IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received July 8, 2016, accepted August 2, 2016, date of publication September 7, 2016, date of current version November 8, 2016.

Digital Object Identifier 10.1109/ACCESS.2016.2606242

Temporal Dynamic Matrix Factorization for Missing Data Prediction in Large Scale Coevolving Time Series

WEIWEI SHI[1], YONGXIN ZHU[1], (Senior Member, IEEE), PHILIP S. YU[2], (Fellow, IEEE), TIAN HUANG[1], CHANG WANG[1], YISHU MAO[1], AND YUFENG CHEN[3]

[1]School of Microelectronics, Shanghai Jiao Tong University, Shanghai 200240, China
[2]Department of Computer Science, University of Illinois at Chicago, Chicago, IL 60607, USA
[3]Shandong Power Supply Company of State Grid, Jinan 250000, China

Corresponding author: Y. Zhu (zhuyongxin@sjtu.edu.cn)

This work was supported in part by the National High Technology and Research Development Program of China (863 Program) under Grant 2015AA050204, in part by the State Grid Science and Technology Project under Grant 520626140020, Grant 14H100000552, and Grant SGCQDK00PJJS1400020, in part by the State Grid Corporation of China, and in part by the National Natural Science Foundation of China under Grant 61373032.

ABSTRACT Data missing in collections of time series occurs frequently in practical applications and turns out to be a major menace to precise data analysis. However, most of the existing methods either might be infeasible or could be inefficient to predict the missing values in large-scale coevolving time series. Also, the evolving of time series needs to be handled properly to adapt to the temporal characteristic. Furthermore, more massive volume of data is generated in many areas than ever before. In this paper, we have taken up the challenge of missing data prediction in coevolving time series by employing temporal dynamic matrix factorization techniques. First, our approaches are optimally designed to largely utilize both the interior patterns of each time series and the information of time series across multiple sources to build an initial model. Based on the idea, we have imposed hybrid regularization terms to constrain the objective functions of matrix factorization. Then, temporal dynamic matrix factorization is proposed to effectively update the initial already trained models. In the process of dynamic matrix factorization, batch updating and fine-tuning strategies are also employed to build an effective and efficient model. Extensive experiments on real-world data sets and synthetic data set demonstrate that the proposed approaches can effectively improve the performance of missing data prediction. Even when the missing ratio reaches as high as 90%, our proposed methods still show low prediction errors. Dynamic performance demonstrates that the methods can obtain satisfactory effectiveness and efficiency. Furthermore, we have also demonstrated how to take advantage of the high processing power of Apache Spark to perform missing data prediction in large-scale coevolving time series.

INDEX TERMS Matrix factorization, missing data prediction, time series, Apache Spark.

I. INTRODUCTION

In the era of Big Data, quintillions of bytes of data are generated every day. Nearly 90 percent of the data in the world today were obtained within the past two years [1]. For example, Facebook has more than one billion active users. So, it is not surprising for the data-driven company to generate petabytes of data each day. On October 4, 2012, the presidential debate between President Barack Obama and Governor Mitt Romney triggered more than 10 million tweets within 2 hours [2]. Another example, the square kilometer array (SKA) built in South Africa and Australia can achieve 10000 times more sensitive vision than any existing radio telescopes, which aims at answering fundamental questions about the Universe. With an 1 terabytes/second data volume, the data generated from the SKA are exceptionally large [3]. Extensive number of research challenges have driven a great deal of interest in the field of Big Data, such as modeling, processing, querying, mining, and distributing large scale repositories [4].

In many applications, the Big Data of interest comprises multiple sequences that each evolves over time [5]. For example, in personal health care systems, sensors are deployed

to monitor individuals by detecting multiple temporal signals, such as heart rate, blood pressure, and sleep quality. In weather forecast systems, multiple coevolving atmospheric and land-soil variables, ranging from temperatures, winds, and precipitation to soil moisture and atmospheric ozone concentration, are collected in real time. In real-time environmental surveillance systems, multiple concentrations of suspended particulate matters and various pollution gases are observed to monitor the environmental health. In this paper, we refer to such temporal sequences as coevolving time series.

The ubiquity of coevolving time series in many practical applications has created great opportunities for mining the time series data. Nevertheless, before extracting great underlying value and critical information from large scale coevolving time series, the raw data quality should be paid much more attention at first. Unfortunately, due to the harsh working conditions or uncontrollable factors, such as the extreme weather, equipment failure or the unstable communication signal, the raw time series in a sensor network usually involve missing values. For instance, while in-service, missing values in power grid surveillance systems can occur for various reasons, such as quick evaporation of acetylene, the existence of contamination on the surface of the platinum alloy of a gas meter, and so on. Yet, in practice, sensors and communication failures are more common factors that produce missing values in many applications. And still worse, immediate fixation of these practical problems is rarely plausible and might cost too much.

The inevitability of generating missing values in coevolving time series and the consequent poor data quality necessitates effective data preprocessing in real time. Traditional methods, such as linear interpolation and support vector machine (SVM), could be used to predict the missing values of a time series. But they are only feasible to be applied to the case where only a low ratio of collected data are missing and the time series vary very steadily. Modeling methods, more commonly used solutions, attempt to discover the underlying patterns and seasonality to predict the missing values using some common sense [6]. Representative modeling approaches include deterministic models, stochastic models, and state space models [7]. For example, Frasconi et al. employed a seasonal kernel to measure the similarity between time-series instances and proposed the seasonal autoregressive integrated moving average model (SARIMA) coupled with a Kalman filter achieved excellent performance of missing data prediction [8]. Li et al. utilized the information of multiple points and proposed probabilistic principle component analysis (PPCA) based imputing method to impute the missed data in transportation systems [9]. Tang et al. proposed a hybrid approach integrating the Fuzzy C-Means (FCM)-based imputation method with the Genetic Algorithm (GA) by utilizing the weekly similarity among data to predict the missing values [10]. Besides, Song et al. used matrix factorization to predict traffic matrices and their method showed more effective performance than traditional methods [11]. Cai et al. proposed a dynamic contextual matrix factorization algorithm to predict the missing values in multi-source time series. Their method was built based on Temporal Bayesian networks (TBN) and obtained good prediction accuracy [12].

Nevertheless, these methods either focus on predicting the missing data in the time series from one single source or could not effectively handle the missing data prediction problems of large scale coevolving time series. For instance, The SARMIA method aims at predicting the missing values with underlying seasonality, and thus the method might have trouble modelling data that have no strict internal seasonality. Especially, when the amount of instances becomes too large, the seasonal kernel computation would cost too much time. Usually, the common matrix factorization method is required to incorporate the internal specific characteristic of the time series data, such as the spatial information, which might restrict the method to one specific application. As probabilistic graphical models, TBN performs not so well with small data sets. Besides, TBN are computationally expensive due to sigma functions and cross-corpora calculations, which might lose its effectiveness in the case of handling big data problems. Moreover, most of the existing methods are designed without exploiting any temporal information. Thus they cannot handle the missing data prediction problem efficiently when new data are available.

The current missing data prediction problems of large scale coevolving time series could be summarized as:

1) how to build effective models to predict the missing values in time series;
2) how to incorporate the temporal information into the proposed methods and update the already trained models when new time series data come;
3) how to deal with the large scale coevolving time series.

In this paper, we will take up the challenge of resolving the above problems. First, to build initial effective models for predicting missing values, we propose to fuse the temporal smoothness of time series and the information across multiple sources into matrix factorization. As each time series rarely fluctuate wildly over time, i.e., time series usually host internal and tangible pattern of temporal smoothness. Thus, we try to take advantage of the characteristic to reduce the prediction error of the missing data prediction in coevolving time series. Concretely, a selective penalty term is employed in the matrix factorization objective function to smooth the time series, i.e., we aim at minimizing the fluctuation of each time series with time. Besides, as coevolving time series are usually collected from multiple sources and there exits valuable correlation information across these sources, we also attempt to fuse that information into matrix factorization to obtain higher performance. Specifically, the correlation information is incorporated in designing one sensor network regularization term, i.e., the correlated sensors based regularization (CSR) term, to constrain the matrix factorization objective function. By combing the smoothness and CSR constraints together, we construct hybrid regularization terms

and build the initial prediction models. Second, we focus on updating the already trained models when new time series samples are available. We take two steps to address the problem. At the first step, we fix one of the latent factors and update the other latent factor related to the temporal dimension based on the new samples. At the second step, as we have updated the latent factors, we adjust the latent factors by fine-tuning based on the whole data set in order to further improve the accuracy of the models. Finally, to deal with large scale coevolving time series, we implement our proposed methods on Big Data platform. Apache Spark is a large scale distributed data processing environment that builds on the principles of scalability and fault tolerance that plays a key role in the success of Hadoop and MapReduce [13]. Here, we implement our proposed approaches on Apache Spark.

By taking advantage of temporal dynamic matrix factorization, four Models are built:

1) DMPO: temporal Dynamic Matrix factorization for missing data Prediction based on One sample updating;
2) DMPOf: temporal Dynamic Matrix factorization for missing data Prediction based on One sample updating with Fine tuning;
3) DMPB: temporal Dynamic Matrix factorization for missing data Prediction based on Batch updating;
4) DMPBf: temporal Dynamic Matrix factorization for missing data Prediction based on Batch updating with Fine tuning.

The experimental results reveal that our proposed methods show superior performance to the traditional and state-of-the-art algorithms.

The contributions of this paper are summarized as follows:

1) We propose novel and effective methods to constrain the matrix factorization by fusing the temporal smoothness of each time series and the information across multiple sources to improve the performance of missing data prediction in coevolving time series. Also, we systematically illustrate how to design matrix factorization objective functions with the carefully designed hybrid regularization terms.
2) We propose temporal dynamic matrix factorization to update the already trained models when new time series samples are available. We elaborate the selection of updating rules and the necessity of incorporating the fine-tuning.
3) We implement and verify the proposed methods with three data sets from real world and one synthetic data set. And for big data analysis, we also implement and verify the proposed methods on Apache Spark platform.

II. PROBLEM FORMULATION

In this paper, we focus on the missing data prediction problem of coevolving time series. To illustrate this problem more clearly, we show an example of coevolving time series in Fig. 1. The left side of the figure shows the initial state of the coevolving time series data set. The right side represents the new data samples to show that the time series are evolving with time.

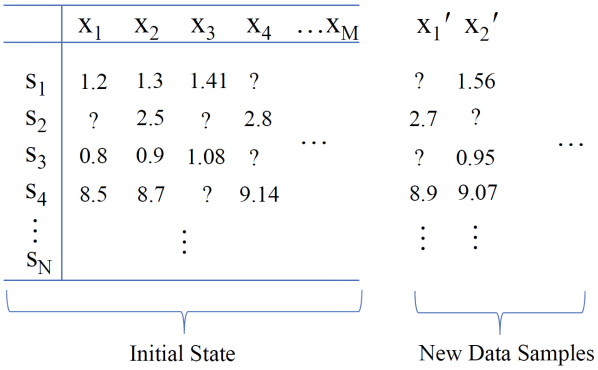


FIGURE 1. Coevolving time series.

TABLE 1. Symbols and Definition.

Symbols	Definition and Description
X	initial data set of coevolving time series
N	number of data sources
M	length of initial time series X
s_i	the time series from the ith data source
x_j	the jth sample in X
W	indicator matrix of X
S, V	latent factors of X
L	dimension of latent factors
x', X'	new data sample(s) of coevolving time series
W'	indicator matrix of X'
x'_j	the jth sample in X'
X^n	new data set of coevolving time series
W^n	indicator matrix of X^n
S^n, V^n	latent factors of X^n
H	similarity functions
B_{ij}	the element at the ith row and jth column of matrix B
B_i	the ith row of matrix B
B^T	transpose of matrix B
$\circ$	Hadamard product

Table 1 lists the main symbols we use throughout this paper. For the initial state, let $X = \{\boldsymbol{s}_1, \boldsymbol{s}_2, \boldsymbol{s}_3, \ldots, \boldsymbol{s}_N\}$ be the multi-source time series collected from N different data sources, and the jth entity in time series data from the ith source can be denoted as X_{ij} for $i = \{1, 2, 3, \ldots, N\}$, $j = \{1, 2, 3, \ldots, M\}$. x_j represents the jth sample of X. In our case, when X_{ij} in multi-source time series is missing, it will be denoted as '?'. In addition, we use a matrix $W \in \mathbb{R}^{N*M}$ to indicate whether the value in X is missing or observed. The entries of W can be defined as:

$$w_{ij} = \begin{cases} 0 & \textit{if } X_{ij} \textit{ is missing} \\ 1 & \textit{otherwise.} \end{cases} \tag{1}$$

for all $i = \{1, 2, 3, \ldots, N\}, j = \{1, 2, 3, \ldots, M\}$.

Similarly, we use the symbol X' to denote the new data set. And the jth sample in X' is denoted as x'_j. The matrix W' indicates the missing values in X'.

The problem of the missing data prediction in coevolving time series can be defined as follows:

Problem 1: Training initial models based on initial data set of coevolving time series.

Given: initial data set of coevolving time series $X \in \mathbb{R}^{N*M}$; an indicator matrix W;

Output: Initial models P_{ini} trained based on X.

Problem 2: Updating initial models for predicting the missing values in the whole data set.

Given: initial data set of coevolving time series X, initial models P_{ini}, new samples of coevolving time series X'; an indicator matrix W';

Output: Updated model P based on X and X'.

III. PROPOSED METHODS

In this section, we describe the details of the proposed methods for missing data prediction in coevolving time series. We begin with the discussion about the baseline solution for the problem. Next, we elaborate how to construct hybrid regularization terms and build an effective initial model. Then, we give the details about how to update the already trained models when new data samples are available. Finally, we give the detailed realization of the proposed methods based on Apache Spark.

A. FULL RANK MATRIX FACTORIZATION

The problem of matrix completion aims at estimating the missing entries of a matrix from a limited number of its entries. Let $X^\star$ denote the matrix whose entries $X^\star_{ij}$ are all observed. Fixed-rank matrix completion amounts to solving the following optimization problem:

$$\min_{X \in \mathbb{R}^{N*M}} \frac{1}{|\Omega|} \|Q_\Omega(X) - Q_\Omega(X^\star)\|_F^2 \quad subject\ to\ rank(X) = L, \tag{2}$$

where Ω is a subset of the complete set of indices $\{(i, j) : i \in \{1, 2, \ldots, N\}$ and $j \in \{1, 2, \ldots, M\}\}$, Q_Ω is called the orthogonal sampling operator, and $|\Omega|$ is the cardinality of the set. The predictions of unknown entities, i.e., missing values, with a low-rank prior would have the interpretation that the whole coevolving time series only depend on few entities [14].

One popular way to parameterize fixed-rank matrices is through matrix factorization. And the singular value decomposition (SVD) is a popular and effective factorization of a real or complex matrix. Given $X \in \mathbb{R}^{N*M}$, the singular value decomposition of X is given by

$$X = U \Sigma Q^T, \tag{3}$$

where $U \in \mathbb{R}^{N*L}$ is a matrix with orthogonal columns, $Q \in \mathbb{R}^{M*L}$, and $\Sigma \in \mathbb{R}^{L*L}$ is a diagonal matrix containing the singular values of X along its main diagonal.

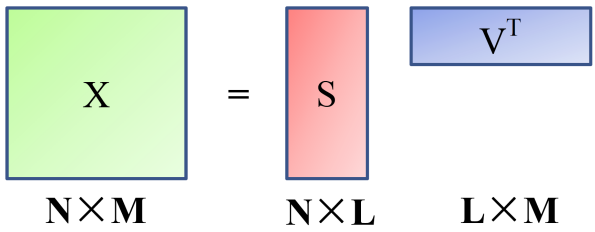


FIGURE 2. Fixed-rank matrix factorizations.

As Fig. 2 shows, the most popular fixed-rank factorization is obtained when the SVD is rearranged as

$$X = (U \Sigma^{\frac{1}{2}})(\Sigma^{\frac{1}{2}} Q^T) = SV^T, \tag{4}$$

where $S = (U \Sigma^{\frac{1}{2}}) \in \mathbb{R}^{M*L}$ and $V = (\Sigma^{\frac{1}{2}} Q^T) \in \mathbb{R}^{M*L}$. $\mathbb{R}^{N*L}$ and $\mathbb{R}^{N*L}$ are the set of full column rank matrix. So this kind of fixed-rank matrix factorization is also known as *full rank matrix factorization*.

A standard formulation of the problem is to determine S and V with respect to the existing components:

$$\min_{S,V} \frac{1}{2} \|X - SV^T\|_F^2. \tag{5}$$

As the original matrix X might contain a great number of missing values, we only need to factorize the observed entities in X. Hence, we have a modified optimization problem:

$$\min_{S,V} \frac{1}{2} \|W \circ (X - SV^T)\|_F^2 + \frac{\lambda_1}{2} \|S\|_F^2 + \frac{\lambda_2}{2} \|V\|_F^2, \tag{6}$$

where $\lambda_1, \lambda_2 > 0$ and $\circ$ denotes the Hadamard product. Two Frobenius regularization terms $\|S\|_F^2$ and $\|V\|_F^2$ are added in order to avoid overfitting [15]. Gradient based approaches can be applied to find a local minimum in Equation (6) due to their effectiveness and simplicity [16]. Equation (6) also contains a nice probabilistic interpretation with Gaussian observation noise, which is detailed in [17]. The product of S and V^T is the reconstructed X and is denoted as $\hat{X}$ in the paper.

In general, from the aforementioned formulation, we can provide a practical configuration of our methods:

$$\min_{S,V} \frac{1}{2} \|W \circ (X - SV^T)\|_F^2 + \frac{\lambda_1}{2} \|S\|_F^2 + \frac{\lambda_2}{2} \|V\|_F^2 + \alpha_S J_S(S) + \alpha_V J_V(V), \tag{7}$$

where α_S and α_V are nonnegative regularization parameters and the terms $J_S(S)$ and $J_V(V)$ are chosen to enforce the desired properties of the time series data. In the following subsections, we will show how to design effective regularization terms according to the properties of coevolving time series in the process of matrix factorization. Also, we will show how to update the latent factors S and V when new time series samples are available.

B. INITIAL MODELS BUILDING VIA HYBRID REGULARIZATION

1) TEMPORAL SMOOTHNESS REGULARIZATION

In the real world, a large amount of time series usually do not fluctuate wildly. For example, the room temperature, the gas concentrations in electric equipments, the energy consumption in cities, and product prices in the market rarely change drastically. In order to fuse the temporal smoothness of each time series, as V denotes the latent matrix with time dimension, optimization problem is improved as:

$$\min_{S,V} \mathcal{L}_V(X,S,V) = \frac{1}{2}\|W \circ (X - SV^T)\|_F^2 + \frac{\lambda_1}{2}\|S\|_F^2 + \frac{\lambda_2}{2}\|V\|_F^2 + \frac{\beta}{2}\|GV^T\|_F^2, \quad (8)$$

where typical examples of the matrix G are the 1st derivative approximation $G_1 \in \mathbb{R}^{(L-1)\times L}$ and the 2nd derivative approximation $G_2 \in \mathbb{R}^{(L-2)\times L}$ [18], given by

$$G_1 = \begin{pmatrix} 1 & -1 & & & 0 \\ & 1 & -1 & & 0 \\ \vdots & & \ddots & \ddots & 0 \\ 0 & & & 1 & -1 \end{pmatrix}, \quad (9)$$

$$G_2 = \begin{pmatrix} -1 & 2 & -1 & & 0 \\ & -1 & 2 & -1 & 0 \\ \vdots & & \ddots & \ddots & 0 \\ 0 & & & -1 & 2 & -1 \end{pmatrix}. \quad (10)$$

As Alternating Least Squares (ALS) can be done effectively, the key idea of which is to find the local optimum solution of S and V alternatively, where the gradients of $\mathcal{L}_V(X,S,V)$ with respect to S_i and V_i could be calculated as:

$$\frac{\partial \mathcal{L}_V}{\partial S_i} = \sum_{j=1}^{M} W_{ij}(S_i V_j^T - X_{ij})V_j + \lambda_1 S_i \quad (11)$$

for all i $\in \{1, 2, \ldots, N\}$,

$$\frac{\partial \mathcal{L}_V}{\partial V_j} = \sum_{i=1}^{N} W_{ij}(S_i V_j^T - X_{ij})S_i + \lambda_2 V_j + \beta V_j G^T G \quad (12)$$

for all j $\in \{1, 2, \ldots, M\}$.

2) CORRELATED SENSORS BASED REGULARIZATION

Coevolving time series are usually collected from multiple sources. In spite of the fact that the different sensors are assigned different tasks, they usually share one common goal and there might exist a strong correlation among some of the sensors. For example, in a smart building equipped with various sensors, the humidity might go up with the temperature. So the humidity sensor may have a strong correlation with the temperature sensor. In environmental monitoring systems, there also might be a high correlation between the chemical and biological sensors as their detection values may possibly change simultaneously. As for the personal medical care, the blood pressure usually increases with the heartbeat, thus the corresponding sensors may have a strong correlation. If one sensor has a strong correlation with another one, we call the two sensors are *correlated*.

As S denotes the latent sensor matrix and there might be strong correlation among correlated sensors, we improve the missing data prediction model based on matrix factorization technique, i.e., Correlated Sensors based Matrix factorization, with the following optimization problem:

$$\min_{S,V} \mathcal{L}_S(X,S,V) = \frac{1}{2}\|W \circ (X - SV^T)\|_F^2 + \frac{\lambda_1}{2}\|S\|_F^2 + \frac{\lambda_2}{2}\|V\|_F^2 + \frac{\alpha}{2}\sum_{i=1}^{N}\|S_i - \frac{1}{|C(i)|}\sum_{c\in C(i)}\rho_{i,c}S_c\|_F^2, \quad (13)$$

where α is the penalty factor and $\alpha > 0$, $C(i)$ denotes the set of the correlated sensors of the ith sensor and $|C(i)|$ is the total number of these correlated sensors. The included scaling factor $\rho_{i,c}$ aims at matching the scale difference between the ith sensor and the cth sensor. In this model, we incorporate one sensor network regularization term, i.e., the Correlated Sensors based Regularization (CSR) term

$$\frac{\alpha}{2}\sum_{i=1}^{N}\|S_i - \frac{1}{|C(i)|}\sum_{c\in C(i)}\rho_{i,c}S_c\|_F^2, \quad (14)$$

in order to minimize the distance between the ith sensor and its correlated sensors. Concretely, if the correlated sensors are $C(i)$, then we deduce that the state of the ith sensor is correlated to the average state of $C(i)$.

The above sensor network regularization imposes a hypothesis that the state between the ith sensor and the average state of $C(i)$ is very close, after scale adjustment. However, this hypothesis is usually invalid in the real world. For instance, there is one temperature sensor, one humidity sensor and one light sensor in a smart room. The temperature sensor might have a stronger correlation with the light sensor than the humidity sensor. Thus, a more practical model should treat the correlated sensors in $C(i)$ differently based on how correlated they are with the ith sensor. As a consequence, the optimization problem in Equation (13) is improved as:

$$\min_{S,V} \mathcal{L}_S(X,S,V) = \frac{1}{2}\|W \circ (X - SV^T)\|_F^2 + \frac{\lambda_1}{2}\|S\|_F^2 + \frac{\lambda_2}{2}\|V\|_F^2 + \frac{\alpha}{2}\sum_{i=1}^{N}\|S_i - \frac{\sum_{c\in C(i)} H(i,c) * \rho_{i,c}S_c}{\sum_{c\in C(i)} H(i,c)}\|_F^2. \quad (15)$$

The sensor network regularization item CSR in Equation (15) is designed to treat each sensor in $C(i)$ differently. The function $H(i,c)$ measures the similarity between the ith sensor and the cth sensor. From this improved regularization item, we know that if the cth sensor is very correlated to the ith sensor, the value of $H(i,c)$ will be large, i.e, it contributes more to the state of the ith sensor. Similarly, the gradients of

$\mathcal{L}_S(X, S, V)$ with respect to S_i and V_i could be calculated as:

$$\frac{\partial \mathcal{L}_S}{\partial S_i} = \sum_{j=1}^{M} W_{ij}(S_i V_j^T - X_{ij})V_j + \lambda_1 S_i + \alpha(S_i - \frac{\sum_{c\in C(i)} H(i,c) * \rho_{i,c} S_c}{\sum_{c\in C(i)} H(i,c)}), \quad (16)$$

$$\frac{\partial \mathcal{L}_S}{\partial V_j} = \sum_{i=1}^{N} W_{ij}(S_i V_j^T - X_{ij})S_i + \lambda_2 V_j, \quad (17)$$

for all i $\in \{1, 2, \ldots, N\}$ and j $\in \{1, 2, \ldots, M\}$.

The above proposed methods aim at taking advantage of either the information across multiple sources or the temporal smoothness of time series. Naturally, it is convincing that the combination of the two characteristics of the coevolving time series can also contribute to improving the performance of missing data prediction. So, we now propose the initial model P_{ini} with hybrid regularization, the objective function of which is:

$$\min_{S,V} \mathcal{L}_H(X, S, V) = \frac{1}{2}\|W \circ (X - SV^T)\|_F^2 + \frac{\lambda_1}{2}\|S\|_F^2 + \frac{\lambda_2}{2}\|V\|_F^2 + \frac{\alpha}{2}\sum_{i=1}^{N}\|S_i - \frac{1}{|C(i)|}\sum_{c\in C(i)} \rho_{i,c} S_c\|_F^2 + \frac{\beta}{2}\|GV^T\|_F^2, \quad (18)$$

The gradients of $\mathcal{L}_H(X, S, V)$ with respect to S_i and V_i could be calculated in the similar way shown in Equation (17) and Equation (17), so they are not included here.

The proposed regularization terms in Equation (18) require a function H to measure the similarity between two sensors, which is a key component of the proposed method. In this paper, we incorporate a popular similarity functions, i.e., Pearson Correlation Coefficient (PCC).

C. TEMPORAL DYNAMIC MATRIX FACTORIZATION

After building the initial models P_{ini} via hybrid regularization, we move on to handle the problem of updating the already trained models when new samples are available. Figure 3 shows the main updating process, which could be divided into two steps. At the first step, we utilize the new sample to obtain the new latent factor V'. At the next step, the complete model is updated based on the whole data set. We will elaborate these two steps in the following text.

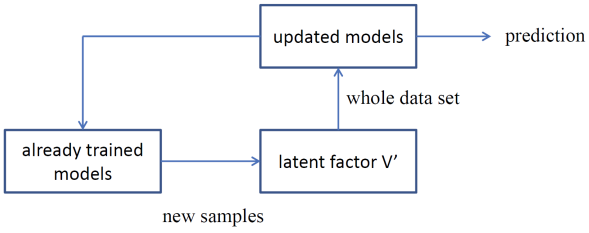


FIGURE 3. The process of updating the models.

When a new sample x' with a few values signs in, the new whole coevolving time series matrix becomes: $X^n = [X\ x']$. As Equation (4) shows, new latent factors S^n and V^n need to be recalculated now. The intuitive method is very computationally expensive as it computes the latent factors repeatedly as long as the new samples are available.

Given the fact that the coevolving time series rarely fluctuate wildly, we reasonably assume that the new sample x' does not have the ability to affect the distribution of the other samples in the latent factor space. That is to say, V^n is the collection of V' and original V, and we set S^n equal to S. So the problem becomes how to obtain the latent factor V' for now. According to the Equation

$$X^n = [X \quad x'] = S^n(V^n)^T = S^n[V \quad V']^T = S[V \quad V']^T, \quad (19)$$

we can get the following Equation:

$$V'^T = S^{-1}x'. \quad (20)$$

In fact, V'^T is one column vector, which needs to be calculated and collected into the original matrix V. As the dimension of the latent factor S of coevolving time series is less than the number of data sources, we deduce that the computation time will be limited in a reasonable range. We denote the above first model as DMPO (temporal Dynamic Matrix factorization for missing data Prediction based on One sample updating).

As the time series coevolve with time, the model is updated repeatedly when new samples are collected along with time. However, when more and more time series samples accumulate, the hypothesis that the latent factors of already trained model remain the same will lose its effectiveness. As a consequence, the performance of the updated model will deteriorate badly. Thus we further update the model by fine-tuning after obtaining V'^T, and build the second model, which is denoted as DMPOf (DMPO with fine-tuning) . As we have obtained $S^n = S$ and $V^n = [VV']$ based on DMPO, we can set these two latent factors as the initial parameter of DMPOf. The initialization will save a great deal of training time as S^n and V^n have been close enough to optimum parameters. In the fine-tuning process, the model is retrained based on the whole data set, i.e., X^n. The training process of DMPOf is similar to the procedure shown in Section III-B, so it is not included here.

Nevertheless, the already trained model does not have to be updated as long as one new sample x comes. Instead, it is a much more practical and efficient strategy to update the model with a batch of new samples. Let $X' = \{x_1, x_2, \ldots, x_m\}$ be the data set of new samples. The Equation (20) could be rewritten as:

$$V'^T = S^{-1}X'. \quad (21)$$

Now we can conclude the general updating steps. First, we remain the latent factor S of the initial model unchanged.

Then, according to Equation (21), based on the new data set X', we calculate the new latent factor V'^T. Next, we simply stack original V with V'^T. Finally, we refine the updated model by fine-tuning. We denote the third model as DMPB (temporal Dynamic Matrix factorization for missing data Prediction based on batch updating). Similarly, by combining DMPB with fine-tuning, we build the fourth model, which is denoted as DMPBf (DMPB with fine-tuning).

D. IMPLEMENTATION OF THE PROPOSED METHODS ON APACHE SPARK

The term Big Data has become the center of attention in various areas over the past few years [19]. The scale of modern coevolving time series data sets is also rapidly growing. And there is an imperative need to develop solutions to harness this wealth of data using statistical methods. Apache Spark is a powerful open source cluster computing framework built around speed, ease of use, and sophisticated analytics. It was originally developed at UC Berkeley in 2009. Since its release, Apache Spark has seen rapid adoption by enterprises across a wide range of industries, such as Netflix, Yahoo, and eBay. Spark's in-memory parallel execution model in which all data will be loaded into memory to avoid the I/O bottleneck benefits the iterative computation. Spark also provides very flexible DAG-based data flows, which can significantly speedup the computation of the iterative algorithms [20]. The two features of Spark bring performance up to 100 times faster compared to Hadoop's two-stage MapReduce paradigm.

Here, we implement our proposed methods on Apache Spark platform. In the process of building initial models, as the CSR regularization term is hardly implemented in matrix form, the solution of Equation (18) needs to be slightly modified. For example, given that the correlated sensors based regularization term only exerts an effect on the gradient of $\mathcal{L}'_{SV}(X, S, V)$ with respect to S_i, we divide the gradient computation into two steps. First, to make the solutions more adaptable to the platform, the gradients of the objective functions are rewritten in matrix form. For example, the gradients of $\mathcal{L}_V(X, S, V)$ with respect to S and V could be calculated as:

$$\frac{\partial \mathcal{L}_V}{\partial S} = W \circ (SV^T - X)V + \lambda_1 S, \tag{22}$$

$$\frac{\partial \mathcal{L}_V}{\partial V} = W \circ (SV^T - X)^T S + \lambda_1 V + \beta V G^T G. \tag{23}$$

Then, the gradient of $\mathcal{L}'_{SV}(X, S, V)$ could be simply summed by:

$$\frac{\partial \mathcal{L}'_{SV}}{\partial S_i} = [\frac{\partial \mathcal{L}_V}{\partial S}]_i - \alpha'(S_i - \frac{\sum\limits_{c' \in C'(i)} H(i, c') * \rho_{i,c'} S_{c'}}{\sum\limits_{c' \in C'(i)} H(i, c')}), \tag{24}$$

for all i $\in \{1, 2, \ldots, N\}$, where $[\cdot]_i$ represents the ith row of the matrix.

In the process of updating models, the new V'^T is obtained by calculating Equation (20) or (21) in the parelleled environment of Apache Spark. And then, we can get the collection of V^T and V'^T. Finally, the updated models P are obtained by retraining the model based on the whole data set X^n described in the last subsection.

E. OVERALL ALGORITHM

Putting everything together, we have the overall algorithm based on DMPOf for solving the missing data prediction problem in coevolving time series. As Algorithm 1 shows, given coevolving time series X, one new data sample x and the other parameters, the algorithm is designed to obtain a more accurate solution of the factors S^n and V^n. At the first step, the algorithm builds initial model by updating S and V until convergence, and the step size γ is updated in each iteration based on the line search strategy [21]. At the second step, V' is calculated according to (20). In the process of fine-tuning, new collected V^{nT} is used as the initial parameter when retraining the matrix factorization model. Finally, the missing values could be obtained from the predicted $\hat{X}^n$. The algorithms of DMPO, DMPB, DMFBf are similar with Algorithm 1, so they are not included here.

Algorithm 1 DMPOf for Missing Data Prediction in Coevolving Time Series

Input: initial data set of coevolving time series X,;
new data sample x;
indicator matrix W and W^n;
dimension of latent factors L;
parameters α, λ, $|C(i)|$;
Output: $\hat{X}^n$: the predicted values of X^n

function buildingModel(X,W)
  repeat
    $\gamma \leftarrow$ computing the best step size;
    for $i = 1$ to N **do**
      $S_i \leftarrow S_i$ - $\gamma \frac{\partial \mathcal{L}_V}{\partial S_i}$ ▷ based on Equation (11)
    end for
    for $j = 1$ to M **do**
      $V_j \leftarrow V_j$ - $\gamma \frac{\partial \mathcal{L}_V}{\partial V_j}$ ▷ based on Equation (12)
    end for
  until Convergence
end function
function updatingModel(x)
  $V'^T \leftarrow S^{-1}x'$ ▷ based on Equation (20)
  $V^{nT} \leftarrow [V^T \quad V'^T]$
  $P \leftarrow$ buildingModel(X^n,W^n) ▷ fine-tuning
end function
Predicted $\hat{X}^n \leftarrow S^n V^{nT}$

Next, we also show one representative algorithm, i.e. DMPBf, on Apache Spark. At the first step of building initial model, as Algorithm 2 shows, the input data X and W are first transformed to resilient distributed data set(RDD), i.e. *rddX* and *rddW*, respectively, which is a new distributed memory abstraction in Spark. Then, to implement the matrix

Algorithm 2 DMPBf Implemented on Apache Spark

Input: the path of initial data set *dataPath*; the path of new data samples *newdataPath*
Output: $\hat{X}^n$: the predicted values of X^n

function buildingModel(*dataPath*)
  repeat
    $rddX, rddW \leftarrow$ SparkContext.textFile(*dataPath*) ▷ X and W
    initialize the parameters; ▷ S, V, L, λ, β, and γ
    $rowMatrixX \leftarrow$ new RowMatrix($rddX$)
    calculate $\frac{\partial \mathcal{L}_V}{\partial S}$ and $\frac{\partial \mathcal{L}_V}{\partial V}$ ▷ based on Equation (23) and Equation (23) ▷ using RowMatrix.multiply
    $S \leftarrow S$.map{ S_i - $\gamma \frac{\partial \mathcal{L}'_{SV}}{\partial S_i}$ } ▷ updating S
    $V \leftarrow V$ - $\gamma \frac{\partial \mathcal{L}'_{SV}}{\partial V} \leftarrow V$ - $\gamma \frac{\partial \mathcal{L}_V}{\partial V}$ ▷ updating V
  until Convergence
end function
function updatingModel(*newdataPath*)
  $rddX'$, $rddW'$, $rddS^{-1} \leftarrow$ SparkContext.textFile(*newdataPath*)
  $rowMatrixS^{-1} \leftarrow$ new RowMatrix($rddS^{-1}$)
  $V'^T \leftarrow rowMatrixS^{-1}$. multiply($X'$) ▷ based on Equation (21)
  $V^{nT} \leftarrow [V^T \quad V'^T]$
  $P \leftarrow$ buildingModel(X^n,W^n) ▷ fine-tuning
end function
Predicted
$rowMatrixS^n \leftarrow$ new RowMatrix($rddS^n$)
$\hat{X}^n \leftarrow rowMatrixS$.multiply($V^{nT}$).collected()

multiplication in Spark, the $rddX$ is transformed as RowMatrix so that it could be multiplied by a local matrix, such as V^T. Likewise, the matrix product in Equation (23) and Equation (23) could be obtained. Next, S and V are updated by the calculated gradients until convergence. At the second step of updating model, V^{nT} is calculated according to (20), which shows much more importance in practical scenarios with large scale data sets as the comutation is based on a batch of new samples. Finally, the predicted $\hat{X}^n$ is obtained by performing RowMatrix multiplication one more time.

IV. EXPERIMENTS

A. DATA SET DESCRIPTION

The details about the four data sets are summarized in Table 3, which are all collected in pratical applications. The data sets consists of two medium scale data sets and two large scale data sets. We only focus on the discussion about the experimental results based on these four data sets, but the proposed methods can also be succefully applied to other data sets, such as [22] and [23].

1) MOTES DATA SET

The motes data set contains temperature time series collected from 54 sensors deployed in the Intel Berkeley Research lab in about one month [24]. Each time series is collected once every 31 seconds. In the experiment, the length of each time series is 14000.

2) SEA-SURFACE TEMPERATURE DATA SET

The Sea-Surface Temperature (SST) data set consists of hourly temperature measurements from Tropical Atmosphere Ocean Project [25]. In the experiment, the length of each time series is 18000.

3) GAS SENSOR ARRAY DATA SET

Gas Sensor Array under dynamic gas mixtures (GSA) data set, a large scale data set, was collected in a gas delivery platform at the ChemoSignals Lab in the BioCircuits Institute, University of California San Diego [26]. GSA contains the acquired time series from 16 chemical sensors exposed to Ethylene in air at varying concentration levels. Each measurement was constructed by the continuous acquisition of the 16-sensor array signals for a duration of about 12 hours without interruption. In the experiment, the length of each time series is 1.5E6.

4) SYNTHETIC DATA SET

Synthetic (SYN) data set, the other large scale data set, is generated by $Asin(\omega y) + cons + noise$, where $A > 0$ denotes the amplitude of sinusoidal function, ω is the angular frequency, $cons$ represents a non-zero center amplitude and $noise \sim N(0, 1)$ is an additive Gaussian noise. In the experiment, the parameters are set as $A \in \{2, 2.5, 3\}$, $con \in \{2, 3, 5\}$, $\omega \in \{1, \pi, 2\pi\}$ and the length of SYN is 1E6.

B. EXPERIMENTAL SETUP

As Table 3 shows the initial length of coevolving time series, which are the basic parameters of the proposed four models. As for the DMPB and DMPBf models, the batch sizes are also shown in the Table. For fair comparison with the baseline methods, the experiments are conducted with the same parameters when we evaluate the performance of the proposed method with different missing ratios.

To evaluate all the methods fairly, we incrementally simulate the data missing of the four data sets with an increasing missing ratio. For example, to increase the missing ratio from 0.80 to 0.90, we randomly move 10% of the total data from the observed data set to the missing data set. In this way, the subsequent missing data set always contains the missing data of the previous one. The missing data prediction of testing data sets is based on the known 10% of available values in the training set.

The parameters λ_1 and λ_2 are both set equal to 0.01 in this paper. For all of the four models, which incorporate temporal smoothness constraints, the 2nd derivative approximation matrix G_2 is employed in the regularization terms. To ensure the existence of the inverse of S in Equation (20) and (21), the latent dimension L is set as the number of sources N.

The algorithm stops when the change of the cost of two latest iterations is lower than a threshold value (1E-7).

TABLE 2. Statistics for the four data sets in the experiment, showing the total number of sensors, the total length of samples in the time series, and the number of training and test data set.

	Property				Experiment	
Data set	#sensors	#time	mean	standard variance	#Initial length	#Batch size
MO	54	14000	21.24	10.78	4000	100
SST	11	18000	19.1	8.97	8000	100
GSA	16	1.5E6	4.46	2.37	1E6	5000
SYN	27	1E6	3.4	3.2	5E5	5000

TABLE 3. Execution time of proposed methods and baseline methods.

Data set	LI	NMF	PMF	BPMF	DCMF	DMPO	DMPOf	DMPB	DMPBf
MO	0.37	16.99	3.33	75.62	1.42	0.048	1.19	0.046	1.3
SST	0.19	7.16	2.24	60.49	1.66	0.035	1.57	0.038	1.65
GSA	1.81	97.75	10.63	221.65	18.25	0.31	10.11	0.36	10.2
SYN	2.19	640.5	30.97	1466.1	94.3	0.35	20.13	1.37	36.32

The line search strategy involved in our methods selects the step size and the step direction simultaneously, which provides values that help to converge to the absolute minimum of the loss function.

Parallel computing experiments are carried out in a cluster of four working machines based on the same experimental setup given above. As there is no reasonably significance in implementing parallel experiments with small or medium data sets, these experiments are only conducted based on the two large scale data sets, i.e., GSA and SYN data sets. The working nodes are virtual machines (VM) and each of them has two cores with an Intel 2400 CPU and 4G memory. The operating system for the cluster is CentOS 7, while the version of Apache Spark platform is 1.4.0 and the Hadoop platform is version 2.6.0.

1) COMPARISON METHODS

The baseline methods are selectively chosen based on their popularity and effectiveness in building prediction models. The comparison methods include:

- **Linear Interpolation**: LI uses the mean value of two nearest neighbors of the missing entries to predict the missing value.
- **Non-negative Matrix Factorization**: NMF is originally proposed for image processing. However, it is commonly used in collaborative filtering recently, which is an alternative method to address the problem of missing data prediction [27].
- **Probabilistic Matrix Factorization**: PMF is another method to address the missing data prediction problem of multi-source time series [28].
- **Bayesian PMF**: BPMF is a fully Bayesian treatment of PMF, which is more appropriate for predicting the missing data with large missing ratios [17].
- **Support Vector Machine**: SVM approach builds regression models based on each source in the sensor network respectively [22].
- **Dynamic Contextual Matrix Factorization**: DCMF is proposed based on joint matrix factorization and linear dynamic systems, which utilizes the context information to predict the missing values of time series [12].

To the best of our knowledge, most of developed algorithms in temporal dynamic matrix factorization are compared with static versions of some baseline algorithms. As the static methods, which include LI, NMF, PMF, BPMF, SVM, cannot deal with dynamic time series, they retrain the models based on the whole data set when new samples are available.

2) EVALUATION METHOD

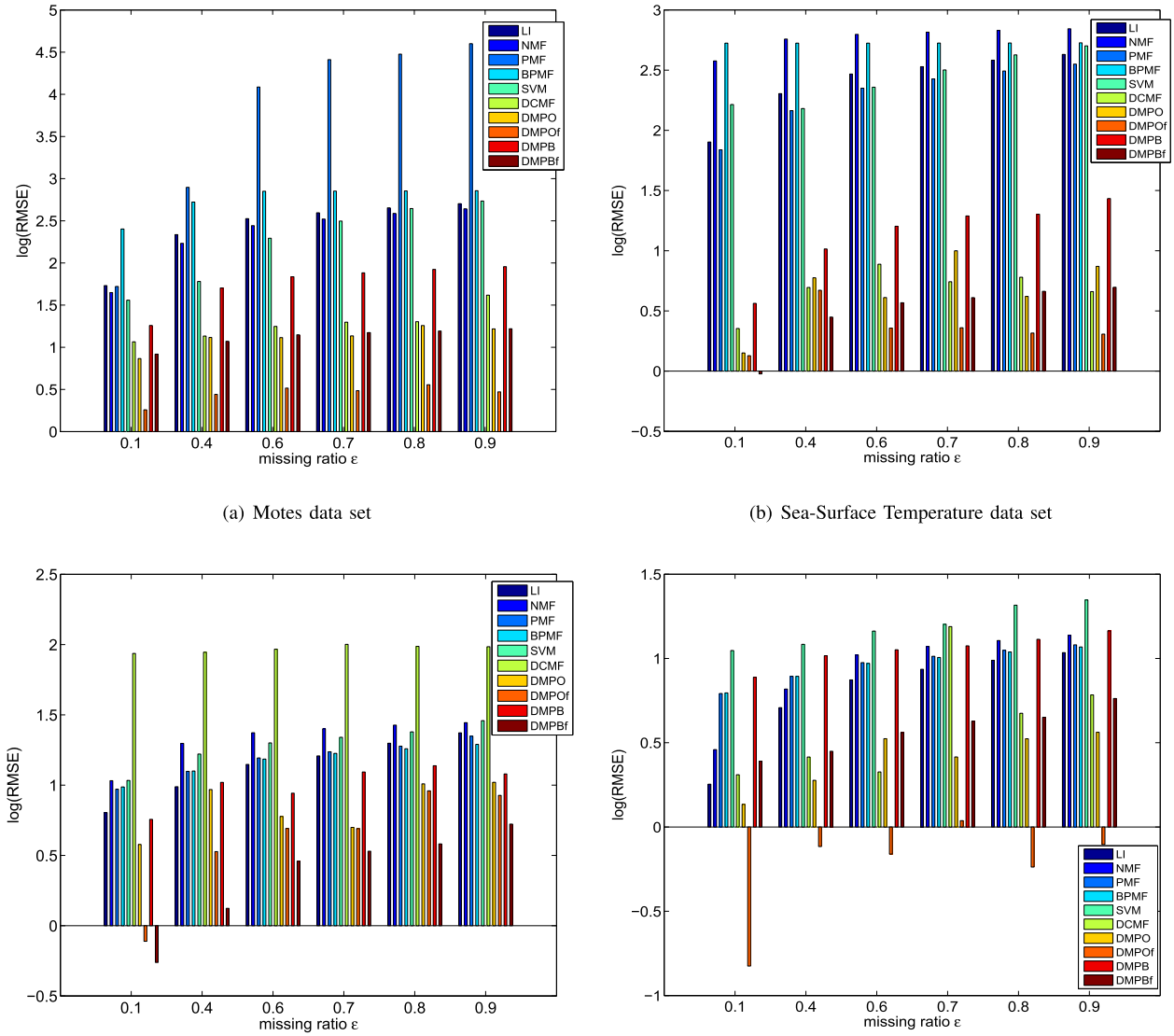
To evaluate the performance of the proposed method, root mean squared error (RMSE) is used to measure the prediction quality. RMSE is a frequently used measure of the differences between values predicted by a model or an estimator and the values actually observed. The RMSE represents the sample standard deviation of the differences between predicted values and observed values. In this paper, RMSE is defined as follows:

$$RMSE = \sqrt{\frac{\sum_{i,j}(1 - W_{ij}^n)(X_{ij}^n - \hat{X}_{ij}^n)^2}{\sum_{i,j}(1 - W_{ij}^n)}}, \tag{25}$$

where X_{ij}^n is the coevolving time series matrix and $\hat{X}_{ij}^n$ is the corresponding predicted value. W^n is the indicator matrix.

C. EXPERIMENTAL RESULTS

Fig. 4 shows the experimental results of the proposed methods and baseline methods on the MO, SST, GSA and SYN data sets. The RMSE of the proposed methods shown in this figure is averaged based on 100 new samples for DMPO and DMPOf models. The RMSE of DMPB and DMPBf is averaged based on 100 new batches of samples. The logarithm of RMSE is shown in vertical axis to present the experimental results more clearly.



(a) Motes data set

(b) Sea-Surface Temperature data set

(c) Gas Sensor Array under dynamic gas mixtures data set

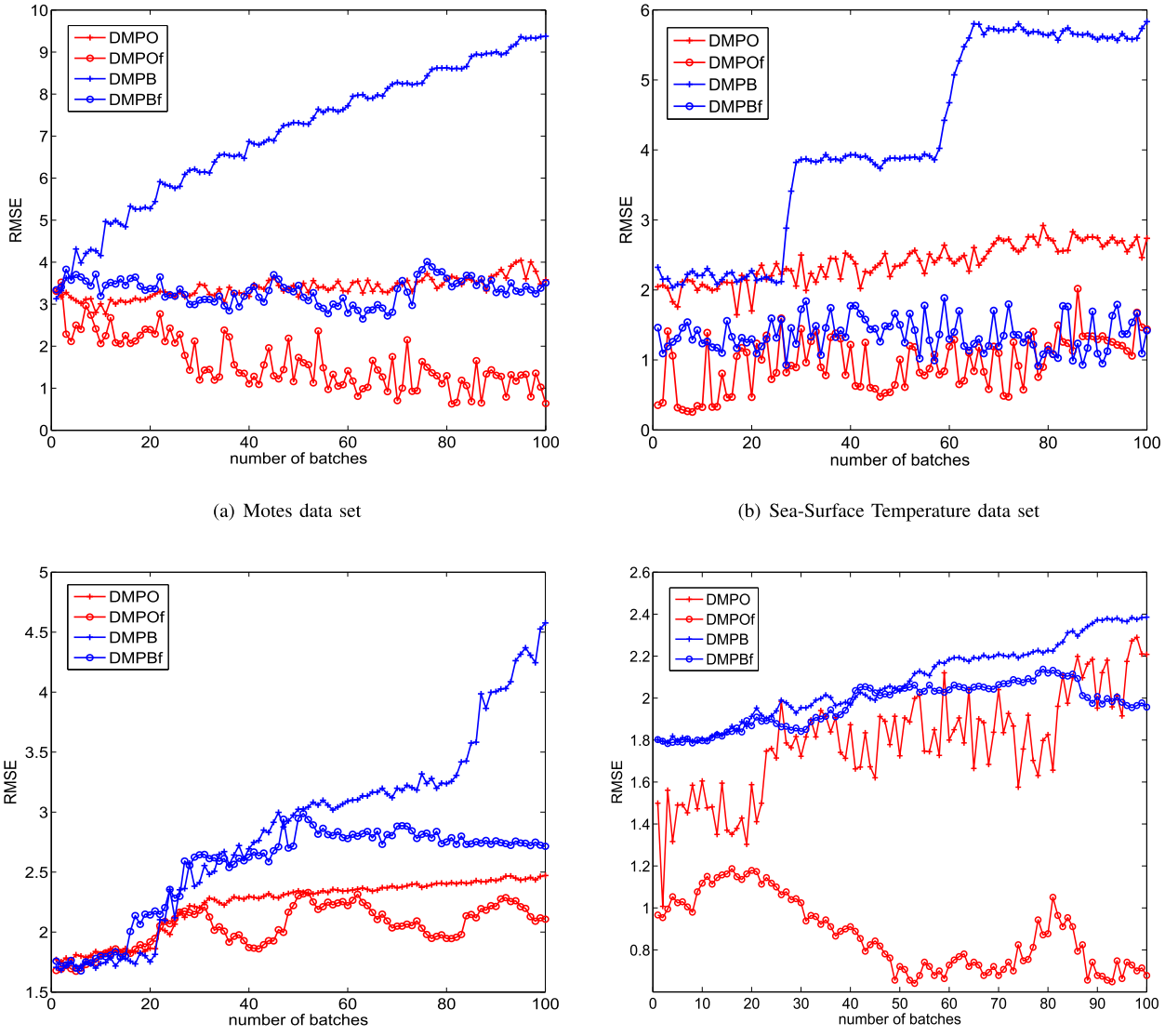
(d) Synthetic data set

FIGURE 4. Missing data prediction performance comparison between proposed methods and baseline methods. (a) Motes data set. (b) Sea-Surface Temperature data set. (c) Gas Sensor Array under dynamic gas mixtures data set. (d) Synthetic data set.

First, in general, the proposed four models show much better performance than the baseline methods, which demonstrates that the proposed matrix factorization methods based on fusing the temporal smoothness of time series and the information across multiple sources are suitable and effective to predict the missing values in multi-source time series. Concretely, let's consider the first model DMPO. We can see that DMPO consistently outperforms the other baseline methods. Quantitatively, as for the Motes data set, when the missing ratio ϵ is equal to 0.4, DMPO achieves the lowest RMSE 3.05, which is about 94% lower than PMF. Even when the missing ratio exceeds 0.6, the RMSE of MFS is still within a reasonable range. As for the other three data sets, the proposed method DMPO also outperforms other baseline methods, which show barely satisfactory results even when the missing ratio is as low as 0.1.

Moreover, with the fine-tuning strategy, the model DMPOf shows much smaller prediction error. Take the SST data set for example, the RMSE of DMPOf is roughly 9.7% lower than DMPO when the missing ratio is equal to 0.4. So, the fine-tuning strategy is an effective and necessary component of the proposed methods.

Next, the model DMPB, which utilizes the batch updating strategy, does not show a satisfactory result. The RMSE of DMPB is even higher than some of the baseline methods, e.g. DCMF. However, after adopting the fine-tuning strategy, the model DMPBf shows much better performance. Take the



(a) Motes data set

(b) Sea-Surface Temperature data set

(c) Gas Sensor Array under dynamic gas mixtures data set

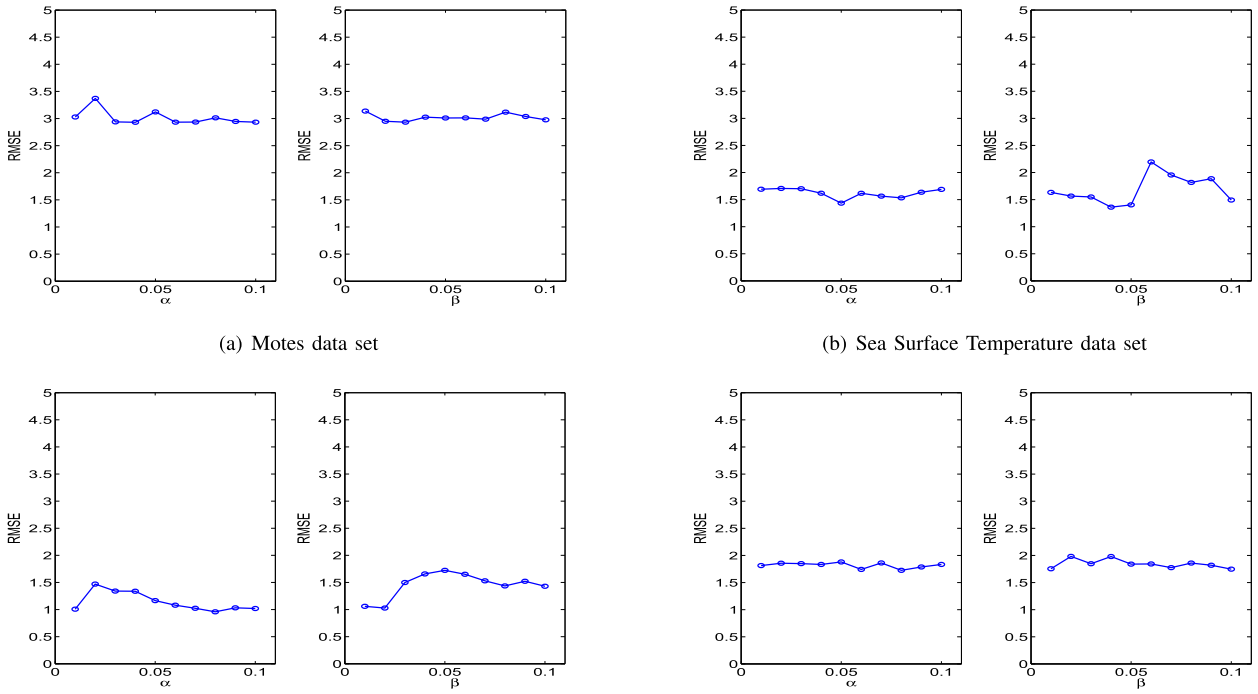
(d) Synthetic data set

FIGURE 5. RMSE with different number of samples or batches. (a) Motes data set. (b) Sea-Surface Temperature data set. (c) Gas Sensor Array under dynamic gas mixtures data set. (d) Synthetic data set.

GSA data set for example, the RMSE of DMPBf decreases about 59% compared with DMPB.

Fig. 5 shows the temporal dynamic performance, i.e., the trend of RMSE, with the number of samples or batches. On the one hand, we can observe that the models based on the batch updating rule show much larger prediction error than the models based on one sample updating. The experimental result is reasonable because the one sample updating rule takes more interations for each sample. On the other hand, for the DMPO model, we can observe that the prediction error is increased with the number of samples. Nevertheless, the RMSE of DMPOf remains stable and is below a resonable value. The much more obvious trend difference could be observed for DMPB and DMPBf. These results reveal that the fine-tuning strategy could reduce the residual error which might accumulate and increase with time during predicting the missing values in dynamic coevolving time series. In other words, the fine-tuning strategy could consistently limit the prediction error below a reasonable value. Besides, the figure also shows good stability and robustness of the proposed methods, as the RMSE shows little variation and is below an acceptable value at different time.

The models based on the batch updating rule, i.e., DMPB and DMPBf, aims at improving the computing efficiency, although they can not ensure consistently better performance. Table 3 gives the execution time of proposed methods and baseline methods. In general, DMPO takes the least time to train the model due to its simple computation. The time



(a) Motes data set

(b) Sea Surface Temperature data set

(c) Gas Sensor Array under dynamic gas mixtures data set

(d) Synthetic data set

FIGURE 6. Impact of Parameters. (a) Motes data set. (b) Sea Surface Temperature data set. (c) Gas Sensor Array under dynamic gas mixtures data set. (d) Synthetic data set.

that DMPOf model takes is increased a lot as it utilizes the fine-tuning strategy by retraining the model, but it is still less than most of the baseline method. The Table also shows that DMPB and DMPBf take more time than DMPO and DMPOf respectively. But given the fact that DMPO and DMPOf update the initial model as long as one new sample comes, the two models will demand much more time if we evaluate the execution time per 100 samples. So the batch updating strategy combined with fine-tuning can ensure both the effectivenss and efficiency of models at the same time.

D. PARAMETERS IMPACT DISCUSSION

In this subsection, we only give the analyses of the parameters of the fourth model DMPBf. Fig. 6 shows the impact of the parameters on the performance of DMPBf.

In general, the RMSE of DMPBf with different parameters is universally below a reasonable value and shows acceptable stability, despite the fact that there is a little bit variation with various parameter as shown in the figure.

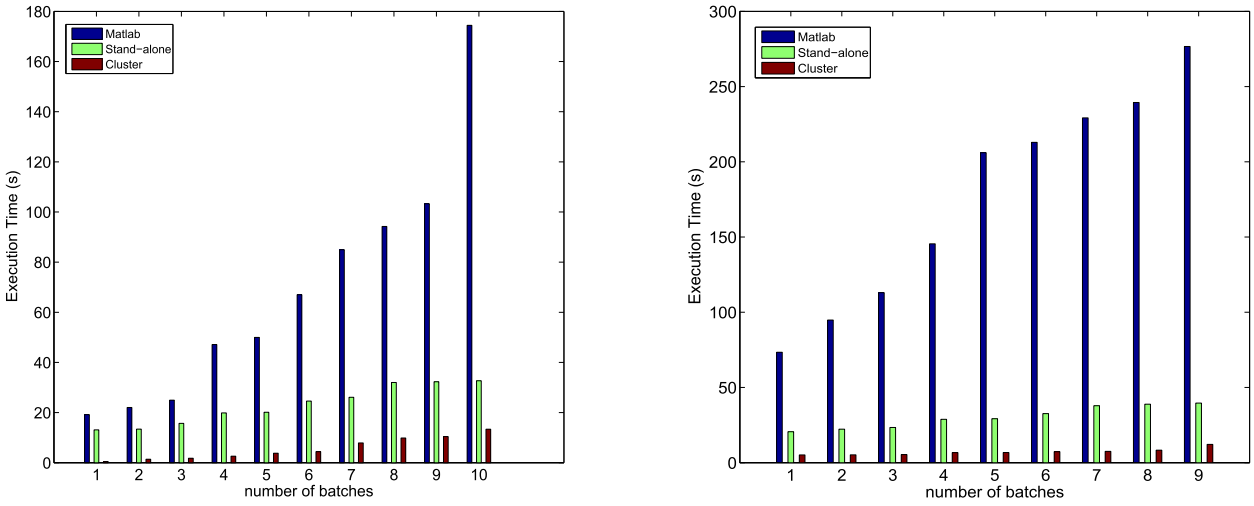
Concretely, the impact of α on the performance is presented. α controls how much information of the sensor network should be incorporated into the optimization problem. In general, as the Fig. 6 shows, the RMSE is not only consistently very low but also shows little variation for most of the different α values. We can observe that the best performance is achieved when α is equal to 0.04, 0.05, 0.08, and 0.06 for the four data sets respectively. We deduce that too small an α would greatly decrease the influence of the sensor regularization term on the matrix factorization. On the other hand, if we employ too large an α, the sensor regularization term would dominate the learning processes. So, an appropriate coefficient α could further improve the performance of the proposed method.

Then, the coefficient β is optimized. As the figure shows, it suffices to say that the model USMS generally achieves good stability with β, although Fig. 6(a) shows a slight fluctuation with the parameter β. Based on the experimental results, we can reasonably set $\beta = 0.03$, $\beta = 0.04$, $\beta = 0.02$, and $\beta = 0.10$ for the four data sets respectively.

E. EVALUATION UNDER PARALLEL ENVIRONMENT

Since the superior performance and efficiency of the proposed methods is obtained, which means that the proposed models can effectively predict the missing values in coevolving time series, we now turn to evaluate performance of the methods when it comes to dealing with the case of big data. As the DMPBf model generally shows the best performance among the proposed five models, we focus on the evaluation of DMPBf under parallel environment. Similar results are observed for the other proposed models. In order to show the scalability of DMPBf, we also perform experiments by a stand-alone computer on Spark platform, which has four cores from an Intel i7 and 8G memory. All the baseline methods are conducted under Matlab version 2012b by the same stand-alone computer.

We conduct the experiments on the execution time of DMPBf with different number of batches of GSA and SYN. Fig. 7 shows the execution time of DMPBf under



(a) Gas Sensor Array under dynamic gas mixtures data set (GSA)

(b) Synthetic data set (SYN)

FIGURE 7. Execution Time with different number of batches. (a) Gas Sensor Array under dynamic gas mixtures data set (GSA). (b) Synthetic data set (SYN).

different operating environment, which includes Matlab, Apache Spark with a stand-alone computer, Apache Spark cluster. We can observe that the DMPBf under parallel environment shows satisfactory scalability as it can deeply reduce the execution time. For example, when the number of batches of GSA data set is equal to 10, DMPBf only takes about 53 seconds, which is roughly 16 times faster than Matlab and 2.3 times faster than stand-alone computer. Thus we may conclude that the propose methods can be effective models for predicting the missing values in large scale coevolving time series.

V. CONCLUSION

In this paper, we have proposed novel methods to predict the missing data in the coevolving time series from multiple sources by temporal matrix factorization, which overcome some of the main defficiencies of the traditional methods such as the unfavorable computing efficiency and the requirement of seasonality and extra auxiliary information. First, the methods build effective initial models by fusing the smoothness characteristic of each time series and valuable correlation information across multiple sources into matrix factorization. Correspondingly, the methods incorporate hybrid regularization, i.e., smoothness and CSR constraints, to optimize the solution of matrix factorization. Second, the batch updating and fine-tuning strategy incorporated in the proposed methods can ensure the effectiveness and efficiency of predicting the missing values when new samples are available. Although the ALS can only ensure finding the local minimum of the target function, the proposed methods ensure computing efficiency and still show satisfactory results. Even when the missing ratio is as high as 90%, the RMSE of the proposed methods is still within reasonable range, while the performance of the state-of-the-art baseline methods has degraded greatly. Finally, the experiments under parallel environment reveal that the DMPBf model can be executed effectively. We conclude that the proposed methods would be strong alternative models for predicting the missing values in large scale coevolving time series in practical applications, such as healthcare monitoring, smart sensing and mobile computing systems. However, the proposed methods aim at handling the homogeneous data sets. In our future work, we might focus on dealing with the heterogeneous data sets.

REFERENCES

[1] X. Wu, X. Zhu, G.-Q. Wu, and W. Ding, "Data mining with big data," *IEEE Trans. Knowl. Data Eng.*, vol. 26, no. 1, pp. 97–107, Jan. 2014.

[2] A. Prem and P. Jayanthi, "Big data sources and data mining," *Int. Edu. Res. J.*, vol. 2, no. 4, 2016.

[3] *Square Kilometre Array*. (2016). [Online]. Available: http://thznetwork.net/index.php/thz-images

[4] M. M. U. Rathore, A. Paul, A. Ahmad, B. W. Chen, B. Huang, and W. Ji, "Real-time big data analytical architecture for remote sensing application," *IEEE J. Sel. Topics Appl. Earth Observat. Remote Sens.*, vol. 8, no. 10, pp. 4610–4621, Oct. 2015.

[5] Z. Si, H. Yu, and Z. Ma, "Learning deep features for dna methylation data analysis," *IEEE Access*, vol. 4, pp. 2732–2737, 2016.

[6] W. Lao, Y. Wang, C. Peng, C. Ye, and Y. Zhang, "Time series forecasting via weighted combination of trend and seasonality respectively with linearly declining increments and multiple sine functions," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, 2014, pp. 832–837.

[7] S.-F. Wu, C.-Y. Chang, and S.-J. Lee, "Time series forecasting with missing values," in *Proc. 1st Int. Conf. Ind. Netw. Intell. Syst. (INISCom)*, 2015, pp. 151–156.

[8] M. Lippi, M. Bertini, and P. Frasconi, "Short-term traffic flow forecasting: An experimental comparison of time-series analysis and supervised learning," *IEEE Trans. Intell. Transp. Syst.*, vol. 14, no. 2, pp. 871–882, Jun. 2013.

[9] L. Li, Y. Li, and Z. Li, "Efficient missing data imputing for traffic flow by considering temporal and spatial dependence," *Transp. Res. C, Emerg. Technol.*, vol. 34, pp. 108–120, Sep. 2013. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0968090X13001095

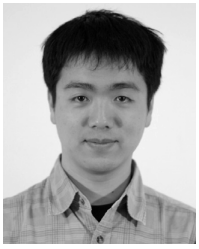
[10] J. Tang, G. Zhang, Y. Wang, H. Wang, and F. Liu, "A hybrid approach to integrate fuzzy C-means based imputation method with genetic algorithm for missing traffic volume data estimation," *Transp. Res. C, Emerg. Technol.*, vol. 51, pp. 29–40, Feb. 2015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0968090X14003192

[11] Y. Song, M. Liu, S. Tang, and X. Mao, "Time series matrix factorization prediction of Internet traffic matrices," in *Proc. IEEE 37th Conf. Local Comput. Netw. (LCN)*, Oct. 2012, pp. 284–287.
[12] Y. Cai, H. Tong, W. Fan, and P. Ji, "Fast mining of a network of coevolving time series," in *Proc. SIAM Int. Conf. Data Mining*, 2015, pp. 298–306. [Online]. Available: http://epubs.siam.org/doi/abs/10.1137/1.9781611974010.34
[13] M. Winlaw, M. B. Hynes, A. Caterini, and H. De Sterck. (2015). "Algorithmic acceleration of parallel ALS for collaborative filtering: Speeding up distributed big data recommendation in spark." [Online]. Available: http://arxiv.org/abs/1508.03110
[14] B. Mishra, G. Meyer, S. Bonnabel, and R. Sepulchre, "Fixed-rank matrix factorizations and Riemannian low-rank optimization," *Comput. Statist.*, vol. 29, no. 3, pp. 591–621, 2014.
[15] Y. Zhang, M. Chen, D. Huang, D. Wu, and Y. Li, "iDoctor: Personalized and professionalized medical recommendations based on hybrid matrix factorization," *Future Generat. Comput. Syst.*, to be published. (2016). [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167739X15003842
[16] H. Ma, D. Zhou, C. Liu, M. R. Lyu, and I. King, "Recommender systems with social regularization," in *Proc. 4th ACM Int. Conf. Web Search Data Mining (WSDM)*, 2011, pp. 287–296. [Online]. Available: http://doi.acm.org/10.1145/1935826.1935877
[17] R. Salakhutdinov and A. Mnih, "Bayesian probabilistic matrix factorization using Markov chain Monte Carlo," in *Proc. 25th Int. Conf. Mach. Learn. (ICML)*, 2008, pp. 880–887. [Online]. Available: http://doi.acm.org/10.1145/1390156.1390267
[18] A. Cichocki, R. Zdunek, A. H. Phan, and S.-I. Amari, *Nonnegative Matrix and Tensor Factorizations: Applications to Exploratory Multi-Way Data Analysis and Blind Source Separation*. New York, NY, USA: Wiley, 2009.
[19] X. Luo, M. Zhou, M. Shang, S. Li, and Y. Xia, "A novel approach to extracting non-negative latent factors from non-negative big sparse matrices," *IEEE Access*, vol. 4, pp. 2649–2655, 2016.
[20] W. Shi *et al.*, "An integrated data preprocessing framework based on Apache spark for fault diagnosis of power grid equipment," *J. Signal Process. Syst.*, vol. 82, pp. 1–16, Mar. 2016.
[21] E. E. Papalexakis, N. D. Sidiropoulos, and R. Bro, "From K-means to higher-way co-clustering: Multilinear decomposition with sparse latent factors," *IEEE Trans. Signal Process.*, vol. 61, no. 2, pp. 493–506, Jan. 2013.
[22] W. Shi *et al.*, "Improving power grid monitoring data quality: An efficient machine learning framework for missing data prediction," in *Proc. IEEE 17th Int. Conf. High Perform. Comput. Commun.*, Aug. 2015, pp. 417–422.
[23] S. De Vito, E. Massera, M. Piga, L. Martinotto, and G. D. Francia, "On field calibration of an electronic nose for benzene estimation in an urban pollution monitoring scenario," *Sens. Actuators B, Chem.*, vol. 129, no. 2, pp. 750–757, 2008. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0925400507007691
[24] M. Samuel. *Intel Lab Data*. http://db.csail.mit.edu
[25] Tropical Atmosphere Ocean. *NOAA/Pacific Marine Environmental Laboratory*. [Online]. Available: http://www.pmel.noaa.gov/tao/proj_over/proj_over.html
[26] J. Fonollosa, S. Sheik, R. Huerta, and S. Marco, "Reservoir computing compensates slow response of chemosensor arrays exposed to fast varying gas concentrations in continuous monitoring," *Sens. Actuators B, Chem.*, vol. 215, pp. 618–629, Aug. 2015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0925400515003524
[27] D. D. Lee and H. S. Seung, "Learning the parts of objects by non-negative matrix factorization," *Nature*, vol. 401, no. 6755, pp. 788–791, Oct. 1999.
[28] A. Mnih and R. R. Salakhutdinov, "Probabilistic matrix factorization," in *Proc. Adv. Neural Inf. Process. Syst.*, 2008, pp. 1257–1264.

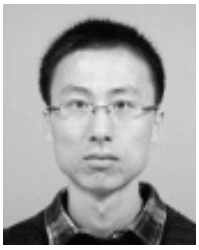
WEIWEI SHI received the bachelor's and M.Sc. degrees from the College of Opto-Electronic Engineering, Nanjing University of Posts and Telecommunications, China, in 2010 and 2013, respectively. He is currently pursuing the Ph.D. degree with the School of Electronic Information and Electrical Engineering, Shanghai Jiao Tong University. His current research interests are focused on data mining and big data.

YONGXIN ZHU (SM'–) received the B.Eng. degree in EE from the Hefei University of Technology, the M.Eng. degree in CS from Shanghai Jiao Tong University, and the Ph.D. degree in CS from the National University of Singapore. He is currently an Associate Professor with Shanghai Jiao Tong University, China. He is also a Visiting Associate Professor with National University of Singapore. He has authored over 90 English journal and conference papers and 30 Chinese journal papers. He has 18 China patents approved. His research interest is in computer architectures, embedded systems and system-on-chip. He is a senior member of the China Computer Federation.

PHILIP S. YU (F'–) received the B.S. degree in electrical engineering from National Taiwan University, the M.S. and Ph.D. degrees in electrical engineering from Stanford University, and the M.B.A. degree from New York University. He is a Distinguished Professor of Computer Science with the University of Illinois at Chicago. His research interest is on big data, including data mining, data stream, database, and privacy. He has authored over 830 papers in refereed journals and conferences. He holds or has applied for over 300 U.S. patents. He is a fellow of the ACM. He holds the Wexler Chair in information technology.



TIAN HUANG received the Ph.D. degree from the School of Microelectronics, Shanghai Jiao Tong University in 2016. His main research interest is data mining for time series, including time series big data indexing, anomaly detecting, computer architecture for time series data mining, and statistical models for time series data. He has authored two SCI journal and 15 EI conference papers. He also has rich experience on software and hardware co-designing.



CHANG WANG received the B.S. degree from the Computer Science and Technology School, Xi'an Polytechnic University in 2010, the M.S. degree from Computer Science School, Wu Han University in 2012. He is currently pursuing the Ph.D. degree with the School of Electronic Information and Electrical Engineering, Shanghai Jiao Tong University, Shanghai, China. His research interests are in the area of high performance computing focused on micro-architecture of future many-core processor and big data storage modeling.

YISHU MAO was born in China in 1993. She received the B.S. degree in electronic information science and technology from Sun Yat-sen University, China, in 2015. She is currently pursuing the M.S. degree with the School of Electronic Information and Electrical Engineering, Shanghai Jiao Tong University, China. In 2015, she was a Visiting Student at Brigham Young University, USA, working on cartilage magnetic resonance image segmentation algorithms. Her research interests include scheduling and high-performance computing.

YUFENG CHEN received the bachelor's degree from Shanghai Jiao Tong University in 1992. He is currently a Senior Engineer. He is also the Director of the Evaluation Center.