Prefetching Scheme for Massive Spatiotemporal Data in a Smart City

Lian Xiong, Zhengquan Xu, Hao Wang, Shan Jia, and Li Zhu

1 State Key Laboratory of Engineering in Surveying, Mapping and Remote Sensing, Wuhan University, Wuhan 430079, China
2 Collaborative Innovation Center for Geospatial Technology, Wuhan 430079, China
3 Hubei Collaborative Innovation Center for High-Efficiency Utilization of Solar Energy, Hubei University of Technology, Wuhan 430068, China

Correspondence should be addressed to Zhengquan Xu; xuzq@whu.edu.cn

Received 14 September 2015; Accepted 31 December 2015

Academic Editor: Liming Chen

Copyright © 2016 Lian Xiong et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Employing user access patterns to develop a prefetching scheme can effectively improve system I/O performance and reduce user access latency. For massive spatiotemporal data, traditional pattern mining methods fail to directly reflect the spatiotemporal correlation and transition rules of user access, resulting in poor prefetching performance. This paper proposed a prefetching scheme based on spatial-temporal attribute prediction, named STAP. It maps the history of user access requests to the spatiotemporal attribute domain by analyzing the characteristics of spatiotemporal data in a smart city. According to the spatial locality and time stationarity of user access, correlation analysis is performed and variation rules are identified for the history of user access requests. Further, the STAP scheme mines the user access patterns and constructs a predictive function to predict the user’s next access request. Experimental results show that the prefetching scheme is simple yet effective; it achieves a prediction accuracy of 84.3% for access requests and reduces the average data access response time by 44.71% compared with the nonprefetching scheme.

1. Introduction

The development of smart cities based on cloud computing and the Internet of Things has generated massive spatiotemporal data, including meteorological data, hydrological data, natural disaster data, and remote-sensing images, with three basic attributes, namely, location, time, and type. Such data are characterized by wide variety, large quantity, high redundancy, and dynamic growth over time. A smart city can quickly and conveniently provide users with rich predefined applications through a network platform based on the users’ demands for spatiotemporal data services such as data visualization, spatiotemporal correlation analysis, temporal emergency aid, and massive information retrieval.

Low latency, high concurrency, and high aggregate bandwidth are the three important criteria for measuring the quality of spatiotemporal data services in a smart city. Under the same bandwidth and computing power, the key factor affecting the quality of a spatiotemporal data service is the system delay in the network environment. Prefetching schemes have been widely used because they can effectively improve the data transfer rate and reduce the user access latency [1]. Therefore, it is important to develop an efficient prefetching scheme for improving the quality of spatiotemporal data services in a smart city.

Compared with nonspatiotemporal data, spatiotemporal data not only have three basic spatiotemporal attributes but also have obvious spatiotemporal correlation of user access; moreover, the corresponding prefetching schemes are different. Based on the different types of data, data prefetching schemes can be divided into two categories in the network environment.

(1) Nonspatiotemporal Data Prefetching. Nonspatiotemporal data prefetching mainly concerns web prefetching and personalized recommendations. In general, user access information is obtained by clustering or correlation analysis of webpages or users in order to mine user access patterns and
develop a prefetching scheme. Pallis et al. [2] employed the association rule to filter webpages visited by users in order to perform webpage clustering; then, they used the clustering result sets to develop a prefetching scheme for overcoming the problem of web access latency. Further, Wan et al. [3] used clustering to develop a method based on random indexing with various weight functions in order to track user access and cluster users with similar activity patterns. Khorasvi and Tarokh [4] adopted a naive Bayesian approach for dynamic mining of user access patterns in order to predict the pages accessed by users. Bamshad et al. [5] developed a personalized webpage recommendation system by using the a priori algorithm to identify pagesets frequently accessed by users; then, they matched the users' currently accessed pages with the frequently accessed pagesets. Matthews et al. [6] proposed a genetic algorithm based on association rules and discovered extra rules that are complementary to existing algorithms, which can facilitate the development of more effective prefetching schemes. Similar studies have been described in the literature [7–13].

(2) Spatiotemporal Prefetching. Spatiotemporal data prefetching mainly concerns WebGIS. The corresponding prefetching schemes use not only the characteristics of the data but also the spatiotemporal correlation of user access. Typically, the characteristics of the data are used to mine user access patterns and develop a prefetching scheme by spatiotemporal correlation analysis, transition probability calculation, access frequency ranking, and other methods related to user access requests or data. In order to overcome the problem of a long delay when users browse large objects in WebGIS, Park and Kim [14] used the spatial clustering characteristic of the Hilbert curve. They divided the entire geographic space using Hilbert curves and gave them appropriate values; then, they proposed a prefetching method based on the spatial locality of user access. Dong et al. [15] exploited the spatial locality of user access and proposed two prefetching methods. The first method computes transition probabilities between tiles and prefetches the most probable tile; the second method uses a “Neighbor Selection Markov Chain” to compute the objects to be prefetched based on the data of the k tiles previously requested. Considering the previous action of a given user, Yeşilmurat and İşler [16] proposed a heuristic prefetching algorithm that analyzes and ranks the previous moves of a user to predict the user's next move; then, it identifies the locations of candidate tiles to be prefetched. Considering both long-term and short-term popularity features for tile access in a geographic space, Li et al. [17] presented a Markov prefetching model in a cluster-based caching system based on the Zipf distribution and verified that the method has a high prefetch hit rate and a short average response time.

From existing studies, it can be seen that, in the network environment, a typical data prefetching scheme is based on current/historical user access information. It analyzes and processes the information at the level of access requests or data by using user access continuity, spatial locality, popularity, association rules between objects, and other methods in order to mine the user access patterns. Then, it predicts user access requests according to these patterns in order to achieve data prefetching.

However, we note that user access to spatiotemporal data usually has obvious spatiotemporal features in a smart city. The general approach mines user access patterns at the level of access requests or data, the results can only reflect this feature indirectly, and they are not useful for developing a high-efficiency prefetching scheme for massive spatiotemporal data. But if we analyze and process user access information at the level of spatiotemporal attributes, then the hidden spatiotemporal correlation and transition rules can be found, and we can develop a more targeted prefetching scheme. Therefore, how to effectively mine the spatiotemporal features and patterns from the user access information is the focus of this paper.

In this paper, we propose a prefetching scheme, STAP, for massive spatiotemporal data in a smart city. The proposed method analyzes the characteristics of spatiotemporal data and the spatiotemporal correlation of user access, parameterizes the history of user access requests, and extracts the spatiotemporal attributes. Then, it uses regional meshing, association rules, and the autoregressive integrated moving average (ARIMA) model in the spatiotemporal attribute domain to perform correlation analysis and identify transition rules, mines user access patterns, and constructs a predictive function to predict the user's next access request, in order to achieve spatiotemporal data prefetching.

The remainder of this paper is organized as follows. Section 2 introduces the motivation and principle of our prefetching scheme. Section 3 describes the implementation of our prefetching scheme, which mainly involves two steps. The first step shows how to (i) mine the user access patterns from the history of user access requests and (ii) construct the predictive function for predicting requests. The second part explains how to (i) predict the user's next access request according to the current one by using the abovementioned predictive function and (ii) prefetch the corresponding data. Section 4 presents and discusses the performance evaluation results of our prefetching scheme. Finally, Section 5 briefly summarizes our findings and concludes the paper.

2. Principle of Prefetching Scheme

2.1. Motivation. Typical prefetching schemes involve two steps. The first step is to mine the user access patterns and construct the predictive function; the second step is to predict access requests and prefetch the data. The first step is usually based on historical user access information as well as the characteristics of the data. It uses clustering, association rules, Markov models, and other methods to mine associate items accessed by users. Then, it merges them to form associate itemsets or uses mathematical functions to describe the correlation between the associate items. Thus, the corresponding request predictive function is constructed. The second step is based on the current user access request, the predictive function is used to predict the next access request of the user, and then the corresponding data is loaded into the cache.
For instance, suppose that \( h_i, 1 \leq i \leq n \), is a user access request; then, the history of user access requests can be expressed as the sequence \( H = (h_1, h_2, h_3, \ldots, h_n) \). The first step is to mine the access pattern, that is, to mine associate items \( h_i \rightarrow h_j \) from \( H \) and construct the request predictive function according to the associate items. The second step involves access request prediction and data prefetching. Take the current user access request \((h_i, h_j)\) as the input for the prediction function. Then, scan the associate itemsets to find the matching associate items \((h_i, h_j) \rightarrow h_k\); the output of the function \( h_k \) is the predicted access request. Finally, prefetch the corresponding data of request \( h_k \) to the cache.

It can be seen that the key aspect of the prefetching scheme is to mine user access patterns and construct the request predictive function. However, unlike ordinary user access features in the network environment, the users obtain spatiotemporal data services based on predefined applications in a smart city, which have obvious spatiotemporal correlation. For example, if a user checks the weather conditions by predefined applications, the access data are current time and location related meteorological data; when searching for the nearby living facilities, the access data, such as restaurants and parking lots, are closely related to the user’s current location. Therefore, if we treat the access request as a whole for direct mining as in the case of traditional mining patterns, the spatiotemporal correlation of user access will not be reflected directly, and the corresponding prediction function will not predict user access requests accurately.

In order to overcome the inherent drawback of mining user access patterns directly at the level of access requests, we start with the characteristics of spatiotemporal data and the spatiotemporal correlation of user access. Then, we mine the user access patterns at the level of spatiotemporal attributes and construct the access request predictive function.

### 2.2. Principle

Suppose that the history of user access requests in a smart city can be expressed as the sequence \( A = \langle a_1, a_2, a_3, \ldots, a_n \rangle \), where each \( a_i, 1 \leq i \leq n \), contains the following information: location attribute \( p \), type attribute \( s \), time attribute \( t \), user IP, and session time. In order to analyze and process the access sequences at the level of the spatiotemporal domain, we parameterize the information and extract the spatiotemporal attributes to form spatiotemporal attribute sequences:

\[
A = \langle (p_1, s_1, t_1), (p_2, s_2, t_2), \ldots, (p_n, s_n, t_n) \rangle
\]

\[
= \{P_n, S_n, T_n\}, \quad \tag{1}
\]

where \( a_i = (p_i, s_i, t_i) \) represents a parameterized request with the extraction results of the spatiotemporal attributes. Specifically, \( P_n = \{p_1, p_2, p_3, \ldots, p_n\} \) represents the sequence of location attributes, \( S_n = \{s_1, s_2, s_3, \ldots, s_n\} \) represents the sequence of type attributes, and \( T_n = \{t_1, t_2, t_3, \ldots, t_n\} \) represents the sequence of time attributes.

Because the spatiotemporal attribute sequences \( \{P_n, S_n, T_n\} \) contain three types of spatiotemporal attributes, it is extremely difficult to find the hidden spatiotemporal correlations and variation rules. However, we observe that, in a smart city, when most users request access to spatiotemporal data, the spatiotemporal attributes of the request have strong self-correlation but weak cross-correlation. That is to say, any two consecutive access requests, \( a_i, a_{i+1} \), have weak correlation between the location attribute \( p_i \) and the type attribute \( s_{i+1} \) but very strong correlation between \( p_i \) and \( p_{i+1} \). For example, when a user checks the current temperature of regional A, there is a huge possibility that he will further query the wind speed, PM2.5 of region A, rather than the water quality of other regions.

Therefore, to simplify access pattern mining, we process spatiotemporal attribute sequences \( \{P_n, S_n, T_n\} \) based on the self-correlation and cross-correlation of the spatiotemporal attributes of the access requests, in order to construct the access request predictive function. The specific steps are as follows:

1. For access requests with self-correlation of spatiotemporal attributes, we analyze the self-correlation of the spatiotemporal attribute sequences to mine associate items \( p_i \rightarrow p_j, s_i \rightarrow s_j, \) and \( t_i \rightarrow t_j \). Then, we construct the independent attribute prediction function \( \text{Pre}(p, s, t) = \{\text{Pre}(p), \text{Pre}(s), \text{Pre}(t)\} \), where \( \text{Pre}(p) \) represents the location attribute predictive function, \( \text{Pre}(s) \) represents the type attribute predictive function, and \( \text{Pre}(t) \) represents the time attribute predictive function.

2. For access requests with cross-correlation of spatiotemporal attributes, we carry out cross-correlation analysis of the spatiotemporal attribute sequences, and we mine associate items \( (p_1, s_1, t_1) \rightarrow (p_j, s_j, t_j) \). Then, we construct the conjoint attribute prediction function \( \text{Pre}''(p, s, t) \).

### 3. Implementation

The method is implemented in two steps. The first step is the offline mining of user access patterns to construct the predictive function, and the second step is the online access request prediction and data prefetching.

#### 3.1. Construction of Predictive Function

The predictive function consists of the independent attribute prediction function \( \text{Pre}'(p, s, t) = \{\text{Pre}(p), \text{Pre}(s), \text{Pre}(t)\} \) and the conjoint attribute prediction function \( \text{Pre}''(p, s, t) \).

#### 3.1.1. Construction of Independent Attribute Predictive Function

The key aspect of the location attribute predictive function \( \text{Pre}(p) \) is the correlation of access requests in the spatial domain. Therefore, we can use the association rule algorithm [18] to mine associate items \( p_i \rightarrow p_j \) from the sequence of location attributes \( P_n = \langle p_1, p_2, p_3, \ldots, p_n \rangle \) and construct...
the location attribute predictive function according to the associate rulesets.

(a) Regional Meshing. The location attribute of spatiotemporal data represents the geographical location of a data source in a smart city, usually expressed by latitude and longitude coordinates \( p = (x, y) \). However, solving the association rules of the location attribute coordinate points directly requires numerous calculations. Moreover, updating, modification, addition, or deletion operations on location attributes will require recalculation for the entire area. Therefore, we exploit regional meshing for the entire area, which allows for both the early solution of rules and late update of associate rules in the cell area, thereby providing partial and incremental solution of association rules and reducing the computation considerably.

Suppose that the geographic area is a two-dimensional Euclidean rectangular space \([0, X] [0, Y]\) in a smart city. We divide it into \( \times \text{row} \times \text{col} \) rectangular cells with coding, where the code of the area covered by the \( i \)th row \( j \)th column is \( g_{ij} = j + \text{col} \times (i - 1) \). Then, for any location attribute coordinate point \( p_k = (x_k, y_k) \) in the geographic area, we assume that it belongs to the cell \( g_{ij} \) if it satisfies the following equation:

\[
\begin{align*}
(i - 1) \frac{X}{\text{row}} \leq x_k & \leq i \frac{X}{\text{row}}, & 1 \leq i \leq \text{row} \\
(j - 1) \frac{Y}{\text{col}} \leq y_k & \leq j \frac{Y}{\text{col}}, & 1 \leq j \leq \text{col}.
\end{align*}
\]  

(2)

Figure 1(a) shows the geographic rectangular area divided into \( 4 \times 5 \) cells and the meshing cell coding. Figure 1(b) shows all the neighbor cells of the cell \( g_{ij} \).

(b) Construction of Predictive Function. Through regional meshing, we can use an association rule algorithm to mine the associate items of each cell from the sequence of location attributes \( P_n = \{p_1, p_2, p_3, \ldots, p_n\} \) and construct the location attribute predictive function \( \text{Pre}(p) \) according to the associate rulesets. The specific steps are as follows:

(1) Calculate the location coordinate sets \( p_{g_{ij}} = \{p_{g_1}, p_{g_2}, \ldots, p_{g_m}\} \) contained in the cell \( g_{ij} \) and its neighbor cells, as shown in Figure 1(b).

(2) Count the number of times every coordinate point \( p_{g_1}, p_{g_2}, \ldots, p_{g_m} \in p_{g_{ij}} \) appears in the sequence of location attributes, that is, \( \text{support} \), and compare it with the predefined \( \text{support threshold} \delta \), to find frequent 1-items. By looping through the location attribute sequence via the connection and cut between the frequent itemsets, we can find frequent 2-items, frequent 3-items, and so on until frequent \( m \)-items, \( 2 \leq m \leq n \), itemsets.

(3) Calculate the confidence of each frequent \( m \)-items and its subset frequent \( m-1 \)-items. Generate association rules \( \{p_{g_1}, p_{g_2}, \ldots, p_{g_m}\} \rightarrow \{p_{g_{ij}}, \phi_{g_{ij}}\} \) on those associate items whose confidence values are greater than the \( \text{confidence threshold} \phi \). Then, form the association rulesets

\[
R(g_{ij}, \phi_{ij}) = \bigcup_{m} \{(p_{g_1}, p_{g_2}, \ldots, p_{g_{m-1}}) \rightarrow \{p_{g_{ij}}, \phi_{g_{ij}}\}\}
\]

(4) Loop through each cell in the geographical area to calculate the location attribute association rulesets and merge them to form the associate rulesets \( R(p_{ij}, \phi_{ij}) = \bigcup_{ij} R(g_{ij}, \phi_{ij}) \) of the entire geographic area. Then, construct the location attribute predictive function,

\[
\text{Pre}(p) = \text{Match} \left( p, R \left( p_{ij}, \phi_{ij} \right) \right),
\]

where \( \text{Match}(\cdot) \) is a rule matching function, whose output is an associate rule matching successfully with location attribute \( p \); the specific methods are as in Section 3.2.1.

(2) Construction of Type Attribute Predictive Function. The key aspect of the type attribute predictive function \( \text{Pre}(s) \) is the correlation of access requests in the type domain. Therefore, we can use the association rule algorithm to mine associate items \( s_1 \rightarrow s_i \) from the sequence of type attributes \( S_n = \{s_1, s_2, s_3, \ldots, s_n\} \) and construct the type attribute predictive function according to the associate rulesets. The specific steps are as follows:

(1) Count the number of times every \( s_i, s_j \in S \), appears in the sequence of type attributes, that is, \( \text{support} \), and compare it with the predefined \( \text{support threshold} \delta \), to find frequent 1-items. By looping through the type attribute sequences via the connection and cut between the frequent itemsets, we can find frequent 2-items, frequent 3-items, and so on until frequent \( m \)-items, \( 2 \leq m \leq n \), itemsets.

(2) Calculate the confidence of each frequent \( m \)-items and its subset frequent \( m-1 \)-items. Generate association rules \( \{s_1, s_2, \ldots, s_{m-1}\} \rightarrow \{s_m, \phi_{s_{m}}\} \) on those associate items whose confidence values are greater than the \( \text{confidence threshold} \phi \), and form the association rulesets \( R(s_{ij}, \phi_{ij}) = \bigcup_{m} \{(s_1, s_2, \ldots, s_{m-1}) \rightarrow \{s_m, \phi_{s_{m}}\}\}\). Then, construct the type attribute predictive function

\[
\text{Pre}(s) = \text{Match} \left( s, R \left( s_{ij}, \phi_{ij} \right) \right).
\]

(3) Construction of Time Attribute Predictive Function. The key aspect of the time attribute predictive function \( \text{Pre}(t) \) is the correlation of access requests in the time domain. Therefore, we can analyze the sequence of time attributes \( T_n = \{t_1, t_2, t_3, \ldots, t_n\} \), to develop a model to describe this underlying correlation.

The time attribute sequence of user access requests is a typical nonstationary sequence influenced by a predefined application, which has obvious trends in a local range. ARIMA is an important and widely used short-term time series prediction model. It can predict future values according to the current and historical values of the sequence, but it requires the sequence to be stationary [19–23]. To this end, we can piecewise represent the time attribute sequence and
perform difference processing to achieve local stationarity. Then, we build the ARIMA model and construct the time attribute predictive function.

**(a) Piecwise Representation of Time Attribute Sequence.**
We use extreme point detection based on the slope change and piecewise representation of the time attribute sequence according to local extreme values in the sequence (the beginning and end value of each curve). The method calculates the slope difference \(|t_i - t_{i-1}| - |t_{i+1} - t_i|/\Delta T\) of the line segment formed by sequence value \(t_i\), 1 < \(i < n\), and its neighbor points \(t_{i-1}, t_{i+1}\), where \(\Delta T\) is the time interval of the access request. Then, we compare the slope difference with a predefined threshold. If it is greater than or equal to the predefined threshold, we assume that \(t_i\) is a local extremum. Finally, by using local extremal, we can piecewise represent the time attribute sequence as follows:

\[
T = \{(t_1, t_{1R}), (t_{2L}, t_{2R}), \ldots, (t_k, t_{kR})\},
\]

where \(t_{1L}\) is the starting value of \(i, i \in k\), segment, \(t_{kR}\) is the end value of \(i, i \in k\), piecewise, and \(k\) is the number of piecewise segments.

**(b) Construction of Predictive Function.**
With the above-mentioned piecewise representation and difference processing, we can realize local stationarity of the time attribute sequence and build ARIMA to construct the time attribute predictive function \(\text{Pre}(t)\) by introducing the \(k\)-step lag operator \(B^k t_{hn} = t_{hn-k}\) and \(d\)-order difference \(w_n = \Delta^d t_{hn} = (1 - B)^d t_{hn}\), \(d = 0, 1, 2\), the standard ARIMA\((p, d, q)\) model can be expressed as follows:

\[
w_n = \varphi_1 w_{n-1} + \varphi_2 w_{n-2} + \cdots + \varphi_p w_{n-p} + \delta + u_t
+ \theta_1 u_{t-1} + \theta_2 u_{t-2} + \cdots + \theta_q u_{t-q},
\]

where \(w_n = \Delta^d t_n = (1 - B)^d t_n\) is the difference order, \(\varphi_1, \varphi_2, \ldots, \varphi_p\) are the autoregressive parameters, \(\theta_1, \theta_2, \ldots, \theta_q\) are the moving average parameters, \(\delta\) is a constant that indicates that the sequence is nonzero mean, and \(u_t\) is white noise sequence.

Suppose that the right-and-left local extreme value of \(j, 1 < j < k\), piecewise is \(t_{1l} = t_m, t_{1r} = t_n\). Then, according to formula (5), we can piecewise represent it as \((t_{1l}, t_{1r}) = t_m, t_{m+1}, \ldots, t_n\), and, through \(d\)-order difference processing, it can be stationary. Judging from the fact that the user access is restricted by the predefined application, the change trend of the time attribute sequence can be only linear and regular, which means that it remains unchanged in terms of cycle and step size, so the parameters are \(p = 1, \varphi_1 = 1\). At the same time, the time attribute sequence of access request is not affected by external random interference, so the parameters are \(u_t = 0, q = 0\).

Finally, we can build ARIMA\((1, d, 0)\) as \(w_n = w_{n-1}\).

Combined with the lag operator \(B^k t_{hn} = t_{hn-k}\) and \(d\)-order difference, the time attribute predictive function \(\text{Pre}(t)\) can be expressed as follows:

\[
\text{Pre}(t_n) = \begin{cases}
t_{n-1}, & d = 0 \\
2t_{n-1} - t_{n-2}, & d = 1 \\
3t_{n-1} - 3t_{n-2} + t_{n-3}, & d = 2.
\end{cases}
\]

3.1.2. Construction of Conjoint Attribute Predictive Function.
Because the access requests with cross-correlation of spatiotemporal attributes account for a very small proportion of the total number of requests, it is difficult to jointly analyze the attributes. Therefore, we analyze only the access requests that have special cross-correlation of spatiotemporal attributes, that is, if the sequence of location attributes \(P_1 = (p_{11}, p_{12}, \ldots, p_{1l})\) and the sequence of type attributes \(S_1 = (s_{11}, s_{12}, \ldots, s_{1l})\) remain unchanged and the length reaches the minimum threshold \(l = 3\), as expressed by the following equation:

\[
P_{l+1} = P_{l+2} = \cdots = P_{l+l},
\]

\[
s_{l+1} = s_{l+2} = \cdots = s_{l+l},
\]

\(l \geq 3\).

Then, we assume that the location attribute and the type attribute remain unchanged in the next access request, and
the time attribute can be predicted by \(\text{Pre}(t)\). Finally, the joint attribute predictive function \(\text{Pre}''(p, s, t)\) can be constructed as

\[
\text{Pre}''(p, s, t) = \begin{cases} 
(p_{n-1}, s_{n-1}, t_{n-1}), & d = 0 \\
(p_{n-1}, s_{n-1}, 2t_{n-1} - t_{n-2}), & d = 1 \\
(p_{n-1}, s_{n-1}, 3t_{n-1} - 3t_{n-2} + t_{n-3}), & d = 2.
\end{cases}
\] (9)

3.2. Prediction Method for Access Requests. The purpose of this section is to show how to use the predictive function to predict the user’s next access request based on the current one.

Suppose that the current user access request can be expressed as the sequence \(B = \langle b_1, b_2, b_3, \ldots, b_m \rangle\), and spatiotemporal attributes sequences are \(B = \langle (p_1, s_1, t_1), (p_2, s_2, t_2), \ldots, (p_m, s_m, t_m) \rangle = \langle \hat{B} = \langle s_m, t_m \rangle \rangle\), where each \(b_i = (p_i, s_i, t_i), b_i \in B\), represents one user access request. First, we parameterize it and extract the spatiotemporal attribute sequence as the input, and the output \(\hat{b}_{m+1} = (\hat{p}_{m+1}, \hat{s}_{m+1}, \hat{t}_{m+1})\) is predicted as the access request.

We define a sliding and adaptive observation window with initial size \(w\), and we take the spatiotemporal attribute sequence \(\{P_W, S_W, T_W\}\), which falls into the observation window, as the input of the predictive function. Then, we judge whether \(\{P_W, S_W, T_W\}\) can satisfy formula (8). If it satisfies (8), we use the conjoint predictive function; otherwise, we use the independent attribute predictive function. As a result, \(\hat{p}_{m+1}, \hat{s}_{m+1}\), and \(\hat{t}_{m+1}\) can be predicted. Here, \(P_W = (p_{m-w+1}, p_{m-w+2}, \ldots, p_m)\), \(S_W = (s_{m-w+1}, s_{m-w+2}, \ldots, s_m)\), and \(T_W = (t_{m-w+1}, t_{m-w+2}, \ldots, t_m)\). (10)

3.2.1. Independent Attribute Predictive Function. For the access requests that satisfy formula (8), we predict the spatiotemporal attribute according to independent attribute prediction \(\text{Pre}'(p, s, t) = \{\text{Pre}(p), \text{Pre}(s), \text{Pre}(t)\}\), and then we form the predictive access request \(\hat{b}_{m+1} = (\hat{p}_{m+1}, \hat{s}_{m+1}, \hat{t}_{m+1})\).

(1) Location Attribute Prediction. Because regional meshing is used for the entire geographic area, before prediction, we need to judge whether the coordinate points belong to the same cell or neighbor cells according to formula (2). If the points belong to the same cell, we trigger the prediction; otherwise, we forgo prediction. The pseudocode for prediction of the location attribute is shown in Pseudocode 1.

Here, \(w'\) is the minimum observation window; \(R'(s_{ij}, \phi_{ij})\) is a temporary ruleset to store matched attribute items and the confidence, and \(P_{W-1} = \langle p_{m-w+2}, p_{m-w+3}, \ldots, p_m \rangle\) is a location attribute sequence with an observation window of one-bit duration. \(\text{Match}(P_W, R(g_{ij}, \phi_{ij}))\) is a rule matching function, whose output is an associate rule item matching successfully with \(P_W\), and the output is NULL when the match fails. For example, if the coordinate points of \(P_W\) belong to the same cell \(g_{ij}\) or neighbor cell, we use the rule matching function \(\text{Match}(P_W, R(g_{ij}, \phi_{ij}))\) to scan associate rulesets \(R(g_{ij}, \phi_{ij})\) for three matched associate items:

\[
\begin{align*}
(\hat{p}_{m+1}, \phi_1) \\
(\hat{p}_{m+1}, \phi_2) \\
(\hat{p}_{m+1}, \phi_3),
\end{align*}
\] (11)

where the confidence satisfies \(\phi_1 + \phi_2 + \phi_3 = 1\). If \(\phi_1 = \max(\phi_1, \phi_2, \phi_3)\), the location attribute of the predicted access request is \(\hat{p}_{m+1} = \hat{p}'_{m+1}\).

(2) Type Attribute Prediction. The type attribute prediction is similar to the location attribute prediction. Scan associate rulesets \(R(s_{ij}, \phi_{ij})\) according to the predictive function \(\text{Pre}(s)\). Find the associate rules matched with the type attribute sequence in the current observation window. Then, select the confidence rules with the largest confidence as the output results. Suppose that \(s_1, s_2, \ldots, s_m \rightarrow (\phi_{m+1}, \phi_1)\) is the associate rule matched successfully with \(S_W\), and \(\phi_1\) is the largest; then, the type attribute of the predicted access request is \(\hat{s}_{m+1} = s_{m+1}'\).

(3) Time Attribute Prediction. The time attribute prediction is based on the predictive function \(\text{Pre}(t)\). First, we perform \(d\)-order difference processing for \(T_W\) to achieve stationarity. Then, we use ARIMA to calculate the time attribute of the predicted access request; the result is given by

\[
\hat{t}_{m+1} = \hat{t}'_{m+1} = \begin{cases} 
2m - t_{m-1}, & d = 1 \\
3t_m - 3t_{m-1} + t_{m-2}, & d = 2.
\end{cases}
\] (12)

3.2.2. Conjoint Attribute Predictive Function. For the access requests that satisfy formula (9), we predict the next access request \(\hat{b}_{m+1}\) according to the conjoint attribute prediction function \(\text{Pre}''(p, s, t)\) and then form the predictive access request:

\[
\hat{b}_{m+1} = \hat{\hat{b}}_{m+1} = (\hat{\hat{p}}_{m+1}, \hat{\hat{s}}_{m+1}, \hat{\hat{t}}_{m+1}) = \begin{cases} 
(p_{m}, s_{m}, t_{m}), & d = 0 \\
(p_{m}, s_{m}, 2t_m - t_{m-1}), & d = 1 \\
(p_{m}, s_{m}, 3t_m - 3t_{m-1} + t_{m-2}), & d = 2.
\end{cases}
\] (13)

3.3. Data Prefetching. Data prefetching is performed to load data into the cache in accordance with the predicted request. To avoid unnecessary consumption of memory and computing resources, we built two data structure queues of length \(\lambda\). One is used to store the actual access requests and the other to
store the predicted requests. By calculating the consistent rate of the two queues, we can judge the degree of credibility of the current predicted requests. If the consistent rate achieves the predefined threshold, we assume that the predicted request is credible, and accordingly we carry out the data prefetching.

Suppose that the actual access request stored in a queue of length $\lambda$ is $\{b_{m-\lambda+1}, b_{m-\lambda+2}, \ldots, b_m\}$ and the predicted request is $\{\hat{b}_{m-\lambda+1}, \hat{b}_{m-\lambda+2}, \ldots, \hat{b}_m\}$. If they satisfy formula (14), we assume that the predicted request $\hat{b}_{m+1} = (\hat{p}_{m+1}, \hat{s}_{m+1}, \hat{t}_{m+1})$ is credible, and we load the corresponding data $d_{m+1}$ into the cache:

$$
\begin{align*}
\hat{p}_{m+1} &= p_{m+1}; \hat{p}_{m-\lambda+1} = p_{m-\lambda+1}; \hat{p}_{m-\lambda+2} = p_{m-\lambda+2}; \ldots; \hat{p}_m = p_m \\
\hat{s}_{m+1} &= s_{m+1}; \hat{s}_{m-\lambda+1} = s_{m-\lambda+1}; \hat{s}_{m-\lambda+2} = s_{m-\lambda+2}; \ldots; \hat{s}_m = s_m \\
\hat{t}_{m+1} &= t_{m+1}; \hat{t}_{m-\lambda+1} = t_{m-\lambda+1}; \hat{t}_{m-\lambda+2} = t_{m-\lambda+2}; \ldots; \hat{t}_m = t_{nr}
\end{align*}
$$

(14)

### 4. Experiment

This section consists of three parts. The first part introduces the performance evaluation metrics for our prefetching scheme. The second part describes the experimental data and methods. The last part presents and discusses the results of the experiments.

#### 4.1. Evaluation Metrics

We propose five criteria for performance evaluation of the proposed prefetching scheme in terms of accuracy, efficiency, and effectiveness.

**Prediction Accuracy.** It is the correct number of predicted requests as a percentage of the total number of requests.

**Prediction Coverage.** It is the number of predicted requests as a percentage of the total number of requests.

**Pattern Mining Time.** It is the time consumed for mining user access patterns from the history of user access requests.

**Request Prediction Time.** It is the average time for predicting an access request.

**Average Response Time.** It is the average response time to obtain a single data item.

#### 4.2. Experimental Data and Methods

The experimental data was obtained from Wuhan smart city network application demonstration platform, which includes 14 types of sensors in different regions; it has been generating sensor data since January 1, 2010, and provides 20 types of predefined applications to the public. We obtained the historical user access information from the user access log in the server for the period from September 1, 2014, to February 16, 2015. After processing, we generated 1,819,008 data access requests. The initial 1,628,183 requests formed the training set for mining user access patterns and constructing the predictive function. The remaining 190,825 requests formed the test set for testing the performance of the prefetching scheme.

The performance of the prefetching scheme is determined by the initial size of the observation window, regional meshing level, support threshold, and confidence threshold. Considering that the pattern mining is performed offline, the objective of the prediction is to choose the association rules with the maximum confidence. In order to choose the maximum number of rules and improve the trigger probability of prediction, we set the support threshold at 0.05% of the total number of access requests; that is, $\delta_p = \delta_s = 0.05\%$, and the confidence threshold was 0.01; that is, $\phi_p = \phi_s = 0.01$. Experiments were conducted to determine the changes in the evaluation metrics with different initial sizes of the observation window ($w = 2, 3, 4, 5,$ and 6) and different regional...
Table 1: Location attribute prediction.

| Regional meshing | Window size | 2   | 3   | 4   | 5   | 6   |
|------------------|-------------|-----|-----|-----|-----|-----|
| 1×1              |             | 86.95/93.92 | 89.11/93.92 | 89.19/93.92 | 89.23/93.92 | 89.27/93.92 |
| 50×50            |             | 81.04/91.11 | 85.93/91.11 | 85.96/91.11 | 86.01/91.11 | 86.08/91.11 |
| 80×80            |             | 80.26/90.32 | 85.11/90.32 | 85.17/90.32 | 85.21/90.32 | 85.26/90.32 |
| 100×100          |             | 79.70/89.79 | 84.58/89.79 | 84.63/89.79 | 84.72/89.79 | 84.75/89.79 |
| 120×120          |             | 79.22/89.27 | 84.07/89.27 | 84.13/89.27 | 84.18/89.27 | 84.24/89.27 |
| 150×150          |             | 78.43/88.49 | 83.27/88.49 | 83.33/88.49 | 83.39/88.49 | 83.43/88.49 |

Table 2: Type attribute prediction.

| Window size     | 2   | 3   | 4   | 5   | 6   |
|-----------------|-----|-----|-----|-----|-----|
| Accuracy (%)    | 94.38 | 95.28 | 95.52 | 95.80 | 96.05 |
| Coverage (%)    | 96.76 | 96.76 | 96.76 | 96.76 | 96.76 |

meshing levels \((\text{row} \times \text{col}) = 1 \times 1, 50 \times 50, 80 \times 80, 100 \times 100, 120 \times 120, \text{and} 150 \times 150\), and the average response time for users to access a single data item was recorded. Finally, we compare our prefetching algorithm (STAP, spatial-temporal attributes prediction) with associate rules discovery prefetching algorithm (ARP) proposed by [5] and neighbor selection Markov Chain prefetching algorithm (MCP) proposed by [15].

4.3. Experimental Results

4.3.1. Prediction Accuracy and Coverage

(a) Location Attribute Prediction. The prediction accuracy and coverage of the location attribute with different regional meshing levels and observation window sizes are summarized in Table 1. As can be seen, for the same observation window size, when the meshing level increases (the area of the cell becomes smaller), the accuracy and coverage decrease. This is because the regional meshing results in the loss of the association rules between nonneighbor cells and leads to unsuccessful trigger prediction of some access requests. In contrast, for the same regional meshing level, as the observation window size increases, the accuracy gradually increases while the coverage remains unchanged. This can be explained as follows. On the one hand, the larger the window size, the greater the amount of available prediction information. On the other hand, the short rules form a subset of the long rules, and the current observation window cannot trigger prediction; the size of the window will adaptively decrease for further prediction until the minimum size is reached.

(b) Type Attribute Prediction. The prediction of the type attribute is related only to the observation window size. As can be seen from Table 2, as the observation window size increases, the prediction accuracy gradually increases from 94.38% to 96.65%, while the prediction coverage remains unchanged at 96.76%. This can be explained as follows. The larger the observation window size, the greater the amount of available prediction information, and the adaptive observation window size achieves exactly the same prediction coverage.

(c) Time Attribute Prediction. The time attributes are predicted by the ARIMA model, because the change trend of the time attribute sequences comprises only three situations (remaining unchanged, changing periodically, and changing in step length). Furthermore, the adaptive observation window size makes the ARIMA model available for all the time attribute sequences. Therefore, the prediction errors appear only in the case of inconsistent change trends of sequence, and all requests falling within the observation windows can be predicted. The experimental results show that the prediction accuracy of the time attribute with different observation window sizes is 88.63%, while the prediction coverage is 100%.

(2) Access Request Prediction. The final prediction request must include three basic attributes: location, time, and type. Therefore, we should synthesize the spatiotemporal attributes predicted previously to form the access request.

The prediction accuracy and coverage of the user access requests with different regional meshing levels and observation window sizes are shown in Figures 2 and 3. We can see that, without meshing for the same observation window size, the prediction accuracy and coverage are the highest, and as the meshing level increases, the prediction accuracy gradually decreases. This is because the regional meshing can result in the loss of association rules between nonneighbor cells and reduce the probability of triggering prediction. In contrast, at the same regional meshing level, the prediction accuracy at different observation window sizes varies slightly, except for a window size of 2, where the prediction accuracy is significantly lower. And the curves for different window sizes overlap completely. These results can be explained as follows. First, the rules mined from user access are very similar when the rule length is greater than 2. Second, when the window size \( w = 2 \), it will fail to trigger joint property prediction. Finally, as stated before, the adaptive observation window size has no effect on the coverage.

Figures 4 and 5 show the prediction accuracy and coverage of the user access of the proposed prefetching scheme STAP and another two schemes, ARP [5] and MCP [15].
Regional meshing levels were set as 50 × 50 in the experiment, the observation window size is the active session window size in ARP, and the number of previous movements was monitored in MCP. As shown in Figure 4, the STAP achieves the highest prediction accuracy compared with the other two prefetching schemes. This is because the ARP and MCP can only predict user access requests that have appeared in history. When the observation window size increases gradually, prediction accuracy of the three prefetching schemes are all improved. The prediction coverage of the user access requests with different observation window size is shown in Figure 5. The prediction coverage of ARP and MCP is lower than STAP, because it cannot trigger the prediction when the user's access request has not happened in history. At the same time, as the observation window size increases gradually, the prediction coverage of ARP and MCP decreases, while STAP remains unchanged because of its adaptability.

4.3.2. Pattern Mining and Prediction Times. Figure 6 shows the time consumed for mining user access patterns from the history of user access requests. As can be seen, the time of construction decreases from 430,401 s to 21,237 s; it falls drastically as the regional meshing level increases. This is because the calculation of associate rules for the coordinate points of the entire geographic area is disaggregated to the calculation of cells and neighbor cells based on regional meshing, and
Figure 6: Patterns mining time.

Figure 7: Request prediction time.

4.3.3. Average Response Time. From the abovementioned experimental results, we can see that, in the request prediction phase, although the prediction accuracy and coverage of requests decrease under the regional meshing, the time consumed for pattern mining is effectively reduced, and more importantly, regional meshing avoids the numerous calculations required for updating the location attribute. In the data prefetching phase, it is clear that the larger the length of the buffer queue, the more credible the request for the previous prediction, and the prefetching data is more accurate. However, it also means that some data of the predicted request fail to be prefetched.

Therefore, to compare the average response time, we set the regional meshing as row \times col = 50 \times 50, the initial size of the observation window as \( w = 3 \), and the length of the buffer queue as \( \lambda = 3 \) and then test the average response time for users to access a single data item of the four schemes: STAP, ARP, MCP, and nonprefetching. From Table 3, we can see that the average response time for users to access a single data item is 0.378 ms under the nonprefetching scheme. When the prefetching mechanism is employed, the average response time is reduced obviously; the proposed scheme STAP gets the minimum average response time, with a 44.71% reduction over nonprefetching mechanism.

Table 3: The average response time (ms).

|                  | Nonprefetching | MCP   | ARP   | STAP  |
|------------------|----------------|-------|-------|-------|
| Time consumed    | 0.378          | 0.258 | 0.245 | 0.209 |

5. Conclusion

In this study, we exploited the spatiotemporal features of user access for spatiotemporal data in a smart city. We mapped the history of user access requests to the spatiotemporal attribute domain to perform correlation analysis and identify variation rules, mined the user access patterns, and developed a simple and efficient prefetching scheme. Specifically, the regional meshing method uses the spatial locality of user access; thus, they not only achieve partial and incremental solutions of association rules but also reduce the computation considerably. Furthermore, the ARIMA model uses the time stationarity of user access and realizes accurate prediction of the time attribute. Experimental results showed that our prefetching scheme is simple yet effective, and it can reduce the user access latency significantly.

Finally, the proposed concept of access pattern mining in the spatiotemporal domain for spatiotemporal data not only has a significant effect on spatiotemporal data prefetching in a smart city but also can be widely used for user-personalized recommendation, active pushing of information, and other network applications based on location services.
Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

This work was supported by the National Key Basic Research and Development Program of China (no. 2011CB302306), the National Natural Science Foundation of China (no. 61471162), and the Open-End Foundation of Hubei Collaborative Innovation Center for High-Efficiency Utilization of Solar Energy (no. HBSKFMS2014032).

References

[1] S. P. Vanderwiel and D. J. Lilja, “Data prefetch mechanisms,” ACM Computing Surveys, vol. 32, no. 2, pp. 174–199, 2000.
[2] G. Pallis, A. Vakali, and J. Pokorny, “A clustering-based prefetching scheme on a Web cache environment,” Computers and Electrical Engineering, vol. 34, no. 4, pp. 309–323, 2008.
[3] M. Wán, A. Jónsson, C. Wang, L. Li, and Y. Yang, “Web user clustering and web prefetching using random indexing with weight functions,” Knowledge & Information Systems, vol. 33, no. 1, pp. 89–115, 2012.
[4] M. Khosravi and M. J. Tarokh, “Dynamic mining of users interest navigation patterns using naive Bayesian method,” in Proceedings of the IEEE 6th International Conference on Intelligent Computer Communication and Processing (ICCP ’10), pp. 119–122, Cluj-Napoca, Romania, August 2010.
[5] M. Bamshad, H. Dai, T. Luo, and N. Miki, “Effective personalization based on association rule discovery from web usage data,” in Proceedings of the 3rd International Workshop on Web Information and Data Management (WIDM ’01), pp. 9–15, Atlanta, Ga, USA, November 2001.
[6] S. G. Matthews, M. A. Gongora, A. A. Hopgood, and S. Ahmadi, “Web usage mining with evolutionary extraction of temporal fuzzy association rules,” Knowledge-Based Systems, vol. 54, pp. 66–72, 2013.
[7] C. Jianxi, W. Qingsong, C. Cheng, and F. Dan, “Adaptive prefetching scheme for storage system in multi-application environment,” IEEE Transactions on Magnetics, vol. 49, no. 6, pp. 2762–2767, 2013.
[8] Y. Chen, S. Byna, and X. Sun, “Data access history cache and associated data prefetching mechanisms,” in Proceedings of the ACM/IEEE Conference on Supercomputing, pp. 1–12, Reno, Nev, USA, 2007.
[9] S. Ahmad and S. Hsien-Hsin, “Data prefetching mechanism by exploiting global and local access patterns,” in Proceedings of the 1st International Journal of Instructional Level Parallelism Data Prefetching Championship (DPC-1 ‘09), Raleigh, NC, USA, February 2009.
[10] Y. Chen, H. Zhu, H. Jin, and X.-H. Sun, “Algorithm-level Feedback-controlled Adaptive data prefetcher: accelerating data access for high-performance processors,” Parallel Computing, vol. 38, no. 10-11, pp. 533–551, 2012.
[11] S. Jiang, X. Ding, Y. Xu, and K. Davis, “A prefetching scheme exploiting both data layout and access history on disk,” ACM Transactions on Storage, vol. 9, no. 3, pp. 317–318, 2013.
[12] Y. Chou, “Low-cost epoch-based correlation prefetching for commercial applications,” in Proceedings of the 40th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO ’07), pp. 301–313, IEEE, Chicago, Ill, USA, December 2007.
[13] H. Tang, X. Zou, J. Jenkins et al., “Improving read performance with online access pattern analysis and prefetching,” in Euro-Par 2014 Parallel Processing, vol. 8632 of Lecture Notes in Computer Science, pp. 246–257, Springer, Basel, Switzerland, 2014.
[14] D.-J. Park and H.-J. Kim, “Prefetch policies for large objects in a web-enabled GIS application,” Data & Knowledge Engineering, vol. 37, no. 1, pp. 65–84, 2001.
[15] H. L. Dong, J. S. Kim, S. D. Kim, K. C. Kim, Y. S. Kim, and J. Park, “Adaptation of a neighbor selection Markov chain for prefetching tiled web GIS data,” in Advances in Information Systems, vol. 2457 of Lecture Notes in Computer Science, pp. 213–222, Springer, Berlin, Germany, 2002.
[16] S. Yeşilmurat and V. İşler, “Retrospective adaptive prefetching for interactive Web GIS applications,” GeoInformatica, vol. 16, no. 3, pp. 435–466, 2012.
[17] R. Li, R. Guo, Z. Xu, and W. Feng, “A prefetching model based on access popularity for geospatial data in a cluster-based caching system,” International Journal of Geographical Information Science, vol. 26, no. 10, pp. 1831–1844, 2012.
[18] J. Han, J. Pei, and Y. Yin, “Mining frequent patterns without candidate generation,” in Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data, pp. 1–12, Dallas, Tex, USA, May 2000.
[19] P. G. Zhang, “Time series forecasting using a hybrid ARIMA and neural network model,” Neurocomputing, vol. 50, no. 17, pp. 159–175, 2003.
[20] J. Contreras, R. Espinola, F. J. Nogales, and A. J. Conejo, “ARIMA models to predict next-day electricity prices,” IEEE Transactions on Power Systems, vol. 18, no. 3, pp. 1014–1020, 2003.
[21] P. Areekul, T. Senjyu, H. Toyama, and A. Yona, “Combination of artificial neural network and ARIMA time series models for short term price forecasting in deregulated market,” in Proceedings of the Transmission & Distribution Conference & Exposition: Asia and Pacific, pp. 1–4, IEEE, Seoul, The Republic of Korea, October 2009.
[22] Y. L. Huai, S. X. Chang, and Y. Liu, “A new method of prefetching I/O requests,” in Proceedings of the 2nd International Conference on Networking, Architecture, and Storage (NAS ’07), pp. 217–224, IEEE, Guilin, China, July 2007.
[23] N. Tran and D. A. Reed, “Automatic ARIMA time series modeling for adaptive I/O prefetching,” IEEE Transactions on Parallel & Distributed Systems, vol. 15, no. 4, pp. 362–377, 2004.