^{th} sensing cycle (in the window). To recover the missing values in R, this approach obtains two non-negative ***matrix factors*** $P \in \mathbb{R}^{|S| \times l}$ and $Q \in \mathbb{R}^{l \times w}$ such that $R \approx PQ$, through STMI, where l stands for the *Size of Latent Space* of STMI.

Typically, there are four key factors affecting the performance of the compressive community sensing: (1) *The Number of Subareas* that each participant covers in each sensing cycle; (2) *The Number of Participants (m)* which, together with the number of subareas per participant, can determine the coverage of collected sensor data; (3) *The Size of Windows (w)* that refers to the number of past sensing cycles used for matrix recovery (i.e., the width of the matrix for matrix completion); (4) *The Size of Latent Space (l)* that determines the rank of matrices for low-rank matrix recovery/completion.

2.2 Definitions and System Models

Given a set of participants, where each participant's mobile device stores the raw sensor data locally (without raw data sharing), our proposed work intends to recover the sensing data of the target area while assuming that any form of aggregation is not allowed among participants. Specifically, we make the following assumptions:

- For all the sensing cycles in T and subareas in S, there exists an unknown spatial-temporal sensor data matrix R^* ($R^* \in \mathbb{R}^{|S| \times T}$), where each element $R^*_{a',t'}$ ($1 \leq a' \leq |S|$ and $1 \leq t' \leq |T|$) refers to the real value of sensor data in the corresponding subarea a' and sensing cycle t'.
- In each sensing cycle (e.g., the t^{th} cycle), each participant (e.g., the j^{th} participant) covers a subset of subareas (i.e., $S_j^t \subseteq S$) in the target area. Thus, all the collected sensor data from the 1^{st} to the t^{th} sensing cycle of the j^{th} participant can form as a matrix $R^j \in \mathbb{R}^{|S| \times t}$, where each element refers to the value of the sensor data collected in the corresponding subarea

and cycle. Note that, to protect the location privacy, R^j is not known by any third party.

- We denote the value of the sensor data collected by the j^{th} participant in sensing cycle t at subarea a as $R^j_{a,t}$. Each sensor datum obtained is assumed to be the true value with (unknown) random noise, i.e., $R^j_{a,t} = R^*_{a,t} + \varepsilon^j_{a,t}$. For any two participants (i.e., the j^{th} and k^{th} participants), they might cover the same subarea (say, $a^t_j \cap S^t_k \neq \emptyset$ is possible), but are with *different* sensor data value obtained, due to the noise.

Our problem is that, in each sensing cycle t, with R^j ($1 \leq j \leq N$) locally stored on each participant's device, there needs to estimate $\widehat{R}_{a,t}$ to

$$\textbf{minimize} \sum_{a=1}^{|S|} (\widehat{R}_{a,t} - R^*_{a,t})^2 \textbf{ for } 1 \leq t \leq T, \qquad (1)$$

while ensuring that the raw sensor/location data sharing is not allowed between the participants.

Note that the $P \in \mathbb{R}^{|S| \times l}$ and $Q \in \mathbb{R}^{l \times w}$ matrix stand for the (location $\times$ patterns) and (time $\times$ patterns) as the lower dimensional representations of the matrix R. For example, in Fig. 1 Phase II, the R matrix is the target matrix for recovering, in which each column of the matrix represents the subareas covered and each row represents the sensing cycles (time). Thus, the sensing value from each cycle forms the Q matrix and definitely is dependent on the neighboring cycles (values) in short term dependence or the overall distribution of cycles in long term dependence. While in the optimization of non-negative matrix factorization, the dependence (the pattern of matrix Q) finally affect the matrix recovering performances comparing to the ground-truth matrix $\widehat{R}$.

3 Aggregation-Free Community Sensing: Framework Design

In this section, we will first introduce the overall design of $\mathcal{AFCS}$. Then, we elaborate on three key phases of $\mathcal{AFCS}$ in detail.

3.1 Overall Design

The overall framework design of $\mathcal{AFCS}$ consists of three phases. In **Phase I**, the mobile devices participate in the sensing task by collecting the spatial-temporal data locally and connect each other (peer-to-peer) via establishing mutual trust networks. In such secured networks, the mobile devices can collaboratively sense and optimize the Eq. 1 by passing the updated massage. Then in **Phase II**, multiple threads of random massage passing begin to process on the top of the mutual trust networks, where each thread is independent and isolated from others. Finally, in **Phase III**, the converged target values will be broadcast to all participants in mutual trust networks. Once receiving target values from all threads, the mobile device can obtain the recovered global data matrix by the spatial-temporal reconstruction.

3.2 Local Data Collection and Mutual Trust Network Establishment with TEEs

As we illustrated, each participant collects its local data as a spatial-temporal matrix $R^j \in \mathbb{R}^{|S| \times t}$, where the matrix represents the data sensing from all cycle T and all covered subareas S. As shown in Fig. 1(1), the collected local data is represented as a matrix with blue uncovered area cells and black covered cells. Then, th e participants will establish mutual trust networks via peer-to-peer communications based on the Trusted Execution Environment (TEE) [18]. As known by its strong confidentiality, executing local updates inside TEEs can hide parameters or raw data from adversaries such as data reconstruction attack (DRA) [29], property inference attack (PIA) [30], and membership inference attack (MIA) [31]. The overhead of loading and running algorithms in TEE (i.e., ARM TrustZone on mobile devices) on each of the community members can be ignored compared to the cost of running the algorithm itself in REE (rich execution environment).

Fig. 2 illustrates the way to connect two mobile devices (participants). The green part stands for the ARM TrustZone (TEE), which allows user-level code to define private regions of memory that cannot be read or accessed by any process running outside, including the operating system and hypervisor. Specifically, in each mobile device, three functional parts involve local updating and the message passing. The collected local data are stored in TEE-secured storage (the first part). Only the second part – ARM TrustZone – can access the local data, while it is also capable of loading Trusted Applications (TAs) (the third part) to process the local updating. To transfer the target applications to the TAs, we provide three optional plans: (1) modifying the application based on the protocol of ARM TrustZone, which is the most secure method; (2) leveraging the Lib OS (e.g., Graphene [32]) to isolate the target application; (3) using software signature [33], [34] without modifying the original applications, which is fast but less secure. It would depend on the circumstance to use one of them to achieve the TAs initialization. After finishing the local updating, the Crypto Engine in TEE is in charge of applying additional encryption methods to the massage waiting for passing to the other trusted participants. Note that ARM TrustZone based TEE can provide all required functionality to achieve the cryptographic operations, including secure key generation, key derivation, symmetric and asymmetric cryptographic operations such as AES, RSA, ECC key data generation, sign, verify, import key data, get public key, etc..

Once two mobile devices sign the agreement (through the key exchange) to trust each other, the TEE parts will establish a channel to passing the encrypted messages. Note that two mobile devices can also establish a soft mutual trust network through an in-between mobile device with which both mobile devices has established the mutual trust networks (e.g.,if $A \leftrightarrow B$, $B \leftrightarrow C$, then $A \overset{\text{Soft}}{\longleftrightarrow} C$). As we defined, only two mobile devices in mutual networks can communicate with encrypted messages and leverage the local data to update the target values.

3.3 Message Passing over Mutual Trust Networks for Decentralized Spatial-Temporal Compressive Sensing

In this section, we will first introduce the mathematical foundation for message passing via parallelized stochastic gradient descent. Then, to further improve the overall communication efficiency, we design a steering pool strategy to

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMC.2022.3178885, IEEE Transactions on Mobile Computing

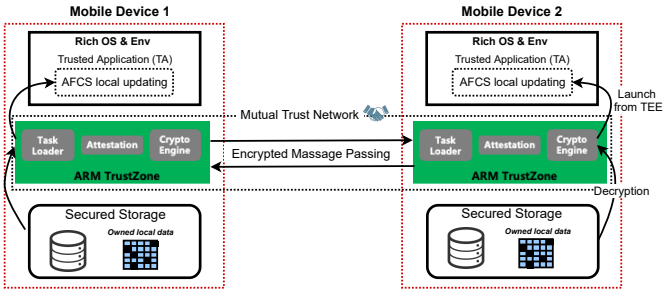


Fig. 2: Initialization the TEE based mutual trust network.

avoid the local deadlock through a completely asynchronous parallelism.

3.3.1 Single-thread Stochastic Gradient Descent

To solve the optimization problem in Eq. 1, we can adopt a group optimization algorithms. Among them, Gradient Descent (GD) is an iterative optimization algorithm, where, with an initial setting of ω, the algorithm updates ω by its gradient information. The SGD algorithm keeps updating ω iteratively until the total number of iterations exceeds the maximum allowed value or the updated error converges. Specifically, we denote the Eq. 1 as minimization problem which minimizes a loss function $\mathcal{L}(\omega|R)$. In each (the $e+1^{th}$) iteration, the GD algorithm updates ω_e and obtains ω_{e+1} using the following scheme:

$$\omega_{e+1} \leftarrow \omega_e - \eta \cdot \sum_{j=1}^{m} \bigtriangledown \mathcal{L}(\omega_t|R^j)/m, \tag{2}$$

where η refers to the step size and $\sum_{j=1}^{m} \bigtriangledown \mathcal{L}(\omega|R^j)$ is the sum of gradients.

However, sometimes, the sum of gradient functions are not available. For example, in distributed computing environments (e.g., spatial-temporal community sensing), R^j's are distributed in multiple mobile devices and are not sharable. In this case, Stochastic Gradient Descent (SGD) algorithm has been proposed to solve the optimization problem in Eq. 1 in an ad-hoc manner. In each iteration, compared to GD, the SGD randomly picks up one R^j from $R^1 \dots R^m$, and obtains ω_{e+1} using the gradient of a single loss function $\mathcal{L}(R^e|R^j)$. Specifically, in the iteration, SGD randomly selects an integer $j \in [1, m]$, then it updates ω using

$$\omega_{e+1} \leftarrow \omega_e - \eta \cdot \bigtriangledown \mathcal{L}(\omega_e|R^j). \tag{3}$$

Note that, in distributed optimization problems, where R^j's are distributed on different participants, the algorithm mentioned above can be implemented as a gossip-based stochastic gradient descent by exchanging the (updated) ω between participants to approximate the optimal solution.

3.3.2 Multi-thread Stochastic Gradient Descent

To further accelerating the optimization process, we leverage the Parallelized SGD framework to solve the optimization problem in Eq. 1. Suppose the SGD algorithm can be regarded as a single thread with the index k, we reclaim the Eq. 3 as

$$\omega_{e+1}^k \leftarrow \omega_e^k - \eta \cdot \bigtriangledown \mathcal{L}(\omega_e^k|R^j), \tag{4}$$

where $k \in \{1 \dots S\}$, S is the size of multiple threads. Note that each k^{th} thread runs an independent SGD algorithm and the k^{th} optimal result $\widehat{\omega}^k$ can be obtained when the SGD algorithm converged. Once we have all the converged $\widehat{\omega}^k$ from S threads, the overall optimal result can be averaged by

$$\bar{\omega} \leftarrow \frac{1}{S} \sum_{k=1}^{S} \omega_k. \tag{5}$$

The multi-thread process runs the SGD algorithm in parallel and does not affect other threads when passing the message among the selected machines.

3.3.3 Parallelized Random Message Passing

Motivated by the above-mentioned characteristics of parallelized SGD, we design a parallelized random message passing mechanism over mutual trust networks to handle the problem of missing spatial-temporal data inference. The raw data is not necessary to share since we only need to transfer ω among the participants in mutual networks and update ω using local data. In this work, the Parallelized Random Message Passing Mechanism is proposed and adopted in the second phase of $\mathcal{AFCS}$. Specifically, the randomly selected mobile devices in its thread update the target value based on the receiving message and passes to the next participant for another round until converged. In Fig. 1, the solid black lines represent one time of message passing from one participant to another one and the one received (marked with received on top on mobile devices) the message will update the target value, while the participant not received message will stay idle for this round of message passing. Note that the black dash lines differentiate from the solid one because they will run more than one round of massage passing until converged. One more setting is designed to secure the message passing (illustrated in the later algorithm design section). When passing the message between two participants, the *receiver* can not send the updated matrix factors back to the *sender*. In contrast, the *sender* can possibly recover the *receiver's* local sensing data by recalculating the return messages.

3.3.4 Lock-free Design

However, multiple tasks (threads) may request the same participant at the same time (in periods have overlaps), which may lead to a deadlock that this participant has to wait with the coming tasks until the current updating process is finished due to the limited computing resource. As we can imagine, multiple threads are possible to be stuck in a single participant while the final broadcasting is affected drastically. To address this issue, we propose a so-called steering pool strategy to mitigate the possible local deadlock. To be specific, each participant creates a status variable when establishing the mutual trust networks. This status variable can indicate if the carrier participant (i.e., own this status variable) is vacant or not (in local updating). We form a steering pool in the mutual trust network, where each participant in the steering pool is busy with a specific thread's updating tasks. Each time when a participant is ready for sending the message, it will randomly select a receiver (next participant) which is excluded from the steering pool. In this case, the aforementioned conflict would be avoided since the participants always send messages to the next participants who are vacant. Note that we assume that each participant can only process a single local updating due to the limited

computing resource and occupied local data. The proposed steering pool strategy can be extended to the settings which allow parallel updating on a single participant.

3.4 Global Spatial-Temporal Data Reconstruction via Broadcasting

Following Phase II, when one of the threads converged on a specific participant (finished the last round of updating), this participant will broadcast the converged target value (P, Q Matrix) to other participants in a mutual trust network. The *Spatial-Temporal Data Recovery* includes two steps, which are *Broadcasting* and *Averaging*. In *Broadcasting*, the mobile devices marked by the checked superscript represent the target value passing through that participant has been converged and will be broadcasted to all the participants (solid black line). Finally, in *Averaging*, the participant will average all the received target values to reconstruct the global data matrix. In the following sections, we will introduce the detailed key algorithms in three phases, respectively.

4 Aggregation-Free Community Sensing: Algorithms Design

In this section, we introduce the underlying algorithms in three phases of $\mathcal{AFCS}$.

4.1 Initialization

Before initializing the *thread* on each participant, we first establish a mutual trust network among m participants. In contrast, all the collected sensor data on the j^{th} participant map to a local data matrix R^j. Then, as shown in Algorithm. 1, $\mathcal{AFCS}$ randomly picks a set of participants which is the *thread* (denoted as the set L with size N) from the secure network of m participants. Next, given the target data matrix $R \in \mathbb{R}^{|S| \times w}$, $\mathcal{AFCS}$ extracts the row and column number of R to construct the initial matrix factors $\hat{P}$ and $\hat{Q}$ on each participant. Specifically, $\hat{P}_j$ is generated by a $|S| \times l$ Gaussian Random Matrix on the j^{th} participant. Similarly, a $l \times w$ Gaussian Random Matrix on the same j^{th} participant can generate $\hat{Q}_j$. To avoid the message as mentioned earlier transferring back between two participants, we initialize a counter i to record passing times (iterations) among participants and set j_p to mark the last participant's index, where the (i, j_p) will be transferred along with the updated matrix factors so that the participant who receives the message can randomly select the next one excluding participant j_p. When the initialization ends, each participant (I_j) in the predefined set L (*thread*) will be assigned a pair of starting matrix factors $\hat{P}_j$ and $\hat{Q}_j$.

4.2 Message Passing and Broadcasting

Given the mapped local data matrix R^j on j^{th} participant, $\mathcal{AFCS}$ intends to approximate the optimal estimation of matrix factors $\hat{P}_j$ and $\hat{Q}_j$ via parallelized stochastic gradient descent on top of non-negative matrix factorization algorithm. Specifically, the initialized $(\hat{P}_j, \hat{Q}_j, \mathbf{0}, \mathbf{null})$ has been allocated on the j^{th} participant, where $\mathbf{0}$ refers to the fact

Algorithm 1: Initializing Thread and Matrix Factors $(\hat{P}, \hat{Q})$ on each Participant in Mutual Trust Networks

Data:
$R_{|S| \times w}$ — the target data matrix
Parameter:
/* Subareas covered by per participant */
$|S|$ — the maximum numbers of subareas
w — the size of windows
l — the size of latent space
begin
  /* Predefine a set of participants */
  Randomly Draw N *Participants* into **Set** L
  /* $L = \{I_1, I_2, ..., I_N\}$ */
  for ***each*** $I_j \in L$ **do**
    /* Initialize matrix factors P, Q on I_j */
    $\hat{P}_j \leftarrow |S| \times l$ Gaussian Random Matrix
    $\hat{Q}_j \leftarrow l \times w$ Gaussian Random Matrix
    /* Initialize the counter and the previous participant index */
    SAVE $(\hat{P}_j, \hat{Q}_j, \mathbf{0}, \mathbf{null})$ as Passing Massage;
  end
end

Algorithm 2: Parallelized Optimization on the j^{th} Participant

Data:
R^j — the local data matrix on the j^{th} participant
F_j — the filter matrix on the j^{th} participant
Parameter:
i — the number iterations
j_p, j — the index of previous and current participant
η — step size
Δ_{min} — the minimum allowed perturbation
t_{max} — the maximum number of allowed updates
λ_P, λ_Q — regularization parameter on P and Q matrices
begin
  /* On receiving the message from the previous participant */
  RECEIVE $(\hat{P}_j, \hat{Q}_j, t, j_p)$
  /* Join the steering pool since it starts updating */
  Pool.append(j)
  /* Noting that "$A \circ B$" means element-wise matrix multiplication */
  $g_p \leftarrow (F_j \circ (R^j - \hat{P}_j \hat{Q}_j))\hat{Q}_j^T - \lambda_P \cdot \hat{P}_j$
  $g_q \leftarrow \hat{P}_j^T (F_j \circ (R^j - \hat{P}_j \hat{Q}_j)) - \lambda_Q \cdot \hat{Q}_j$
  $\hat{P}_j \leftarrow \hat{P}_j - \eta \cdot g_p$
  $\hat{Q}_j \leftarrow \hat{Q}_j - \eta \cdot g_q$
  /* Set the negative elements to zero */
  $\hat{P}_j, \hat{Q}_j \leftarrow Truncate(\hat{P}_j, \hat{Q}_j)$
  $i \leftarrow i + 1$
  /* Checking convergence conditions */
  $\Delta = max\left\{|g_p|_\infty, |g_q|_\infty\right\}$
  if $\Delta \geq \Delta_{min}$ ***AND*** $i \leq t_{max}$ **then**
    /* Not converged, continuing the algorithm */
    $j_{next} \leftarrow$ **Draw a random number from** 1 **to** m **except** j_p **and** j_{st} **if** $st \in Pool$;
    SEND $(\hat{P}_j, \hat{Q}_j, i, j)$ **to the** j_{next}^{th} **Participant**;
  else
    /* Converged, sharing the estimates to all participants */
    BROADCAST $(\hat{P}_j, \hat{Q}_j)$ to **All Participants**;
  end
  /* Excluded from the steering pool after updating */
  Pool.erase(j)
end

that no update has been executed and "**null**" refers to there is no previous participant (coming from the organizer) which has updated the matrix factors (the index of the previous participant is empty). Then the algorithm processes the updating task on each participant from the predefined *thread* (L) in parallel.

Suppose two dense matrix factors are $P \in \mathbb{R}^{|S|\times l}$ and $Q \in \mathbb{R}^{l\times w}$, the target minimization loss function over m participants through parallelized stochastic gradient descent is as follow:

$$\hat{P},\hat{Q} \leftarrow \underset{P\in\mathbb{R}^{|S|\times l},Q\in\mathbb{R}^{l\times w}}{argmin} \left\{ \frac{1}{m}\sum_{j=1}^{m} \left\| F_j \circ (R^j - PQ) \right\|_F^2 + \lambda_P \left\|P\right\|_F^2 + \lambda_Q \left\|Q\right\|_F^2 \right\}, \tag{6}$$

where l is the size of latent space, "$\circ$" means element-wise matrix multiplication, $\|\cdot\|_F$ is the Frobenius norm, λ_P and λ_Q are regularization parameters. Particularly, parallelly starting on each participant I_j, Algorithm. 2 first receives the input $(\hat{P}_j, \hat{Q}_j)$ from the last involved participant in the secure network (or initialized from the organizer in the first run). Next it updates the $(\hat{P}_j, \hat{Q}_j)$ using the mapped local data matrix R^j with the missing-value filter matrix F_j, and randomly picks up the next participant except the previous one (j_p) from the secure participants network and sends the updated $(\hat{P}_j, \hat{Q}_j)$ to this chosen participant. The matrix F_j is a matrix filling with 0 (missing) and 1 (collected) which can set the missing elements in matrix R^j to zero by the element-wise multiplication. We mainly use it to prevent the missing value in the local data matrix R^j from affecting the gradient updating in $(\hat{P}_j, \hat{Q}_j)$. In addition, we leverage the *Truncate()* function, where the negative values in matrix factors $(\hat{P}_j, \hat{Q}_j)$ will be set to zero, then ensuring the non-negativeness of $(\hat{P}_j, \hat{Q}_j)$ when finishing each update.

Algorithm. 2 keeps picking up the next participant for updating, until the times of updates i exceeds the maximal number of updates, or the updating process converges (i.e., $max\left\{|g_p|_\infty, |g_q|_\infty\right\} \leq \Delta_{max}$). Similar procedures are starting on each participant I_j, and the related matrix factors keep updating independently. However, as we are concerned in Section 3.3, the deadlock may occur when multiple threads request the same participant. We designed a steering pool strategy to filter out the occupied participants and ensure no conflict in each participant to mitigate the issue. The specific procedure is illustrated in the first if clause, where we randomly draw the index of the next involved participant except the one from the steering pool. Correspondingly, after the chosen next participant receives the passing information, it will join the pool as a steering member isolated from being selected for any thread. Then, when the local updating finishes, this participant will be excluded from the steering pool and return to the normal status, which is visible and available for the sensing task again.

Once the updating process completes on each participant, Algorithm 2 broadcasts all m participants with the final global estimation of $(\hat{P}, \hat{Q})$ pairs. To this end, each participant receives the same group of $(\hat{P}, \hat{Q})$ for recovery of the target data matrix.

Algorithm 3: Mobile Sensing Recovery on the j^{th} Participant

Data:
$\hat{P}_j, \hat{Q}_j$ — the received matrix factors from all the *threads*
begin
  /* Average all $\hat{P}_j, \hat{Q}_j$ */
  $\bar{P} \leftarrow \frac{1}{N}\sum_{j=1}^{N} \hat{P}_j$
  $\bar{Q} \leftarrow \frac{1}{N}\sum_{j=1}^{N} \hat{Q}_j$
  /* Recover the target overall data matrix */
  $\hat{R} \leftarrow \bar{P}\bar{Q}$
end

4.3 Reconstruction

As we have introduced in Section III-D, each participant can recover the target data matrix $\hat{R}$ based on the optimal estimated matrix factors $(\hat{P}, \hat{Q})$.

Given the received matrix factors $(\hat{P}_j, \hat{Q}_j)$ which are from the *threads*, Algorithm. 3 first separately average the $\hat{P}$ and $\hat{Q}$ from $j = 1$ to N. Then, to recover the target data matrix, the algorithm multiplies the averaged matrix factors $(\bar{P}, \bar{Q})$ and obtains the well-estimated target data matrix $\hat{R}$.

4.4 Algorithm Analysis

In this section, we brief the analytical results of the proposed algorithms from communication complexity and convergence rate aspects.

Given the overall set of subareas (S), the size of the latent space (l), the size of the windows (w), in each iteration, m participants in each *thread* would send out messages, while each participant sends a $|S| \times l$ matrix and a $l \times w$ matrix (i.e., P and Q matrices). In this way, the system-wide communication complexity in the worst-case (after the completion of t_{max} iterations of message-passing) should be $\mathcal{O}\left((|S| \cdot l + l \cdot w) \cdot t_{max} \cdot m\right)$. We could further improved it with the compressed row storage (CRS) representation, where the CRS format puts the subsequent non-zeros of the matrix rows in contiguous memory locations [35]. Specifically, let n_{pnz}, n_{qnz} denote the number of nonzero element in P, Q matrix. The CSR format saves on memory only when $n_{pnz} < (|S| \cdot (l-1) - 1)/2$ and $n_{qnz} < (l \cdot (w-1) - 1)/2$. Suppose we can eventually achieve the requirement, then the improved communication complexity is $\mathcal{O}\left((2n_{pnz} + 2n_{qnz} + |S| + l + 2) \cdot t_{max} \cdot m\right)$ for each thread. Thus, the memory cost of this redundant storage is likely insignificant for a sufficiently large matrix.

Suppose P^* and Q^* are the optimal solutions of the problem listed in Eq. 1, while $\bar{P}$ and $\bar{Q}$ (appeared in Algorithm 3) are two approximation results obtained by our algorithm. According to [36], the approximation error of $||P^* - \bar{P}||_F \to 0$ and $||Q^* - \bar{Q}||_F \to 0$, when $t_{max\to+\infty}$ and thread size N is sufficiently large. Note that with a larger N, the proposed algorithm can achieve a faster rate of convergence of the approximation error with increasing t_{max}. For more theoretical analysis, please refer to [36].

Discussion on the convergence rate – According to [37], we intend to obtain the number of iterations to reach the desired accuracy η. Based on the fast rate bound in [37], the iterations for stochastic gradient descent (single thread) to reach accuracy η is $\mathcal{O}\left(\frac{dvk^2}{\eta}\right)$, where v, d are the dimension factors (constant), k is the indicator of the difficulty of the

optimization [38]. Ignoring the effect of dimensions (i.e., v and d), setting them as 1, and assuming k as $1/\lambda$. In terms of the guarantee for the stationary distribution of parallel stochastic gradient decent [36] (Theorem 12), we assume $G = 1$ and $||\nabla_c||_L = 1$. Then according to [37]'s claim, we interpret the iteration bound as $\frac{1}{\eta\lambda^2}$ here. In other word, we have to set $\frac{1}{\lambda}$ machines to run $\frac{1}{\eta\lambda}$ time, which is the same order of computation, but a significant speedup of a factor of $\frac{1}{\lambda}$ in wall clock time. Another benefit we can obtain from the parallel setting is that the algorithm could be customized to arbitrary precision. Through halving η and doubling the $T = \frac{(\ln k - (\ln \eta + \ln \lambda))}{2\eta\lambda}$, we can halve the error accordingly. Furthermore, the bound [36] (Theorem 12) illustrates how much parallelization can help with achieving a desired accuracy.

5 Experiments

In order to evaluate the $\mathcal{AFCS}$ algorithm, we use three real-world datasets including the *Temperature (TEMP)*, *PM 2.5 air quality (PM25)*, and *Citi Bike trip (BIKE)* datasets, where the Experimental Setup section will cover all the settings and assumptions. Based on the above dataset, we first introduce the baseline algorithms which are commonly used in sensor data recovery. Specifically, the baseline algorithms adopt the ***matrix completion*** method and leverage the ***centralized computing*** patterns to recover the target sensing data. Then, we compare the performance of $\mathcal{AFCS}$ with baseline algorithms on three real-world datasets.

5.1 Experimental Setup

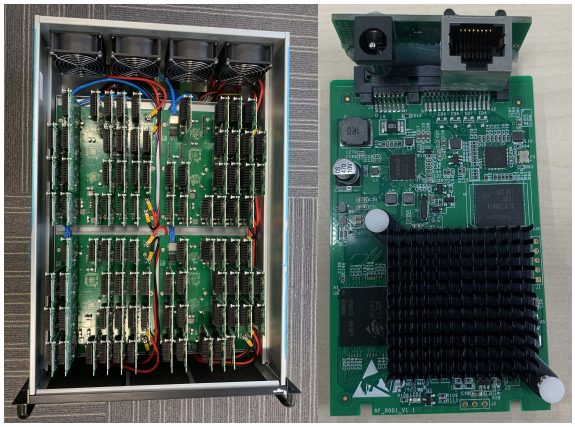


Fig. 3: Experiment Testbed.

5.1.1 Testbed Setup

To mimic the decentralized spatial-temporal compressive sensing environment, we establish a virtual mobile infrastructure (VMI), which is an implementation of ARMVM [39] technology. VMI has been used to host massive mobile clients on a server in the data center or the cloud (testbed in our experiments). Mobile clients are executed remotely and they are rendered via mobile optimized display protocols through networks. Specifically, we use the flagship SoC of Rockchip RK3399 [40] to virtualize mobile devices in the sensing network, where each RK3399 chip can carry on multiple instances of mobile clients (at most 10 instances). As shown in Fig. 3 left side, a single server has a large amount of slots to lunch multiple chips. Through setting up multiple clients, theoretically we can virtualize hundreds of participants for community sensing tasks simultaneously.

For the chip we adopt, RK3399 is packed with Dual Cortex-A72 and Quad Cortex-A53 and Mali-T860MP4 GPU, providing high computing and multi-media performance, rich interfaces and peripherals. The software of RK3399 supports a wide range of APIs which can provide us with fully implementation on real network traffic (e.g., message passing, broadcasting, and local updating lags). As shown in Fig. 3, right side is the bare RK3399 chip and left side shows the grouped RK3399 array on a server, where our testbed has multiple such kind of servers to emulate the large-scale real mobile sensing networks. Once the testbed has been set up, we reserve the 16MiB ARM TrustZone secure memory on a 4GB LPDDR4 RAM in each virtualized mobile device. Similar to the settings in [41], we run the local updating with full CPU frequency as these settings can influence the ARM TrustZone's performance [42]. Every chips was expected to run a Trusty [43] instance to emulate the execution with TEE for $\mathcal{AFCS}$. Certain network latency has been included to simulate real mobile traffics. Secure P2P connections have been established with various encryption strategies between any two chips.

5.1.2 Datasets Setup

For *TEMP* [44] and *PM25* [45] datasets, the sensing value of temperature (°C)/PM2.5 (air quality index) are located on each participant's mobile sensor in varying time slots (sensing cycle) and at different subareas. In detail, the *TEMP* dataset contains the temperature readings in 57 cells (Subareas), and each sensing cycle lasts for 30 minutes. The *PM25* dataset includes the PM2.5 air quality values on 36 stations (Subareas) with the same sensing cycle. For *BIKE* dataset [46], we adopt the Citi Bike trip histories in New York City on September 1st, 2019. We simulate the sensing task for observing the amount of in&out bikes from 200 bike stations in 24 hours and the time unit here is segmented as half an hour (48 time units per day). Specifically, at each station, the mobile devices record the number of outbound bikes as n_{out} and the number of inbound bikes as n_{in}, where the actual value in each cell (station) is represented as $b = n_{out} - n_{in}$. Since the value could be negative which is invalid when applying NMF, we calibrate the whole dataset with Min-max normalization as $\hat{b}_i = \frac{b_i - b_{min}}{b_{max} - b_{min}}$ at i^{th} station.

In order to simulate the settings of the centralized computing patterns, we aggregate the collected sensing data from each participant. In details, we follow the aforementioned three phases to set the appropriate value of four key factors: *the Number of Participants (m), the Number of Subareas that each participant covers in each sensing cycle, the Size of Windows (w)* and *the Size of Latent Space (l)*. Note that each participant can sense the temperature/PM2.5 at a subset of subarea. Specifically, we use the maximum number of subareas s ($1 \leq s \leq |S|$) in the experiments, assuming the participant can cover no more than s subareas. To simulate the scenario that each participant can cover various number of subareas,

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMC.2022.3178885, IEEE Transactions on Mobile Computing

the actual number of subareas covered by the participant will follow the discrete uniform distribution $\mathrm{U}\{1, s\}$.

5.2 Baseline Algorithms

In this section, we briefly introduce three baseline algorithms, where their advantages and drawbacks are listed as compared to $\mathcal{AFCS}$ algorithm.

- ***Spatio-Temporal Compressive Sensing (STCS)*** – *STCS* [12] leverages the sparsity regularized matrix factorization to fill in the missing values in a certain matrix accounting for spatial-temporal properties. Based on the low-rank nature of real-world data matrices, STCS first exploits global and subarea structures in the data metrics. Then, it recovers the original matrices through matrix factorization under spatial-temporal constraints. Indeed, STCS advances ideas from compressive sensing and provides a highly effective (high accuracy and robustness) approach to solve the problem of missing data interpolation.
- ***Robust Principle Component Analysis (RPCA)*** and ***Truncated Singular Value Decomposition (TSVD)*** – *RPCA* [20] is derived from a widely used statistical procedure of principal component analysis (PCA), where RPCA performs well on solving the problem of matrices recovering. With respect to a mass of missing observations, RPCA aims to recover a low-rank matrix through random sampling techniques [47].
  TSVD [21] is also commonly used to approximate a low-rank matrix. Different from the traditional singular value decomposition, TSVD sets all but the first k largest singular values equal to zero and uses only the first k columns of the corresponding unitary matrices. With the optimality property, this method provides an efficient way to recover the target sensing matrix.
- ***Fast Iterative Shrinkage-Thresholding Algorithm (FISTA)*** and ***Douglas–Rachford Splitting Algorithm (DRS)*** – *FISTA* is an upgraded extension of iterated shrinkage-thresholding algorithms (ISTA) [48] for solving linear inverse problems (e.g., matrix recovery problem) in signal or image processing. The family of *FISTA* methods, which can be viewed as an extension of the classical gradient algorithm, is popular due to its simplicity and robustness for solving large-scale problems even with dense matrix data. Here we adopt *FISTA* [22] as one baseline which shows a significantly better rate of convergence compared to the *ISTA*. For *DRS*, it is well-known as an operator splitting method that has been widely applied for solving a certain class of convex composite problems, which including the low-rank matrix recovery and sparse matrix completion problems in the compressive sensing domain. Although, variants of *DRS* have been proposed to handle the matrix completion problem such as parameterized DRS [49] and DRFS [50], we evaluate the vanilla *DRS* [23] for generalization in our settings.

5.3 Experimental Results

In this section, we report the performance of $\mathcal{AFCS}$ and other five baselines on *TEMP*, *PM25*, and *BIKE* datasets. Specifically, we use the *Absolute Error*, which is the averaged element-wise difference $\left(\sum_{a=1}^{|S|}\sum_{t=1}^{|T|}|\widehat{R}_{a,t} - R^*_{a,t}|/(|S| \cdot |T|)\right)$ between the recovered matrix ($\widehat{R}$) and the original data matrix (R^*), as the indicator of the performance.

5.3.1 Results on TEMP Dataset

First, we present a comparison of algorithms with the settings of the maximum number of subareas (covered by each participant) ranging from 1 to 5 in Fig. 4. Due to the overall better performances of $\mathcal{AFCS}$, CSWA, and STCS, we present the entire comparison in (a) and only compare $\mathcal{AFCS}$ and CSWA with STCS in other three settings (the same in Figures 5, 6 and 7 as well). Note that we draw a black horizontal line in each of the figures, which represents the simple averaging method – we calculate the mean value of the observed matrix to fill in the final recovered matrix – to show the performance gap[1] to other matrix completion methods. Specifically, in Fig. 4(a), 10 participants are involved. Then we vary the number of participants from 10 to 30 in the increment of 10 in Figs. 4(b), (c) and (d). We observe that the error is around 0.14 to 0.23 with varying maximum number of subareas from 1 to 5. It is noteworthy that $\mathcal{AFCS}$ can outperform CSWA and STCS with an obvious gap under these settings.

Second, we also compare $\mathcal{AFCS}$ with baseline algorithms by varying the number of participants in the secure P2P network. In Fig. 5(a), the maximum number of subareas is 1. Then we increase it from 1 to 3 in the increment of 1 in Figs. 5(b), (c) and (d). In each comparison between $\mathcal{AFCS}$, CSWA, and STCS, the error decreases when the number of participants increases for both of these three algorithms. This demonstrates that the larger group of participants can improve the performance of the matrix recovery, where intuitively, the participants can cover more subareas and sensing cycles. Similar to the previous setting, $\mathcal{AFCS}$ can achieve a slightly better performance comparing to CSWA and STCS as well.

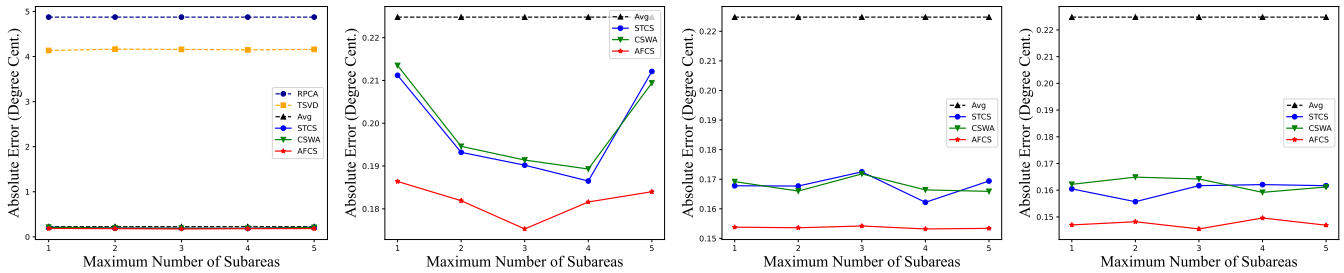
Further, we alter the values of two aforementioned key factors, such as *Size of Windows* and *Size of Latent Space*, to observe the variation of the error. Fig. 6 shows that the error decreases when the window size increases from 20 to 50. Note that for each size of latent space in Figs. 6(b), (c) and (d), the decreasing trends of the error are almost the same, and the performance of $\mathcal{AFCS}$ still can dominate CSWA and STCS. Fig. 7 exhibits that the error increases when the size of the latent space increases from 2 to 10. Thus, for *TEMP* datasets, the small size of latent space can better approximate the original data matrix when it is low-rank. Thus the performance of $\mathcal{AFCS}$ is still better than CSWA and STCS, as shown in Figs. 7(b), (c), and (d).

5.3.2 Results on PM25 Dataset

We conduct experiments with similar settings as *TEMP* datasets. Since the performances of RPCA, TSVD, are still

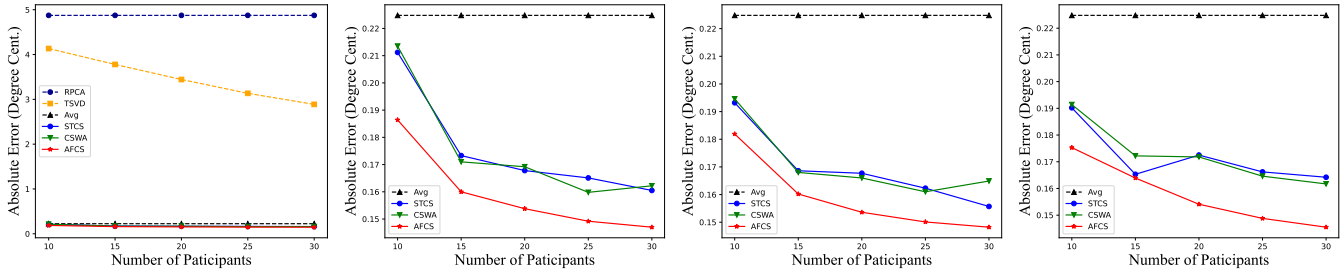
1. the simple averaging sometimes can obtain even better performance (less error) than matrix completion methods in specific settings and datasets.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMC.2022.3178885, IEEE Transactions on Mobile Computing



(a) $m = 10$ (Number of Participants) (b) $m = 10$ (Number of Participants) (c) $m = 20$ (Number of Participants) (d) $m = 30$ (Number of Participants)

Fig. 4: Performance Comparison with Varying *Maximum Number of Subareas* (s) per Participant per Cycle on *TEMP* Datasets.



(a) $s = 1$ (Maximum Num of Subareas) (b) $s = 1$ (Maximum Num of Subareas) (c) $s = 2$ (Maximum Num of Subareas) (d) $s = 3$ (Maximum Num of Subareas)

Fig. 5: Performance Comparison with Varying *Number of Participants* (m) on *TEMP* Datasets.

not as good as the other two algorithms, we only present the comparison between the proposed $\mathcal{AFCS}$, CSWA, and STCS here. Specifically, in TABLE. 1, we list the *Absolute Error* of these three algorithms with the varying numbers of participants (m) and the window size (w). When the number of participants increases, the error is decreasing intuitively. On the contrary, the error increases with the increased size of the window. However, $\mathcal{AFCS}$ performs comparably to STCS (the same as CSWA), sometimes even better (e.g., for $w = 20, w = 30, w = 40$). In TABLE. 2, we show the performance with varying sizes of latent space and the number of subareas covered by each participant. The results reveal that the number of subareas does not affect the error significantly, while with the larger latent space, the error is smaller with *PM25* datasets. Under these two settings, the performance of $\mathcal{AFCS}$ can still compete with STCS and CSWA. Note that for each setting, we present the performance on the varying factor while keeping the other factor at optimal value. Also, it is worth noting that the overall error is small on the average (10 with PM2.5 index ranging from 1 to 500) in all of $\mathcal{AFCS}$, CSWA, and STCS.

TABLE 1: Performance Comparison (*Absolute Error*) with Varying *Number of Participants* (m) and *Size of Windows* (w) on *PM25* Datasets.

	Number of Participants (m)			Size of Windows (w)		
	10	20	30	20	30	40
STCS	15.185	11.864	9.353	8.517	10.090	11.955
CSWA	15.563	11.686	9.561	8.844	10.232	12.028
$\mathcal{AFCS}$	15.366	11.984	9.375	8.372	9.872	11.228

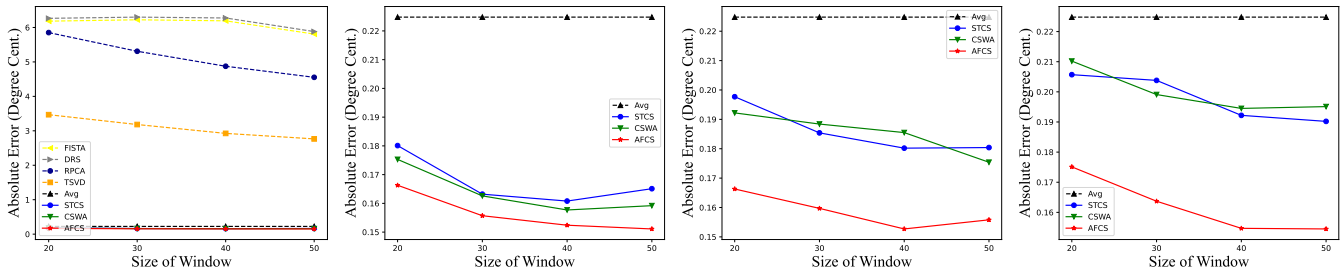
TABLE 2: Performance Comparison (*Absolute Error*) with Varying *Size of Latent Space* (l) and *Maximum Number of Subareas* (s) on *PM25* Datasets.

	Size of Latent Space (l)			Maximum Number of Subareas (s)		
	2	4	6	1	2	3
STCS	11.518	9.353	8.719	8.220	8.719	8.516
CSWA	11.777	9.561	8.945	8.166	8.945	8.844
$\mathcal{AFCS}$	11.362	9.757	8.481	8.169	8.667	8.372

5.3.3 Results on BIKE Dataset

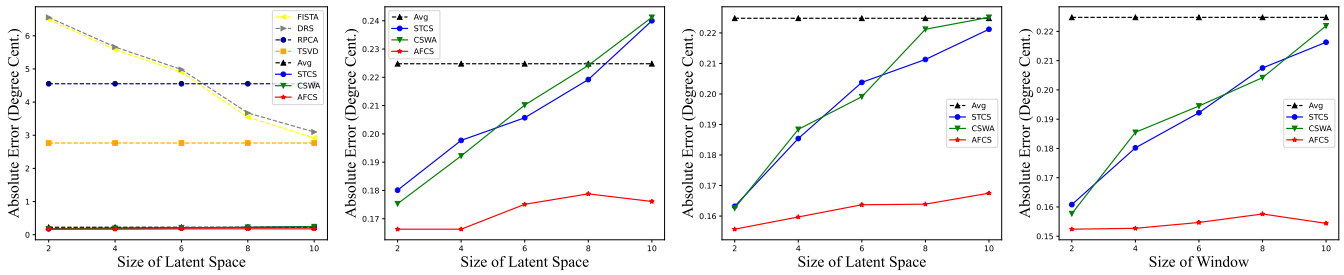
We further check the performance of baseline and proposed methods on *BIKE* dataset and show the results of only $\mathcal{AFCS}$ with CSWA and STCS due to the significant performance gap from the other four baselines. As presented in TABLE. 3, when the number of participants increases, the error is decreasing intuitively. Different from the *TEMP* and *PM25* datasets, the error does not show a significant increase (even slightly decrease) with the increased size of the window. This is caused by the nature of the dataset, where the sensing value for some cells (subareas) in sequential time slots could change drastically due to the peak of bike usage (i.e., value increase or decrease sharply) in *BIKE* dataset, which differs comparably lot from the other two datasets (i.e., stable sensing value in a small-time window). Yet, we can still observe that the $\mathcal{AFCS}$ can compete with STCS (and CSWA), sometimes marginally better. In TABLE. 4, we present the performance with varying sizes of latent space and the number of subareas covered by each participant. The results reveal that the number of subareas and the size of latent space in our selected range (in the same range as the other two datasets) does not affect the error significantly with *BIKE*

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMC.2022.3178885, IEEE Transactions on Mobile Computing



(a) $l = 2$ (Size of Latent Space) (b) $l = 2$ (Size of Latent Space) (c) $l = 4$ (Size of Latent Space) (d) $l = 6$ (Size of Latent Space)

Fig. 6: Performance Comparison with Varying *Size of Window* (w) on *TEMP* Datasets.



(a) $w = 20$ (Size of Windows) (b) $w = 20$ (Size of Windows) (c) $w = 30$ (Size of Windows) (d) $w = 40$ (Size of Windows)

Fig. 7: Performance Comparison with Varying *Size of Latent Space* (l) on *TEMP* Datasets.

datasets. Under such settings, the performance of $\mathcal{AFCS}$ can still compete with STCS and CSWA. Not surprisingly, the $\mathcal{AFCS}$ achieves more robust results (i.e., error with less variation as $Var \leq 0.1$) benefited from parallelism compare to the CSWA. As a city-level task to recover the number of bikes in each station, the scale of the approximation error is around 2.2 (best we achieved), which is acceptable in modern applications.

TABLE 3: Performance Comparison (*Absolute Error*) with Varying *Number of Participants* (m) and *Size of Windows* (w) on *BIKE* Datasets.

	Number of Participants (m)			Size of Windows (w)		
	10	30	50	20	30	40
STCS	8.781	2.655	2.207	3.507	3.369	2.207
CSWA	8.726	3.233	3.233	4.324	4.111	3.112
$\mathcal{AFCS}$	8.750	2.383	2.203	3.376	3.380	2.202

TABLE 4: Performance Comparison (*Absolute Error*) with Varying *Size of Latent Space* (l) and *Maximum Number of Subareas* (s) on *BIKE* Datasets.

	Size of Latent Space (l)			Maximum Number of Subareas (s)		
	3	5	7	1	2	3
STCS	2.247	2.264	2.218	2.218	2.290	2.207
CSWA	3.227	2.687	3.111	2.687	3.197	3.053
$\mathcal{AFCS}$	2.248	2.202	2.161	2.284	2.217	2.255

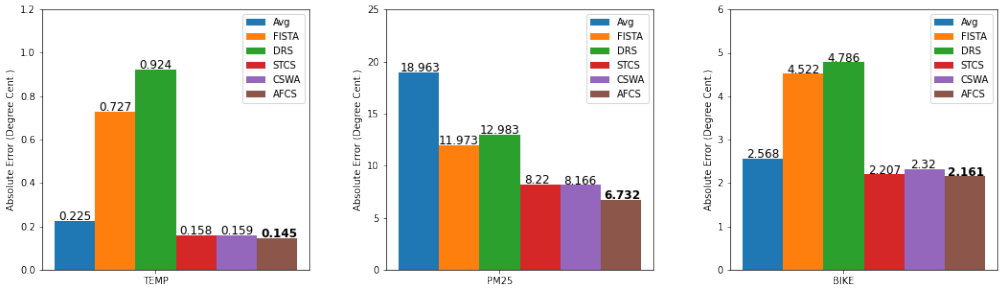
5.3.4 *Fine-tune Results*

To compare the overall performance of baselines and proposed algorithms straightforwardly, we carefully tune the hyper-parameters for each algorithm on each dataset and report the best performance we obtained in Fig. 8. The absolute errors of $\mathcal{AFCS}$ are $0.145°$ in city-wide temperature sensing, 6.732 units of PM2.5 index in urban air pollution sensing and 2.161 bikes in city-wide bike sharing prediction. As we can observe, $\mathcal{AFCS}$ can compete with STCS on all three datasets. Also, compared to the CSWA, $\mathcal{AFCS}$ shows a marginally better performance. The fine-tuned corresponding hyper-parameters of $\mathcal{AFCS}$ are revealed as follows,

- TEMP – thread size: 7, the max number of subareas: 1, window size: 40, the number of participants: 25, latent space size: 2.
- PM25 – thread size: 40, the max number of subareas: 1, window size: 20, the number of participants: 30, latent space size: 1.
- BIKE – thread size: 20, the max number of subareas: 1, window size: 40, the number of participants: 50, latent space size: 7.

Remarks. Based on the performance analyses on varying value of four key parameters, we categorize them into three groups – insensitive, positively related, and negatively related. As shown in our series of experiments (Fig. 4-7, and TABLE 1-4), the maximum number of subareas (denoted as $|S|$) is the only one insensitive factor, where the error rates on three real-world datasets are rarely affected by this parameter. For the number of participants (denoted as m) and the window size (denoted as w), the curve of the error rate is in a downward trend (in certain range at least), which shows

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMC.2022.3178885, IEEE Transactions on Mobile Computing



(a) TEMP dataset (b) PM25 dataset (c) BIKE dataset

Fig. 8: Fine-tuned Performance Comparison

that m and w are negatively related to the reconstruction error rate. As the last factor, the size of latent space (denoted as l) is obviously in the positively related group. Note that the m, w, and l have the convergence points on three datasets, where we explore the combination of these four parameters to achieve the overall fine-tune results.

For applying the fine-tune results in practice, the communication efficiency could be one of the important factors, which affects the decision making (the choosing of the key parameters) to use AFCS. Similar to the group splitting in terms of lowering the error rate, the four key parameters which have different impacts on the communication complexity should be treated as well. Since increasing the value of all four key parameters can bring positive impact on the communication complexity, we intend to choose smaller values of these parameters and keep an acceptable error rate. For choosing the parameter $|S|$, it is obvious that 1 should be an appropriate value for both the communication and the error rate, where we have the same conclusion on three datasets. While for the parameter l, w, t_max, and m, all are positively related the communication complexity. Recalling the convergence rate as an order of the number of iterations in Section 4.4 (assuming the number of threads is the same as m), we can reform the communication complexity as $\mathcal{O}\left(\frac{\mu \cdot l^2 \cdot (w+1)^2 \cdot m^2}{\eta}\right)$, where η is the target error rate and μ is the constant factor determined by the data sample [37]. Thus, to balance the communication complexity and error rate, we need to follow the above summarized policy, where lowering a target error rate sometimes takes a tremendous of additional efforts when those basic factors (i.e., l, w, and m) are considerably large. The usage of AFCS should make a trade-off especially for a large-scale mobile network.

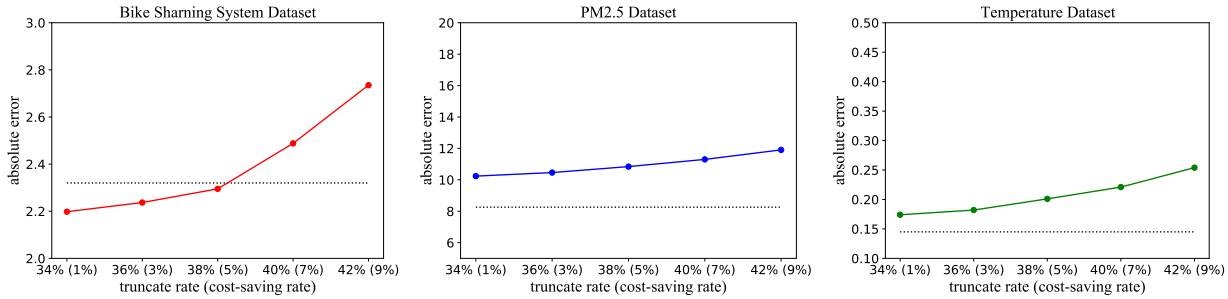
5.4 Ablation Studies on Truncation Operation

We additionally report the effectiveness of truncation in Algorithm 2. Recalling the definition of the *truncation()* function, we artificially reset specific cells of the matrix which are less than a threshold to zero. The major motivations are (1) setting negative values to zero to enable the local NMF updating, (2) incorporating a certain sparsity to improve the found decomposition [51], and (3) saving storage and communication cost when loading and passing large-scale P, Q matrix via compressed sparse row (CSR) [35] format. As shown in Fig. 9, we investigate the relationship between approximation error with the "truncate rate", where we define it as the number of the truncated cells divided by the total number of cells in the target matrix. Since the cost-saving occurs when the truncate rate is greater than about 33% according to the CSR format, we only report the approximation error above this boundary. Note that the black dash line indicates the approximation error without truncation so that we can acknowledge the actual performance gain or loss (trade-off) when applying truncation. For simplicity, we use the aforementioned best hyper-parameters to build the algorithm on three datasets for truncation experiments. Fig. 9(a) demonstrates that an appropriate truncation rate (less than 38.2%) could reduce the error of approximation and also helps save the storage and communication cost related to the P, Q matrix processing. For Fig. 9(b)(c), the error line is above the borderline when the truncation rate is greater than 33% and gradually increases if more cells of the matrix are truncated. In such a case, we need to compromise with the recovery error sometimes in order to save the expected amount of the cost.

Sparsity. Basically, the error rate curve is in an "U" shape when the sparsity of P, Q metrics increases in a certain range. The range of U shape depends on the characteristics of the dataset, where in our experiments, the turning points for three real-world datasets are around 30%. Note that we adjust the sparsity by changing the ratio of truncation beyond the scale of the sparsity of original data from each participant. As shown in Fig. 10, we first draw a line of base error rate when sparsity of P, Q metrics is 33%. Then, we show the error rate increases along with the growth of truncation rate (sparsity). Since we intend to show the cost saving rate (the critical point is $1/3$ for sparsity according to the compressed sparse row format), we only present the error rate variation when sparsity is greater than 33%.

Note that, when applying the truncation in applications with a large or extremely large dataset (with corresponding large P, Q metrics), the cost saving is still significant compared to the slightly greater error. As an example, for PM25 dataset, achieving 9% of cost-saving rate could bring about 3 megabyte of actual storage cost, where the full dataset contains 22 monitoring stations, and 23300 hour time series from August 24th, 2012 to March 8th, 2013. Whereas the degradation on error is only 1 unit from 10 unit to 11 unit on average. Thus, we can conclude that the reasonable trade-offs could be made depending on specific datasets, and the

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMC.2022.3178885, IEEE Transactions on Mobile Computing



(a) BIKE Dataset (b) PM25 Dataset (c) TEMP Dataset

Fig. 9: The Performance (Absolute Error) of $\mathcal{AFCS}$ with Varying Ratio of Truncation (Cost-Saving Rate).

proposed $\mathcal{AFCS}$ shows promising practicability that the recovery error is not very sensitive to the truncation rate for a wide range in most cases.

5.5 Ablation Studies on Lock-Free Speed Up

One of the most critical characteristics of $\mathcal{AFCS}$ is breaking the bottleneck of conflict when passing and processing the message in the network due to a design of steering pool. As we introduced, we observe that a specific participant has to wait until the current thread's task is finished, while this causes a locked status which may have a significant impact on the progress of other threads when they frequently request one participant. To fully recur such a scenario and demonstrate the effectiveness of the designed steering pool, we experiment to check the speed-up ratio of the varying size of thread and varying ratio of calculating cost divided by passing cost comparing to the $\mathcal{AFCS}$ without steering pool.

Since the speed-up ratio is affected by several factors, we first introduce two key influence factors in community sensing, which are the communication cost in decentralized/parallel mobile computing networks and the local updating cost on each of the mobile devices. For the communication cost, we need to estimate the additional cost for encryption when passing the P, Q matrix in our settings. As shown in Fig. 10, we test the actual storage cost and time cost of four common used encryption algorithms including AES [52], RSA [53], hybrid AES+RSA [54], and ECC [55] on the P, Q matrix (CSR format) with varying size of latent spaces. To guarantee the equal strength of cryptographic for all four algorithms, we generate encryption keys with different lengths according to the TABLE. 5 from NAS[2]. Most cryptographic experts [56], [57] recommend that current systems offer at least 128 bits of security (selected in our experiments). For example, it is generally thought that 128 bits of security can be achieved with 128-bit AES keys, 256-bit ECC keys, and 3072-bit RSA keys. In our experiment, ignoring the implementation issues, all four algorithms we adopt with specified key lengths will generally have the same level of security.

2. National Security Agency – "Commercial National Security Algorithm Suite", 2015

TABLE 5: Security comparison for various algorithm key size combinations.

Security Bits	AES	RSA	ECC	MIPS Years of Protection
80	80	1024	160	2010
112	112	2048	224	2030
128	128	3072	256	2040
192	192	7680	384	2080
256	256	15360	512	2120

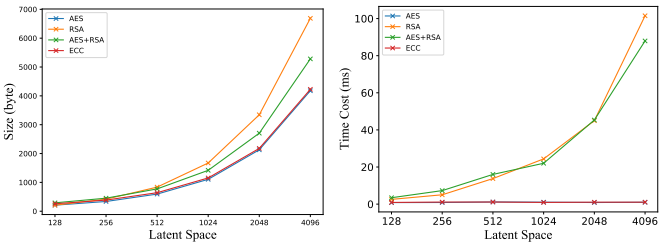


Fig. 10: Encryption Efficiency

When it comes to performance at 128-bit security levels, RSA is generally reported to be 10 times slower than ECC for private key operations such as signature generation or key management. We estimate all the corresponding operations cost rather than only key size and key generation time in three real-world datasets, where the actual implementation cost is reported. As we can observe, Fig. 10(a) presents the storage size of the encryption message when the size of latent space increases. The curve shows exponential increases of the storage size for all algorithms, while ECC and AES consume slightly small size of storage. For time cost, Fig. 10(b) shows that ECC and AES take much less time to process the encryption compared to RSA and hybrid algorithms. Since the RSA dominates the key generation time cost (has a much longer key than AES's), the curve of the hybrid algorithm has a similar performance in the figure. Note that all four algorithms are available in $\mathcal{AFCS}$ framework as the options for the user's preference and specific applications. Since the ECC presents a competitive performance in our experiment (mobile spatial-temporal data encryption), we adopt ECC as the baseline encryption method for the following speed-up ratio experiment.

For another part of the main cost – local updating cost, we calculate the processing time of Algorithm 2 in the real-

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMC.2022.3178885, IEEE Transactions on Mobile Computing

world mobile sensing environment. Then we combine these two parts to conduct the overall speed-up ratio experiments in Fig. 11. Specifically, we test with varying sizes of thread to check the speed-up ratio of $\mathcal{AFCS}$ comparing to the $\mathcal{AFCS}$ without steering pool strategy. Since the actual cost for Calculating (local updating) and Passing (communication) are fluctuated based on the encryption algorithm adopted and the variation of implementations (all other miscellaneous), we set a series of proportions of calculating cost divided by passing cost, ranging from $1/8$ to 8. For consistency, we fix the number of participants and passing times as 100. As shown in four curves, the speed-up effect becomes significant when the size of the thread increases and the Calculating cost is much greater than the Passing cost. It is reasonable that when the Calculating cost is less than or close to the Passing cost, the blocking on a mobile device from queuing for multiple requests in mutual networks is alleviated. However, in real-world mobile sensing networks, especially with complicated models and large-scale datasets, local updating still dominant the overall time consumption through the whole process of sensing tasks. Thus, the steering pool strategy under the settings of multi-thread is necessary and could bring us a considerable efficiency improvement in communication. We also report the acceleration of $\mathcal{AFCS}$ with fine-tuned parameters on three datasets compared to the version without steering pool strategy. As we can observe, $\mathcal{AFCS}$ shows a great speed-up ($2\times$) on PM25 data set. Note that, notwithstanding the slight efficiency improvement brought by steering pool strategy on TEMP (with 20 threads) and BIKE (with 7 threads) datasets[3], we can still sacrifice an acceptable amount of recovery accuracy to achieve a huge scaling performance in real-world applications. For example, we can raise the thread size up to 80 to obtain a $4.5\times$ speed-up on PM25 data set while only suffering a tiny degradation on accuracy (increase the error from 6.732 to 7.454).

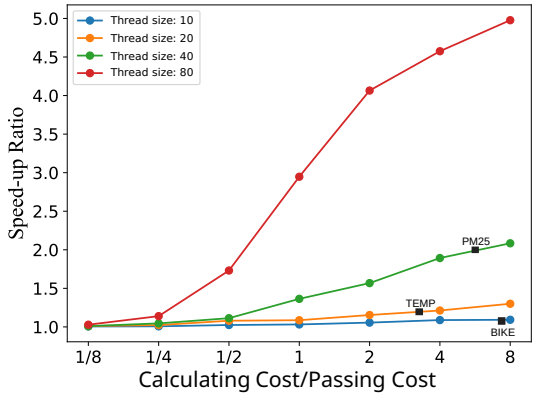


Fig. 11: The speed-up ratio of the real mobile sensing networks for varying size of thread in 100 participants. The iteration number is fixed to 100.

Finally, we further investigate the effect of varying thread sizes on recovery error. As shown in Fig. 12, we test the $\mathcal{AFCS}$ with the increased size of threads on the TEMP and BIKE dataset. The error curves for both datasets present decreasing trends when the number of threads increases, which indicates that with a suitable setting of thread size, $\mathcal{AFCS}$ can achieve a sub-optimal compared to the single thread.

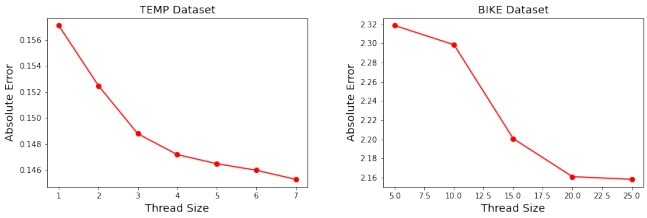


Fig. 12: The performance improvement with increasing number of threads.

5.6 Summary of Experiments

With three real-world datasets, we compared the proposed $\mathcal{AFCS}$ with the baseline algorithms STCS, RPCA, TSVD, FISTA, DRS, and CSWA. For all of the datasets, $\mathcal{AFCS}$ significantly outperforms RPCA, TSVD, FISTA, and DRS in most cases. Moreover, compared to the centralized algorithm STCS and single-thread algorithm CSWA, $\mathcal{AFCS}$ also presents its competitiveness, with a low approximation error (0.2° in city-wide temperature, 10 units of PM2.5 index in urban air quality, and 2 bikes in city-wide bike-sharing systems). Even in some settings, the $\mathcal{AFCS}$ has a lower approximation error than STCS and CSWA, which demonstrates the advantages of $\mathcal{AFCS}$.

Beyond the performance on error, we also conduct experiments further to check the communication efficiency of $\mathcal{AFCS}$. As we introduced in the framework design part, a variety of factors, including additional security techniques, truncation, and steering pool strategy, are taken into consideration in the implementation on real mobile sensing environments. Although there is no perfect setting to balance both the recovery accuracy and the communication efficiency, we demonstrate the adaptability and scalability of $\mathcal{AFCS}$ for large-scale datasets. The overall performance of $\mathcal{AFCS}$ is still promising in most scenarios, and the $\mathcal{AFCS}$ supports the customized adjustments to tackle the trade-off between error rate and communication efficiency.

6 Related Work and Discussion

In this section, we summarize the related work which motivates us to propose $\mathcal{AFCS}$ and compare $\mathcal{AFCS}$ with our previous studies. Then, we further discuss the open issues which are remained to be discovered and addressed.

6.1 Related Works

Compressive sensing theory [58], has been proved to be a powerful tool to reconstruct a sparse vector or matrix based on the sparsity property of vector or low-rank property of matrix. In this case, a large number of applications based on compressive sensing have been proposed, such as network traffic reconstruction [59], environmental data recovery [60], road traffic monitoring [61], and face recognition

3. The detailed parameter settings are revealed in the previous Fine-tune Results section.

on smartphones [62]. The above work primarily focuses on designing effective algorithms to minimize the reconstruction error under different scenarios, while another branch of works intends to minimize the number of the allocated tasks and meet a predefined data quality requirement. The representative works include CCS-TA [12], BCCS [63], and SPACE-TA [64].

Besides compressive sensing, there still exist other state-of-the-art methods to infer missing values for uncovered/unreported cells, such as multi-channel singular spectrum analysis [65] and expectation maximization [25]. Deep learning-based generative models such as [66], [67], [68] shows a great potential on the spatial-temporal environment monitoring with relatively higher precision.

However, almost all of the above-mentioned methods are centralized and depend on data aggregation as the prerequisite, while it is becoming unrealistic to gathering all the data to a single server since data privacy gains much more focus in modern computing platforms and data aggregation will increase the risk of private information leakage. And for deep-learning based model, the performance is degraded since the mobile device has limited computation power and cannot fully implement the complicated and deep structures of generative models. Inspired by the Federated learning [69] and TEE technology [70], we design a communication-efficient community sensing framework that avoids data aggregation and provides additional security compared to previous works.

Comparison with the previous version. The proposed $\mathcal{AFCS}$ is an extension of the previous work *CSWA* [24]. Specifically, we mainly target the following four novel parts to improve the performance: (1) To further accelerate the optimization process of the previous version *CSWA*, we upgrade the original framework to a parallelized one, where the multiple threads take the place of single thread and add supported algorithms to make it happen. (2) We strengthen the security level of the whole framework from two main aspects: local update and communication. For local updates, we establish a TEE for each of the participants in the predefined mutual trust network. For communication, we provide a variety of encryption techniques to protect the message passing. For both of them, we conduct the corresponding experiments to analyze the efficacy and efficiency. (3) We also analyze the new communication complexity of $\mathcal{AFCS}$, and introduce the parallelized stochastic gradient descent, which is the foundation of $\mathcal{AFCS}$. (4) According to the changes above in algorithms and theoretical properties, we conduct intensive complementary experiments, which include threads impact experiments, new performance comparison experiments among the baseline algorithm on three real-world datasets. The extended results are shown in the new Experiments section.

6.2 Discussion on Open Issues

For the open issues that have not been fully solved or studied in this work, we divide them into three respects as follows:

Optimization Algorithms. In this research, we adopted gossip-based parallel stochastic gradient descent to solve the distributed optimization problem for decentralized spatial-temporal compressive sensing. However, some other advanced optimization methods can obtain better performance. For example, De and Goldstein have proposed an efficient distributed stochastic gradient descent algorithm with variance reduction [71], Qian and Tan formulate the problem as an optimization of distributed coordinate gradient descent [72]. We can also extend the proposed algorithm $\mathcal{AFCS}$ on non-convex optimization functions [73], [74].

Parallelism and Independent Updating. In the design of $\mathcal{AFCS}$, we leverage the completely independent and parallel updating scheme with the steering pool strategy, where participants in any thread do not affect each other when updating P, Q matrix. The number of threads also has a positive correlation with the recovery accuracy in specific ranges as shown in Fig. 12. For a predefined thread of tasks, each participant in principle should wait for all threads' final broadcasting results, while in practice, participants can process the reconstruction in an asynchronous way if there is a strict time limit. We intend to research on partially independence to increase the flexibility and efficiency of updating, which is motivated by the asynchronous SGD [75], [76].

Energy Consumption. The focus of this work is to explore a possible decentralized community sensing strategy with the design of protecting local privacy information. Since we lunch the experiments on the testbed, a virtual mobile infrastructure, we can only mimic the communication and calculation cost for AFCS rather than the actual energy consumption in mobile devices. Unlike the training of Deep Learning models, the proposed AFCS adopts a lightweight NMF as the basis and updates the matrix using the parallel stochastic gradient descent. For each time of local update on one participant (mobile device), there is only one time of gradient calculation taken place so that the energy consumption is considerably less than most of the other optimization routines (e.g., several epochs of deep training). Considering the limited battery volume on mobile device, the calculation could process passively when the mobile device is in a charging mode and even possibly work in the background when the mobile user actively enables this task. Furthermore, we can leverage some incentives allocation mechanisms [77], [64] to motivate the mobile user to participate (with their mobile device) in the sensing task.

For the communication and calculation cost of AFCS, we introduce ablation studies in Section 5.5 and show the priority of our designed lock-free mechanism, which speeds up the overall sensing process. The costs (i.e., duration of communication, encryption, and calculation) are estimated through our testbed with virtual nodes represented as participating mobile devices. Since this work mainly targets algorithm-level applicability, the energy consumption on virtual testbeds differs from the real mobile sensing network, which is not worthy of being reported. We would like to further research the energy consumption estimation of real mobile devices in future work, where a real experimental environment with mainstream mobile devices awaits to be established.

Federated Machine Learning. Federated machine learning (FedML) promises to be a cornerstone technology in Decentralized AI. A current open challenge in FedML is the secure and trusted execution of machine learning code at local clients. FedML provides strong input privacy by enabling the training of machine learning models without moving

and disclosing data. Much research is currently invested into security and FedML, for example to guard against attacks on the training data itself with the purpose of biasing the model, so-called data-poisoning attacks [78], or reverse-engineering of training data using predictions from the federated ML model, so called inference attacks [79]. These are general problems in adversarial ML and can be mitigated using general approaches also in a federated setting. The proposed $\mathcal{AFCS}$ aims to develop and critically evaluate a pilot solution to this problem based on TEE and encrypted message passing. Compared to the CSWA and other baselines spatial-temporal matrix completion algorithms, we additionally enhance the security of the community sensing and message passing.

An open problem that is specific to FedML due to its intrinsic distributed nature is the threat associated with accidental or malicious modification of the client itself (identity), or the code run by clients when computing model updates (remote computation). If a dishonest member of the federation, an external attacker, or a misconfigured device is able to modify the training code, the consequences can range from mild such as affected convergence rate in training, to backdoor attacks rendering the model useless. Solutions that can help ensure the veracity of client execution have a great potential to help the widespread adoption of federated learning technologies for decentralized AI.

7 Conclusion

In this paper, we proposed $\mathcal{AFCS}$ – a novel community sensing paradigm. $\mathcal{AFCS}$ is designed to extract the environmental information in each subarea, without aggregating sensor and location data from the participants who partially cover the monitored area. We incorporate TEEs in each participant's mobile device to secure the storage, computation and communication of spatial-temporal data. On top of TEEs, $\mathcal{AFCS}$ establishes mutual trust networks with encrypted P2P connections. Then, $\mathcal{AFCS}$ proposes a novel Decentralized Spatial-Temporal Compressive Sensing framework based on Parallelized Stochastic Gradient Descent. Specifically, through learning the low-rank matrix structure via distributed optimization, $\mathcal{AFCS}$ approximates the value of sensor data in each subarea for each sensing cycle using the sensor data that locally stored in each participant's TEE instance. According to the theoretical analysis on the parallelized stochastic gradient decent [36], $\mathcal{AFCS}$ is capable of recovering the Spatial-Temporal information with bounded approximation error using the mutual trust communications of controllable complexity. The experiment results based on three real-world datasets demonstrates that $\mathcal{AFCS}$ has low approximation error and performs comparably to (sometimes even better than) state-of-the-art algorithms based on the data aggregation and centralized computation. The additional experiments on efficacy and efficiency further exhibit the adaptability and scalability of $\mathcal{AFCS}$.

Acknowledgement

An earlier version of this manuscript has been published as "CSWA: Aggregation-free spatial-temporal community sensing" [24], in Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 32, 2018.

References

[1] D. Zhang, L. Wang, H. Xiong, B. Guo, 4w1h in mobile crowd sensing, IEEE Communications Magazine 52 (8) (2014) 42–48.
[2] S. Ji, Y. Zheng, T. Li, Urban sensing based on human mobility, in: Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing, ACM, 2016, pp. 1040–1051.
[3] J. Yoon, B. Noble, M. Liu, Surface street traffic estimation, in: Proceedings of the 5th international conference on Mobile systems, applications and services, ACM, 2007, pp. 220–232.
[4] R. Herring, A. Hofleitner, S. Amin, T. Nasr, A. Khalek, P. Abbeel, A. Bayen, Using mobile phones to forecast arterial traffic through statistical learning, in: 89th Transportation Research Board Annual Meeting, 2010, pp. 10–14.
[5] S. Hachem, A. Pathak, V. Issarny, Probabilistic registration for large-scale mobile participatory sensing, in: Pervasive Computing and Communications (PerCom), 2013 IEEE International Conference on, IEEE, 2013, pp. 132–140.
[6] B. Pan, Y. Zheng, D. Wilkie, C. Shahabi, Crowd sensing of traffic anomalies based on human mobility and social media, in: Proceedings of the 21st ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, ACM, 2013, pp. 344–353.
[7] K. Aberer, S. Sathe, D. Chakraborty, A. Martinoli, G. Barrenetxea, B. Faltings, L. Thiele, Opensense: open community driven sensing of environment, in: Proceedings of the ACM SIGSPATIAL International Workshop on GeoStreaming, ACM, 2010, pp. 39–42.
[8] T. Liu, Y. Zheng, L. Liu, Y. Liu, Y. Zhu, Methods for sensing urban noises, Tec. Rep. MSR-TR-2014-66.
[9] H. Xiong, D. Zhang, L. Wang, H. Chaouchi, Emc 3: Energy-efficient data transfer in mobile crowdsensing under full coverage constraint, IEEE Transactions on Mobile Computing 14 (7) (2015) 1355–1368.
[10] X. Sheng, J. Tang, W. Zhang, Energy-efficient collaborative sensing with mobile phones, in: INFOCOM, 2012 Proceedings IEEE, IEEE, 2012, pp. 1916–1924.
[11] D. Zhang, H. Xiong, L. Wang, G. Chen, Crowdrecruiter: selecting participants for piggyback crowdsensing under probabilistic coverage constraint, in: Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing, ACM, 2014, pp. 703–714.
[12] 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.
[13] L. Wang, D. Zhang, Y. Wang, C. Chen, X. Han, A. M'hamed, Sparse mobile crowdsensing: challenges and opportunities, IEEE Communications Magazine 54 (7) (2016) 161–167.
[14] L. Wang, D. Zhang, P. Wang, C. Animesh, X. Han, W. Xiong, Space-ta: Cost-effective task allocation exploiting intra- and inter-data correlations in sparse crowdsensing, ACM Transactions on Intelligent Systems and Technology.
[15] L. Wang, D. Yang, X. Han, T. Wang, D. Zhang, X. Ma, Location privacy-preserving task allocation for mobile crowdsensing with differential geo-obfuscation, in: Proceedings of the 26th International Conference on World Wide Web, International World Wide Web Conferences Steering Committee, 2017, pp. 627–636.
[16] 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.
[17] S. Pinto, N. Santos, Demystifying arm trustzone: A comprehensive survey, ACM Computing Surveys (CSUR) 51 (6) (2019) 1–36.
[18] J.-E. Ekberg, K. Kostiainen, N. Asokan, Trusted execution environments on mobile devices, in: Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security, 2013, pp. 1497–1498.
[19] I. Hegedűs, G. Danner, M. Jelasity, Decentralized learning works: An empirical comparison of gossip learning and federated learning, Journal of Parallel and Distributed Computing 148 (2021) 109–124.
[20] H. Gao, J.-F. Cai, Z. Shen, H. Zhao, Robust principal component analysis-based four-dimensional computed tomography, Physics in medicine and biology 56 (11) (2011) 3181.
[21] S. Isam, I. Kanaras, I. Darwazeh, A truncated svd approach for fixed complexity spectrally efficient fdm receivers, in: Wireless Communications and Networking Conference (WCNC), 2011 IEEE, IEEE, 2011, pp. 1584–1589.

[22] A. Beck, M. Teboulle, A fast iterative shrinkage-thresholding algorithm for linear inverse problems, SIAM journal on imaging sciences 2 (1) (2009) 183–202.

[23] S. Li, H. Qi, A douglas–rachford splitting approach to compressed sensing image recovery using low-rank regularization, IEEE Transactions on Image Processing 24 (11) (2015) 4240–4249.

[24] J. Bian, H. Xiong, Y. Fu, S. Das, Cswa: Aggregation-free spatial-temporal community sensing, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 32, 2018.

[25] T. Schneider, Analysis of incomplete climate data: Estimation of mean values and covariance matrices and imputation of missing values, Journal of Climate 14 (5) (2001) 853–871.

[26] D. Kondrashov, M. Ghil, Spatio-temporal filling of missing points in geophysical data sets, Nonlinear Processes in Geophysics 13 (2) (2006) 151–159.

[27] L. Kong, M. Xia, X.-Y. Liu, M.-Y. Wu, X. Liu, Data loss and reconstruction in sensor networks, in: INFOCOM, 2013 Proceedings IEEE, IEEE, 2013, pp. 1654–1662.

[28] Y. Mao, L. K. Saul, Modeling distances in large-scale networks by matrix factorization, in: Proceedings of the 4th ACM SIGCOMM conference on Internet measurement, ACM, 2004, pp. 278–287.

[29] Y. Aono, T. Hayashi, L. Wang, S. Moriai, et al., Privacy-preserving deep learning via additively homomorphic encryption, IEEE Transactions on Information Forensics and Security 13 (5) (2017) 1333–1345.

[30] L. Melis, C. Song, E. De Cristofaro, V. Shmatikov, Exploiting unintended feature leakage in collaborative learning, in: 2019 IEEE Symposium on Security and Privacy (SP), IEEE, 2019, pp. 691–706.

[31] R. Shokri, M. Stronati, C. Song, V. Shmatikov, Membership inference attacks against machine learning models, in: 2017 IEEE Symposium on Security and Privacy (SP), IEEE, 2017, pp. 3–18.

[32] C.-C. Tsai, K. S. Arora, N. Bandi, B. Jain, W. Jannen, J. John, H. A. Kalodner, V. Kulkarni, D. Oliveira, D. E. Porter, Cooperation and security isolation of library oses for multi-process applications, in: Proceedings of the Ninth European Conference on Computer Systems, 2014, pp. 1–14.

[33] S. Zhao, Q. Zhang, Y. Qin, W. Feng, D. Feng, Sectee: A software-based approach to secure enclave architecture using tee, in: Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, 2019, pp. 1723–1740.

[34] B. Langenstein, R. Vogt, M. Ullmann, The use of formal methods for trusted digital signature devices., in: FLAIRS Conference, 2000, pp. 336–340.

[35] A. Buluç, J. T. Fineman, M. Frigo, J. R. Gilbert, C. E. Leiserson, Parallel sparse matrix-vector and matrix-transpose-vector multiplication using compressed sparse blocks, in: Proceedings of the twenty-first annual symposium on Parallelism in algorithms and architectures, 2009, pp. 233–244.

[36] M. Zinkevich, M. Weimer, L. Li, A. J. Smola, Parallelized stochastic gradient descent, in: Advances in neural information processing systems, 2010, pp. 2595–2603.

[37] L. Bottou, O. Bousquet, The tradeoffs of large scale learning, Advances in neural information processing systems 20.

[38] J. E. Dennis Jr, R. B. Schnabel, Numerical methods for unconstrained optimization and nonlinear equations, SIAM, 1996.

[39] S.-C. Oh, K. Kim, K. Koh, C.-W. Ahn, Vimo (virtualization for mobile): a virtual machine monitor supporting full virtualization for arm mobile systems, Proc. Advanced Cognitive Technologies and Applications, COGNITIVE.

[40] https://github.com/rockchip linux, Edge specs | rockchip rk3399 datasheet | linux development board, Github.

[41] F. Mo, H. Haddadi, K. Katevas, E. Marin, D. Perino, N. Kourtellis, Ppfl: privacy-preserving federated learning with trusted execution environments, arXiv preprint arXiv:2104.14380.

[42] J. Amacher, V. Schiavoni, On the performance of arm trustzone, in: IFIP International Conference on Distributed Applications and Interoperable Systems, Springer, 2019, pp. 133–151.

[43] Google, https://source.android.com/security/trusty/index.html, Trusty TEE, 2016.

[44] F. Ingelrest, G. Barrenetxea, G. Schaefer, M. Vetterli, O. Couach, M. Parlange, Sensorscope: Application-specific sensor network for environmental monitoring, ACM Transactions on Sensor Networks (TOSN) 6 (2) (2010) 17.

[45] Y. Zheng, F. Liu, H.-P. Hsieh, U-air: When urban air quality inference meets big data, in: Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining, ACM, 2013, pp. 1436–1444.

[46] https://www.citibikenyc.com/system-data, Citi bike trip histories.

[47] M. A. Fischler, R. C. Bolles, Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography, Communications of the ACM 24 (6) (1981) 381–395.

[48] J. M. Bioucas-Dias, M. A. Figueiredo, A new twist: Two-step iterative shrinkage/thresholding algorithms for image restoration, IEEE Transactions on Image processing 16 (12) (2007) 2992–3004.

[49] F. Bian, X. Zhang, A parameterized douglas–rachford splitting algorithm for nonconvex optimization, Applied Mathematics and Computation 410 (2021) 126425.

[50] F. J. A. Artacho, J. M. Borwein, M. K. Tam, Douglas–rachford feasibility methods for matrix completion problems, The ANZIAM Journal 55 (4) (2014) 299–326.

[51] P. O. Hoyer, Non-negative matrix factorization with sparseness constraints., Journal of machine learning research 5 (9).

[52] J. Daemen, V. Rijmen, Reijndael: The advanced encryption standard., Dr. Dobb's Journal: Software Tools for the Professional Programmer 26 (3) (2001) 137–139.

[53] L. M. Kohnfelder, Towards a practical public-key cryptosystem., Ph.D. thesis, Massachusetts Institute of Technology (1978).

[54] V. S. Mahalle, A. K. Shahade, Enhancing the data security in cloud by implementing hybrid (rsa & aes) encryption algorithm, in: 2014 International Conference on Power, Automation and Communication (INPAC), IEEE, 2014, pp. 146–149.

[55] F. Amounas, E. El Kinani, Ecc encryption and decryption with a data sequence, Applied mathematical sciences 6 (101) (2012) 5039–5047.

[56] M. Alimohammadi, A. Pouyan, Performance analysis of cryptography methods for secure message exchanging in vanet, International Journal of Scientific & Engineering Research 5 (2) (2014) 912.

[57] P. G. Argyroudis, R. Verma, H. Tewari, D. O'Mahony, Performance analysis of cryptographic protocols on handheld devices, in: Third IEEE International Symposium on Network Computing and Applications, 2004.(NCA 2004). Proceedings., IEEE, 2004, pp. 169–174.

[58] E. J. Candès, B. Recht, Exact matrix completion via convex optimization, Foundations of Computational mathematics 9 (6) (2009) 717–772.

[59] Y. Zhang, M. Roughan, W. Willinger, L. Qiu, Spatio-temporal compressive sensing and internet traffic matrices, in: ACM SIGCOMM Computer Communication Review, Vol. 39, ACM, 2009, pp. 267–278.

[60] P. Zhang, S. Wang, K. Guo, J. Wang, A secure data collection scheme based on compressive sensing in wireless sensor networks, Ad Hoc Networks 70 (2018) 73–84.

[61] Y. Zhu, Z. Li, H. Zhu, M. Li, Q. Zhang, A compressive sensing approach to urban traffic estimation with probe vehicles, IEEE Transactions on Mobile Computing 12 (11) (2012) 2289–2302.

[62] Y. Shen, W. Hu, M. Yang, B. Wei, S. Lucey, C. T. Chou, Face recognition on smartphones via optimised sparse representation classification, in: IPSN-14 Proceedings of the 13th International Symposium on Information Processing in Sensor Networks, IEEE, 2014, pp. 237–248.

[63] S. He, K. G. Shin, Steering crowdsourced signal map construction via bayesian compressive sensing, in: IEEE INFOCOM 2018-IEEE Conference on Computer Communications, IEEE, 2018, pp. 1016–1024.

[64] 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 Transactions on Intelligent Systems and Technology (TIST) 9 (2) (2017) 1–28.

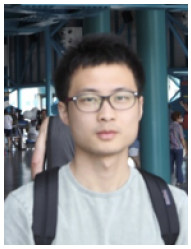
[65] E. Rangelova, W. Van der Wal, M. Sideris, P. Wu, Spatiotemporal analysis of the grace-derived mass variations in north america by means of multi-channel singular spectrum analysis, in: Gravity, geoid and earth observation, Springer, 2010, pp. 539–546.

[66] J. Du, H. Chen, W. Zhang, A deep learning method for data recovery in sensor networks using effective spatio-temporal correlation data, Sensor Review.

[67] Y. Liang, Z. Cui, Y. Tian, H. Chen, Y. Wang, A deep generative adversarial architecture for network-wide spatial-temporal traffic-state estimation, Transportation Research Record 2672 (45) (2018) 87–105.

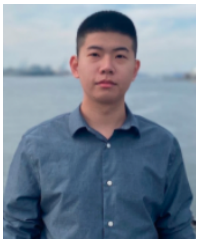
[68] X. Li, G. Cong, A. Sun, Y. Cheng, Learning travel time distributions with deep generative model, in: The World Wide Web Conference, 2019, pp. 1017–1027.

[69] Q. Yang, Y. Liu, Y. Cheng, Y. Kang, T. Chen, H. Yu, Federated learning, Synthesis Lectures on Artificial Intelligence and Machine Learning 13 (3) (2019) 1–207.
[70] F. Mo, H. Haddadi, Efficient and private federated learning using tee, in: Proc. EuroSys Conf., 2019.
[71] S. De, T. Goldstein, Efficient distributed sgd with variance reduction, in: 2016 IEEE 16th international conference on data mining (ICDM), IEEE, 2016, pp. 111–120.
[72] Y. Qian, C. Tan, D. Ding, H. Li, N. Mamoulis, Fast and secure distributed nonnegative matrix factorization, IEEE Transactions on Knowledge and Data Engineering.
[73] H. Qian, Y. Yu, Scaling simultaneous optimistic optimization for high-dimensional non-convex functions with low effective dimensions, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 30, 2016.
[74] Y.-m. Cheung, J. Lou, Efficient generalized conditional gradient with gradient sliding for composite optimization, in: Twenty-Fourth International Joint Conference on Artificial Intelligence, 2015.
[75] J. Chen, X. Pan, R. Monga, S. Bengio, R. Jozefowicz, Revisiting distributed synchronous sgd, arXiv preprint arXiv:1604.00981.
[76] C. Xie, S. Koyejo, I. Gupta, Zeno++: Robust fully asynchronous sgd, in: International Conference on Machine Learning, PMLR, 2020, pp. 10495–10503.
[77] H. Xiong, D. Zhang, Z. Guo, G. Chen, L. E. Barnes, Near-optimal incentive allocation for piggyback crowdsensing, IEEE Communications Magazine 55 (6) (2017) 120–125.
[78] V. Tolpegin, S. Truex, M. E. Gursoy, L. Liu, Data poisoning attacks against federated learning systems, in: European Symposium on Research in Computer Security, Springer, 2020, pp. 480–501.
[79] M. Nasr, R. Shokri, A. Houmansadr, Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning, in: 2019 IEEE symposium on security and privacy (SP), IEEE, 2019, pp. 739–753.



Jiang Bian (Member, IEEE) received the Ph.D. degree of Computer Engineering in University of Central Florida, Orlando, FL. He received the B.Eng degree of Logistics Systems Engineering in Huazhong University of Science and Technology, Wuhan, China, in 2014, and the M.Sc degree of Industrial Systems Engineering in University of Florida at Gainesville, FL, in 2016. He is currently with the Big Data Laboratory, Baidu Research, Beijing, China. His research interests include Ubiquitous Computing, and AutoDL.

Haoyi Xiong (Senior Member, IEEE) received the Ph.D. degree in computer science from Telecom SudParis jointly from Université Pierre et Marie Curie, Évry, France, in 2015. From 2016 to 2018, he was a Tenure-Track Assistant Professor with the Department of Computer Science, Missouri University of Science and Technology, Rolla, MO and a Postdoc at University of Virginia, Charlottesville, VA from 2015 to 2016. He is currently a Principal Research Scientist at Big Data Lab, Baidu Research, Beijing, China, and also a Graduate Faculty Scholar affiliated to University of Central Florida, Orlando FL. His current research interests include AutoDL and ubiquitous computing. He has published more than 70 papers in top computer science conferences and journals.



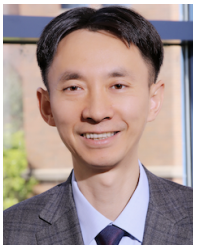
Zhiyuan Wang (Student Member, IEEE) is a Ph.D. student in system engineering at the University of Virginia, Charlottesville, VA. He received his bachelor's degree from the Department of Computer Science at Xiamen University, in 2021. His current research interests include ubiquitous computing, mobile sensing, and their applications on health and social issues including mental health and communicable diseases. This works was done in part, when Mr. Wang is a research intern at Baidu.

Jingbo Zhou (Member, IEEE) received the B.E. degree from Shandong University, Jinan, China, in 2009 and the Ph.D. degree from the National University of Singapore, Singapore, in 2014. He is currently a Staff Research Scientist with the Business Intelligent Lab of Baidu Research, working on machine learning problems for both scientific research and business applications, with a focus on knowledge graphs, natural language processing and spatio-temporal data mining. He was Program Committee Member among top conferences in the data mining and artificial intelligence communities like ACL, IJCAI, KDD, and AAAI.

Shilei Ji is a staff engineer at Baidu, Beijing, China, working on data security and privacy protection for both scientific research and business innovation, with a focus on secure multi-party computation, federated learning, and confidential computing. He obtained a master's degree from Tiangong University in 2009.

Hongyang Chen (Senior Member, IEEE) received the B.S. and M.S. degrees from Southwest Jiaotong University, Chengdu, China, in 2003 and 2006, respectively, and the Ph.D. degree from The University of Tokyo, Tokyo, Japan, in 2011. From 2011 to 2020, he was a Researcher with Fujitsu Ltd., Tokyo. He is an Adjunct Professor with Hangzhou Institute for Advanced Study, University of Chinese Academy of Sciences, China. He is currently a Senior Research Expert with Zhejiang Laboratory, China. He has authored or coauthored 100+ refereed journals and conference papers in top venues, and has been granted or filed more than 50 PCT patents. His research interests include the IoT, data-driven intelligent networking and systems, machine learning, localization, location-based big data, B5G, and statistical signal processing.

Daqing Zhang (Fellow, IEEE) is a Chair Professor with the Key Laboratory of High Confidence Software Technologies (Ministry of Education), School of Computer Science, Peking University, China, and Telecom SudParis, IP Paris, France. His research interests include context-aware computing, urban computing, mobile computing, big data analytics, pervasive elderly care, etc. He has authored more than 300 technical papers in leading conferences and journals. He received his Ph.D. degree from the University of Rome "La Sapienza", Italy, in 1996. He was the General or Program Chair for more than 17 international conferences, giving keynote talks at more than 20 international conferences. He is the Associate Editor for the IEEE Pervasive Computing, ACM Transactions on Intelligent Systems and Technology, and the Proceeding of ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies.



Dejing Dou (Senior Member, IEEE) is the Head of Big Data Lab (BDL) and Business Intelligence Lab (BIL) at Baidu Research. He is also a full Professor (on leave) from the Computer and Information Science Department at the University of Oregon and has led the Advanced Integration and Mining (AIM) Lab since 2005. He has been the Director of the NSF IUCRC Center for Big Learning (CBL) since 2018. He was a visiting associate Professor at Stanford Center for Biomedical Informatics Research during 2012-2013. Prof. Dou received his bachelor degree from Tsinghua University, China in 1996 and his Ph.D. degree from Yale University in 2004. His research areas include artificial intelligence, data mining, data integration, NLP, and health informatics. Dejing Dou has published more than 100 research papers.