Communication-Efficient Distributed Multiple Reference Pattern Matching for M2M Systems

Abstract—**In the big data era, it is very common to encounter the ad hoc snapshot query that requires a fast response from many local machines in which all the data are distributed. In the scenario when the query is more complex such as multiple large patterns, the communication cost for sending the entire query to all the local machines for processing is too high. This paper aims to address this issue. Given a reference set of multiple and large-size patterns, we propose an approach to finding its k nearest and farthest neighbors. By decomposing the reference patterns into a multi-resolution representation and with the novel distance bound designs, our method guarantees the exact results in a communication-efficient manner. Analytical and empirical studies show that our method outperforms the state-of-the-art methods in saving significant bandwidth usage, especially when the number of machines and the size of reference patterns are large.**

I. Introduction

Pattern matching in distributed environments is generally considered a challenging but important task for applications relevant to machine-to-machine (M2M) systems. In such settings where a large amount of local machines are involved in computation and storage, a primary goal is often to minimize the amount of communication needed to compute the answer. This paper aims at advancing the current state-of-the-art on distributed pattern matching from 'single reference pattern' to 'multiple reference patterns', and proposes a general framework to handle both k nearest and farthest neighbor search of the multiple reference pattern set, while significantly reducing the communication cost, mainly the bandwidth consumption.

Consider a first scenario in which, through sensor data in a specific area, a scientist detects some unusual and potentially dangerous event (e.g., the dramatic oscillation of CO2 level), and wants to learn quickly whether a similar event has happened at other places. To do so, it is required to use the signal obtained by multiple sensors in one area to match sensor signals produced in the other areas. A second scenario assumes a distributed database of historical sensor readings such as the past 50 years' temperature information for many locations. Researchers might want to specify a set of time series that they identify with a certain known event (e.g., El Niño, solar activity, or the increased spread of a pest-borne disease) and query the distributed database to determine the wheres and whens of the most similar patterns. A third scenario assumes that we are monitoring certain environmental levels at many locations, and we would like to issue a warning whenever a location's pattern of recent levels deviates significantly, but perhaps subtlely, from the recent patterns at a set of reference locations, because it might indicate an abnormal environmental event is happening at that location (e.g., hazardous material being improperly transported).

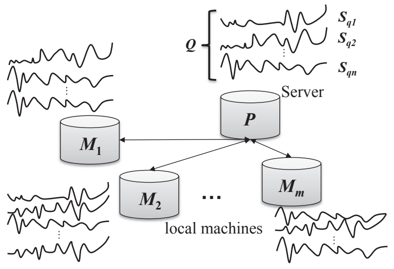


Fig. 1. The system model

Designing a general framework that can handle the above scenarios requires addressing several challenges. First, the query to be matched may consist of multiple patterns, in order to provide a more robust reference set beyond what any one pattern might provide. For example, there may be more than one "signature" pattern for an event or more than one nearby sensor monitoring an event. The multiple patterns in a single query may be highly correlated, such as when collected by nearby sensors (1^{st} and 2^{nd} scenarios), or only moderately correlated, such as when collected from a set of reference locations (3^{rd} scenario). Second, the query to be matched is often a one-time (i.e., snapshot) query, either posed as an ad hoc query (1^{st} and 2^{nd} scenarios) or as a continuous sequence of queries such that recent readings determine the next query in the sequence (3^{rd} scenario). Third, we need to handle not only *similarity* (1^{st} and 2^{nd} scenarios) but also *dissimilarity* (3^{rd} scenario) search. Fourth, we seek the k most similar (or k most dissimilar) neighbors from across a potentially large collection of *distributed* data sources. Finally, because in many situations there are bandwidth limitations and concerns of energy consumption as well as cost during transmission, it is usually critical to design an approach that requires as little transmission between sites as possible. To be more precise, our goal is to reduce the bandwidth consumption while not missing any k most similar/dissimilar neighbors.

To address the above challenges, we propose a new framework that, given multiple reference patterns, allows us to find their exact k-nearest (most similar) and k-farthest (most dissimilar) neighbors, denoted as kNN and kFN, in a distributed environment where bandwidth is limited. In M2M applications such as the aforementioned three scenarios, a huge amount of measurement readings over a period of time are collected. Therefore the multiple reference patterns to be dealt with in this papaer are mainly multiple time series. The system diagram can be seen in Fig. 1, where there are m distributed machines, each monitoring one or more series of measurement readings, and a server orchestrating the processing of kNN and kFN discoveries. Given a set Q of multiple time series patterns as the query at the server, the goal is to find a set of k time series among all m local machines with the highest similarity (or dissimilarity) to the query. Our primary cost metric is the total number of bytes exchanged between the server and the local machines to answer the query. We do not explicitly model the small cost that may be required to assemble the query at the server, e.g., in the 3^{rd} scenario. Also, while we do not explicitly model response time, our solutions are highly parallel and fast.

Handling queries that contains multiple instances has attracted certain level of attention in the past years. In multiple pattern matching in text search or bioinfomatics applications, the inputs are assumed to be multiple strings and the algorithms report all occurrences of the input strings. It is different from our goal in finding kNN/kFN in continuous time series while limiting bandwidth consumption. Furthermore, the assumption of penalization aims at speeding up the process, rather as our case to assume data are coming from distributed sources. Another string of research along the line is called the exemplar-based learning or pattern-based event intrusion detection. They assume all patterns are similar to some extent to perform learning or anomaly detection, while in our framework such assumption is released and we consider a more challenging setup in distributed data sources. Furthermore, our unified framework allows us to handle both similarity and dissimilarity matching, which have been treated as two independent problems in most of the previous works.

A naive solution to kNN and kFN search of a set of multiple reference patterns is to transmit the whole query set Q from the server to all local sites. After receiving the queries, each local site computes the similarity between the locally maintained measurement readings and the query set and reports the k nearest/farthest neighbors to the server. Finally, the server determines the true kNN/kFN after it receives $m * k$ results from m peer sites. This kind of scheme was called Concurrent Processing (CP) in [1] and a probabilistic processing (PRP) method, which reduces the amount of data required to be transmitted back to server from local sites, was further suggested by the authors to save the bandwidth cost of CP. However, sending the all the patterns in the query set to a large number of sites still consumes huge bandwidth. To handle this problem, a method called LEEWAVE has been proposed to leverage the multi-resolution property of the Haar wavelet transformation of time series [2]. However, LEEWAVE only deals with single time series pattern as the query input and did not discuss how to deal with kFN queries. In this work, we will prove that LEEWAVE cannot completely handle the multiple reference pattern matching query and propose a generalized and unified framework for solving this problem.

Our framework, called MSWAVE, is designed based on the following insights. First, to handle multiple reference patterns as queries, we propose three distance metrics, namely *single-linkage distance*, *average-linkage distance*, and *complete-linkage distance*, which report the shortest, average, and largest distances among all the distances of a candidate time series to each reference pattern in the query set. The above three distance measures analogize the single-link, average-link, and complete-link clustering models. Second, based on the characteristics of Haar Wavelet transformation, MSWAVE pre-processes each series of measurement readings by decomposing it into a multi-resolution representation. Instead of sending the whole query set to the local sites, the server is designed to send them in a level-wise manner from the coarse resolutions with fewer data to the finer one with more data. We further derive and maintain certain similarity range bounds for each of the three distance measurements. The upper and lower bounds can be incrementally updated. More importantly, a similarity range can be proven to gradually become tighter as we move from one Wavelet coefficients level to the next. These increasingly tightened similarity ranges enable effective pruning of the candidate time series that leads to decent improvement over the bandwidth consumption without causing any false dismissal. Although Wavelet level-wise pruning strategy has been proposed in the literature [2], [3], in this paper we not only generalize it to multiple reference patterns scenario but also make several changes that yields significant enhancement. Different from the previous proposal that requests the computation of similarity bounds on the server side, MSWAVE employs a strategy that allows it to shift the computation to the local site. By doing so, we can further introduce tighter bounds as well as a different transmission mechanism that allows us to save more bandwidth.

We conduct a theoretical analysis of MSWAVE to show its bandwidth savings. We also conduct extensive experiments using both real and synthetic data. The results show that our solution significantly outperforms the competitive approaches in total bandwidth consumption in a variety of different setup for searching both kNN and kFN.

Our contributions can be summarized as follows:

1. We propose MSWAVE, a general communication-efficient framework to identify both kNN and kFN instances given multiple reference patterns, which are multiple time series, in a distributed environment. To our knowledge, this is the first solution proposed for such purpose.

2. Methodology-wise, we propose to use average, closest, and furthest neighbor distance to process multiple query (dis)similarity, analogical to average-link, single-link, and complete-link in clustering. We then take advantage of the multiple-resolution property of wavelet coefficients, and then for each distance metrics we derive upper and lower bounds of the similarity between each candidate time series to the query (i.e., multiple reference patterns). Such bound can be exploited to prune impossible candidates for more efficient search. Finally, in further contrast to prior approaches, we propose to shift the computation from the server to local sites to further reduce the bandwidth consumption.

3. We conduct theoretical analysis and proofs to validate several of our arguments, including the infeasibility/feasibility of the existing/proposed approaches, and derive the equation to represent the theoretical gain by shifting the bound computation to local sites. We further conduct extensive experiments to evaluate MSWAVE's bandwidth savings.

II. PRELIMINARIES

In this section we first describe the state-of-the-art approach, LEEWAVE, for answering distributed kNN queries for a *single* time series. Then we formally define our distributed *multiple* time series query problem and discuss why the prior approach is inadequate for dealing with the proposed problem.

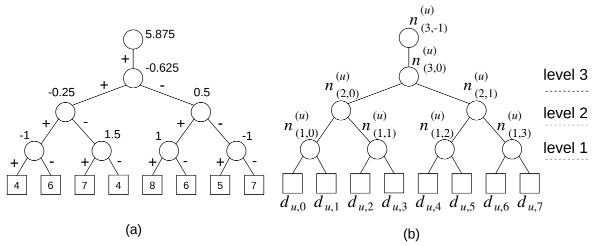


Fig. 2. (a) The error tree for a time series {4,6,7,4,8,6,5,7}; (b) The notation of an error tree proposed in [9].

A. Distributed kNN for Single Time Series

The conventional kNN (or kFN) search for a single reference time series aims at finding k time series of the smallest (largest) distance to a given reference time series. In this paper, we will focus on the Euclidean distance metric: Given two time series S_{ref} and S_x of length T, $Dst(S_{ref}, S_x)$ is defined to be the squared sum of the Euclidean distance between them. Namely,

$$Dst(S_{ref}, S_x) = \sum_{i=1}^{T} (S_{ref}[i] - S_x[i])^2, \qquad (1)$$

where $S_{ref}[i]$ and $S_x[i]$ are the values of S_{ref} and S_x, respectively, at timestamp i.

To deal with time series matching problems, Wavelet transformation, especially the Haar Wavelet [4], has been applied in a variety of studies such as [5], [3], [6], [2], [7]. In Haar wavelet decomposition, each time series is decomposed into multiple resolutions and can be represented using an error tree structure [8], as shown in Fig. 2(a). Note that only the non-leaf node coefficients are retained. The notation $n^{(u)}_{(l,p)}$ shown in Fig. 2(b) is used to represent the coefficient at level l having offset p of time series S_u.

Given wavelet coefficients of two time series, $Dst(S_{ref}, S_x)$ can be generated directly from the coefficients themselves:

$$Dst(S_{ref}, S_x) = \sum_{l}((\sum_{p}[n^{(ref)}_{(l,p)} - n^{(x)}_{(l,p)}]^2) \times 2^l). \qquad (2)$$

Furthermore, the distance can be generated in a top-down level-wise manner as suggested in LEEWAVE [2]:

$$\begin{aligned} Dst(S_{ref}, S_x) &= \sum_{l=1}^{L} Dst^l(S_{ref}, S_x) \\ &= accDst^{\rho}(S_{ref}, S_x) + \sum_{l=1}^{\rho-1} Dst^l(S_{ref}, S_x), \end{aligned} \qquad (3)$$

where

$$Dst^l(S_{ref}, S_x) = \sum_{p}[n^{(ref)}_{(l,p)} - n^{(x)}_{(l,p)}]^2 \times 2^l, \qquad (4)$$

$$accDst^{\rho}(S_{ref}, S_x) = \sum_{l=\rho}^{L} Dst^l(S_{ref}, S_x), \qquad (5)$$

ρ represents the current level, and L is the height of the error tree.

In LEEWAVE, instead of simultaneously distributing all the relevant coefficients of the reference series S_{ref} to all the local sites, the server only sends the coefficients one level at a time, starting from the top (the coarsest) level[1]. Each local site responds with the level distance $Dst^l(S_{ref}, S_x)$ for its time series S_x. Using such information, the server can determine the similarity range (i.e., upper/lower bounds) of each local time series to the reference series:

$$\begin{aligned} &accDst^{\rho}(S_{ref}, S_x) \le Dst(S_{ref}, S_x) \le \\ &\quad accDst^{\rho}(S_{ref}, S_x) + \sum_{l=1}^{\rho-1}\sum_{p}([n^{(ref)}_{(l,p)}]^2 + [n^{(x)}_{(l,p)}]^2) \times 2^l \\ &\quad + 2 \times \sqrt{\sum_{l=1}^{\rho-1}\sum_{p}([n^{(ref)}_{(l,p)}] \times 2^l)^2 \times \sum_{l=1}^{\rho-1}\sum_{p}[n^{(x)}_{(l,p)}]^2}. \end{aligned} \qquad (6)$$

Based on these bounds, the server progressively (level-wise) informs each local site as to whether its time series is still a candidate for kNN, and if not, the site drops out of the computation.

Note that the server can compute the lower and upper bounds in Eq. (6) from its $n^{(ref)}_{(l,p)}$ terms and three summation terms provided by a local site. This saves considerable bandwidth compared to the local site sending its complete time series. Also, it has been proven in [2] that the derived upper bound is non-increasing and the lower bound is non-decreasing when moving from one level to the next. These increasingly tightened similarity ranges enable effective pruning of candidates without any false dismissals.

B. Defining Distributed kNN/kFN Search for Multiple Time Series

Before discussing the limitations of the above framework, we first formulate the distributed kNN and kFN search problem for multiple time series.

Let $Q = \{S_{q1}, \ldots, S_{qn}\}$ be a set of n reference time series of length T. To match a candidate time series to the given set of multiple time series, we propose the following three linkage distances.

Definition 1: The *single-link*, *average-link*, and *complete-link* distances of a time series S_x to a reference set $Q = \{S_{q1}, \ldots, S_{qn}\}$ are defined as

$$\begin{aligned} d_{sin}(Q, S_x) &= \min_{1\le i\le n} Dst(S_{qi}, S_x), \\ d_{avg}(Q, S_x) &= \sum_{i=1}^{n} Dst(S_{qi}, S_x)/n, \text{ and} \\ d_{com}(Q, S_x) &= \max_{1\le i\le n} Dst(S_{qi}, S_x). \end{aligned} \quad ■$$

The idea is analogous to the single-link, average-link, and complete-link distances used in clustering. Based on the idea, a time series S_x is considered close to a group of time series Q if either there exists a certain time series in Q that is very similar to S_x (i.e., d_{sin}), or most of the time series in Q have

[1]Please note that when the length of time series T is not a power of 2, the corresponding wavelet coefficients can be represented with multiple error trees of different heights because any integer value can always be represented as the summation of different powers of two. In this case, sending coefficients that may come from different sub-trees in a level-wise manner still works.

to be close enough to S_x (i.e., d_{avg}), or the most dislike time series has to be still somehow similar to S_x (i.e., d_{com}).

With Definition 1, we can now define the kNN and kFN search problem for multiple time series queries.

Definition 2: Given a server P with a reference time series set $Q = \{S_{q1}, \ldots, S_{qn}\}$, each of length T, and a set of distributed local machines $M_1, \ldots, M_m$, each with one or more time series of length T, a *distributed kNN (kFN) search for query* Q is to find the exact k time series among all the machines that have the smallest (largest, respectively) linkage distance, either single-link, average-link, or complete-link as predefined by the user. ∎

C. Limitations of the Existing Framework

One immediate question is whether the LEEWAVE framework from Section II-A can be exploited directly to handle the multiple time series case. A simple idea would be to use LEEWAVE independently for each of the time series in Q, and then try to use these answers to re-construct the overall kNN according to d_{sin}, d_{avg}, or d_{com}. We call this framework LEEWAVE-M. Unfortunately, such strategy cannot guarantee exact solutions except for the single-link measure in kNN and complete-link measure in kFN. Let us see a simple counter example for 1NN search.

Example 1: Suppose we have a two length-1 reference time series S_{q1} = {2}, S_{q2} = {-2}, and candidate time series S_1 = {0}, S_2={3}, and S_3={-3} stored in local machine M_1 and candidate time series S_4={4} and S_5={5} stored in local machine M_2. For M_1, S_2 gets returned for S_{q1} and S_3 for S_{q2}; while for M_2, S_4 gets returned for both reference series. Considering both sites, S_2 is the 1NN for S_{q1} and S_3 is the 1NN for S_{q2}. However, the true 1NN results under $d_{avg}(Q, S_x)$ and $d_{com}(Q, S_x)$ are both S_1, which was not even selected to be returned to the server.

Similarly, we can find counter examples for kFN under the average and single-link distance.

Besides the limitation of being able to solve only 2 out of the 6 cases (i.e., 3 linkage distance metrics for 2 kinds of NN queries) proposed here, directly apply LEEWAVE-M cannot be considered a bandwidth-efficient approach because each reference series in Q is processed independently.

The third limitation of LEEWAVE lies in its server-oriented computation strategy. Most of the bounds are calculated on the server site based on sum terms sent by the local sites; for the multiple time series scenario, this strategy wastes bandwidth.

In this paper, we introduce the MSWAVE framework to deal with the above limitations.

III. MSWAVE ALGORITHM AND ANALYSIS

In MSWAVE, we also leverage the multi-resolution property of the Haar wavelet decomposition of time series. The server P distributes the reference time series set $Q = \{S_{q1}, \ldots, S_{qn}\}$ in a level-wise manner. That is, P sends the coefficients of each $S_{qi} \in Q$ to the local machines, one level at a time starting from the highest level L. At each level, we further prune the candidates, until the final k answers are found. While similar to LEEWAVE at this high level, MSWAVE must overcome the limitations outlined in the prior section. To do this, first we derive new formulas for computing the similarity ranges of the three linkage distances between the reference set and a candidate time series (Section III-A). These ranges must be effective at pruning yet guarantee no false dismissals. Second, we devise a correct and bandwidth-efficient protocol for the data exchanges between the server and the multiple local machines (Section III-B). We present two variants: MSWAVE-S, which computes the bounds at the server, and MSWAVE-L, which computes the bounds at the local machines. Finally, we provide a theoretical analysis of the bandwidth consumption of both variants, which demonstrates the effectiveness of MSWAVE at reducing bandwidth (Section III-C).

A. Computation of Distance Bounds

We start from the similarity range of the distance between each individual reference time series S_{qi} to some candidate S_x. Similar to Eq. (6), we can derive the upper bound UB and the lower bound LB of $Dst(S_{qi}, S_x)$ when all the coefficients for S_{qi} from the highest level L to the current level ρ are distributed out from the server P as follows.

$$LB(qi,x) = accDst^{\rho}(S_{qi}, S_x). \tag{7}$$

$$\begin{aligned} UB(qi,x) &= accDst^{\rho}(S_{qi}, S_x) \\ &+ \sum_{l=1}^{\rho-1}\sum_{p}([n_{(l,p)}^{(qi)}]^2 + [n_{(l,p)}^{(x)}]^2) \times 2^l \\ &+ 2 \times \min\Bigg(\sqrt{\sum_{l=1}^{\rho-1}\sum_{p}([n_{(l,p)}^{(qi)}] \times 2^l)^2 \times \sum_{l=1}^{\rho-1}\sum_{p}([n_{(l,p)}^{(x)}])^2}, \\ &\sqrt{\sum_{l=1}^{\rho-1}\sum_{p}([n_{(l,p)}^{(x)}] \times 2^l)^2 \times \sum_{l=1}^{\rho-1}\sum_{p}([n_{(l,p)}^{(qi)}])^2}\Bigg). \end{aligned} \tag{8}$$

Note that Eq. (8) is an enhanced version of the upper bound compared to that in Eq. (6). Because the roles of S_{qi} and S_x are interchangeable in the squared terms in Eq. (6), a tighter upper bound is obtained by choosing the minimum among the two choices. Our experiments will show that such subtle change noticably improves the pruning performance. This new bound does not violate the non-increasing property proved in [2] because the smaller bound from two non-increasing bounds is chosen here.

Now we can derive the similarity range for each linkage distance defined in Definition 1. For $d_{avg}(Q, S_x)$, the average of all $Dst(S_{qi}, x)$ for all $S_{qi} \in Q$, because the distance is non-negative, we can simply derive the new bound as follows.

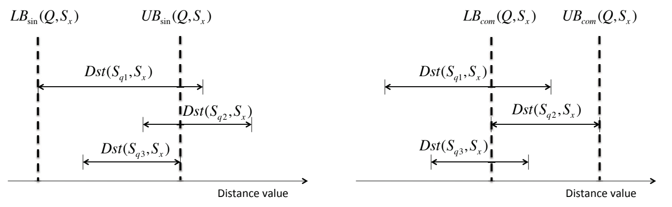
$$LB_{avg}(Q, S_x) = \frac{1}{|Q|}\sum_{i=1}^{|Q|} LB(qi, x) \tag{9}$$

$$UB_{avg}(Q, S_x) = \frac{1}{|Q|}\sum_{i=1}^{|Q|} UB(qi, x) \tag{10}$$

For $d_{sin}(Q, S_x)$, the two bounds are

$$LB_{sin}(Q, S_x) = \min_{1 \le i \le |Q|} LB(qi, x) \text{ and} \tag{11}$$

$$UB_{sin}(Q, S_x) = \min_{1 \le i \le |Q|} UB(qi, x). \tag{12}$$



(a) The upper and lower bounds for $d_{sin}(Q, S_x)$.

(b) The upper and lower bounds for $d_{com}(Q, S_x)$.

Fig. 3. Upper/lower bound computation for different linkage distance metrics.

For $d_{com}(Q, S_x)$, the two bounds are

$$LB_{com}(Q, S_x) = \max_{1 \le i \le |Q|} LB(qi, x) \text{ and} \tag{13}$$

$$UB_{com}(Q, S_x) = \max_{1 \le i \le |Q|} UB(qi, x). \tag{14}$$

The illustration of the bounds for $d_{sin}(Q, X)$ is shown in Fig. 3(a). As we want to choose the closest distance of a candidate time series to the reference set Q, we can set the lower (upper) bounds of $d_{sin}(Q, X)$ using the smallest ones among all $LB(qi, x)$ ($UB(qi, x)$, respectively), and for $d_{com}(Q, X)$ using the largest ones among all $LB(qi, x)$ ($UB(qi, x)$), as shown in Fig. 3(b).

Now we prove the similarity ranges bounded by these lower and upper bounds will shrink as more levels of coefficients are disseminated to local machines.

Theorem 1: $UB_{avg}(Q, S_x)$ is non-increasing and $LB_{avg}(Q, S_x)$ is non-decreasing when the coefficients of $S_{qi} \in Q$ are disseminated from level ρ to level $\rho - 1$

Proof: As argued above, it readily follows from [2] that Eq. (7) is non-decreasing and Eq. (8) is non-increasing. Therefore, $LB_{avg}(Q, S_x)$ must be non-decreasing and $UB_{avg}(Q, S_x)$ must be non-increasing as they are each just the average of a set of such non-decreasing and non-increasing bounds. ■

Theorem 2: $UB_{sin}(Q, S_x)$ is non-increasing and $LB_{sin}(Q, S_x)$ is non-decreasing when the coefficients of $S_{qi} \in Q$ are disseminated from level ρ to level $\rho - 1$

Proof: Let $l(\rho)$ and $u(\rho)$ be the reference time series in Q that have the smallest lower bound and smallest upper bound of the similarity range to S_x at level ρ. That is,

$$\begin{aligned} l(\rho) &= \arg\min_{1 \le i \le |Q|} LB(qi, x)|_\rho \quad \text{and} \\ u(\rho) &= \arg\min_{1 \le i \le |Q|} UB(qi, x)|_\rho, \end{aligned}$$

where $|_\rho$ represents the corresponding bound values are derived at level ρ. We have

$$\begin{aligned} LB_{sin}(Q, S_x)|_\rho = LB(q_{l(\rho)}, x)|_\rho \le LB(q_{l(\rho-1)}, x)|_\rho \\ \le LB(q_{l(\rho-1)}, x)|_{\rho-1} = LB_{sin}(Q, S_x|_{\rho-1}), \end{aligned}$$

where the first inequality holds because $l(\rho)$ is the arg min at level ρ and the second inequality holds because Eq. (7) is non-decreasing. Thus, $LB_{sin}(Q, S_x)$ is non-decreasing.

Similarly, because Eq. (8) is non-increasing, a symmetric argument shows that $UB_{sin}(Q, S_x)$ is non-increasing. ■

Finally, the symmetry between the bounds for d_{syn} and d_{com} yields the following.

Theorem 3: $UB_{com}(Q, S_x)$ is non-increasing and $LB_{com}(Q, S_x)$ is non-decreasing when the coefficients of $S_{qi} \in Q$ are disseminated from level ρ to level $\rho - 1$.

B. The MsWave *Protocol*

We are now ready to describe the details of how MsWave processes a distributed kNN or kFN multiple time series query in a level-wise manner and how P progressively prunes the candidates. More importantly, we will introduce two schemes to solve this problem. We will describe what data are exchanged between the server and local machines before presenting the pruning strategy for both kNN and kFN.

1) MsWave-*S: Server Computes Bounds:* Fig. 4 presents the MsWave-S protocol. At the initial step, the server P sends the highest level-L coefficients of each S_{qi} in Q to all M local machines. Each local machine then extracts wavelet coefficients of the same level for each time series to be matched. It then returns the following numbers for each local candidate time series S_x: the level-L distance, $Dst^L(S_{qi}, S_x)$, for $i = 1$ to $|Q|$, and three other numbers that will be used by P to generate the bounds for pruning: $\sum_{l=1}^{L-1} \sum_p [n_{(l,p)}^{(x)}]^2$, $\sum_{l=1}^{L-1} \sum_p ([n_{(l,p)}^{(x)}]^2 \times 2^l)$ and $\sum_{l=1}^{L-1} \sum_p ([n_{(l,p)}^{(x)}]^2 \times 2^l)^2$. After receiving these numbers from each candidate time series, P updates the lower and upper bounds based on Eq. (7) and Eq. (8) for each candidate time series.

It will then do some initial pruning to remove any candidates that cannot be among the top k neighbors. To prune candidates for kNN queries, P first sorts the candidate time series in an ascending order based on the upper bounds. Any candidate time series whose similarity lower bound is higher than the upper bound of the k^{th} time series in the sorted list cannot be in the final answer, and thus is pruned. As the bound is proved to be monotonically shrinking with the increasing of level, we can guarantee that there is no false dismissal under this pruning strategy. Similarly, for kFN query, any candidate time series whose similarity upper bound is smaller than the k^{th} largest lower bound cannot be in the final answer. Then, it moves to the next level.

For any given level l, P sends the level-l coefficients of each $S_{qi} \in Q$ and the id of the pruned time series to the local machines. The local machine will return two level-specific numbers for each candidate time series: $Dst^l(S_{qi}, S_x)$ for $i = 1$ to $|Q|$ and $\sum_p [n_{(l,p)}^{(x)}]^2$. P will then use them to update the upper/lower bounds, always making them tighter. With the bounds of each candidate series to each reference time series, P will further compute the similarity range of a linkage distance predefined by users of each series to the reference set Q based on Eq. (9)–(14). With increasingly tighter ranges, P can better prune the candidate list. The algorithm ends when there are no more than k candidate time series left.

2) MsWave-*L: Local Machines Compute Bounds:* Note that with MsWave-S, the local machines consume certain bandwidth to send back the level distances of each time series

Procedure: MsWave-S for a kNN/kFN multiple time series query
Input: k, $Q = \{S_{q1}, \ldots, S_{qn}\}$, a linkage distance (single, average or complete)
Output: The k most similar/dissimilar time series to Q according to the designated linkage distance

The server P:	*A local machine M_i:*
1. Send coefficients of each $S_{qi} \in Q$ at level L to all M local machines.	2. For each local candidate time series, S_x, compute and return $(Dst^L(S_{qi}, S_x)\ \forall S_{qi} \in Q$, $\sum_{l=1}^{L-1}\sum_p [n_{(l,p)}^{(x)}]^2$, $\sum_{l=1}^{L-1}\sum_p([n_{(l,p)}^{(x)}]^2 \times 2^l)$, $\sum_{l=1}^{L-1}\sum_p([n_{(l,p)}^{(x)}]^2 \times 2^l)^2)$ to P.
3. Compute the upper and lower bounds based on Eq. (7)-(8) for each candidate time series to each reference series. Then compute the similarity range for each candidate time series to Q according to the designated linkage distance based on Eq. (9)-(14). Do the first pruning.	
4. Repeat steps 5–7 for levels $l = L-1, L-2, \ldots, 1$ until **done**{	
5. Send level coefficients of each $S_{qi} \in Q$ and the id of the pruned candidate series to local machines.	6. Compute and return a 2-tuple $(Dst^l(S_{qi},\ S_x)\ \forall S_{qi} \in Q$, $\sum_p [n_{(l,p)}^{(x)}]^2)$ for each local candidate time series, S_x.
7. Update the upper and lower bounds based on Eq. (7)-(8) for each candidate time series to each reference series. Then update the bounds of the linkage distance based on Eq. (9)-(14). Do corresponding pruning for kNN or kFN. Set **done** to **true** if there are no more than k candidate time series left.	
8. }	
9. Ask the local machines for the contents of the final k time series.	10. Send back corresponding contents.

Fig. 4. Protocol for distributed kNN/kFN query processing using MsWave-S.

Procedure: MsWave-L for a kNN/kFN multiple time series query
Input: k, $Q = \{S_{q1}, \ldots, S_{qn}\}$, a linkage distance (single, average or complete)
Output: The k most similar/dissimilar time series to Q according to the designated linkage distance

The server P:	*A local machine M_i:*
1. Send level L coefficients, $\sum_{l=1}^{L-1}\sum_p [n_{(l,p)}^{(qi)}]^2$, $\sum_{l=1}^{L-1}\sum_p([n_{(l,p)}^{(qi)}]^2 \times 2^l)$, and $\sum_{l=1}^{L-1}\sum_p([n_{(l,p)}^{(qi)}]^2 \times 2^l)^2$ of each $S_{qi} \in Q$ to all M local machines.	2. For each local candidate time series, S_x, compute the individual bounds with each reference time series using Eq. (7)-(8). Then compute and return the two linkage distances bounds based on Eq. (9)-(14).
3. Do the first pruning by sorting the upper/lower bounds of each candidate series.	
4. Repeat steps 5–7 for levels $l = L-1, L-2, \ldots, 1$ until **done**{	
5. Send level coefficients of each $S_{qi} \in Q$ and the id of the pruned candidate series to local machines.	6. Update the corresponding upper/lower bounds based on the computation defined in Eq. (9)-(14) and return the two linkage distance bounds for each local candidate time series.
7. Do corresponding pruning for kNN or kFN according to the updated bounds of each candidate series sent back from the local machines. Set **done** to **true** if there are no more than k candidate time series left.	
8. }	
9. Ask the local machines for the contents of the final k time series.	10. Send back corresponding contents.

Fig. 5. Protocol for distributed kNN/kFN query processing using MsWave-L.

to multiple reference time series, i.e., $Dst^l(S_{qi}, S_x)$ for $i = 1$ to $|Q|$, which grows linearly with Q. When $|Q|$ becomes large, the MsWave-S protocol might not be as efficient.

To deal with this issue, we propose another scheme, MsWave-L, which computes the similarity bounds of the linkage distance at the local machines. By doing so, we need not send the level distances for each reference time series, but instead only 2 single bound values for the whole query set. Compared to the MsWave-S protocol, MsWave-L improves bandwidth consumption by a factor of $\Theta(Q)$.

Fig. 5 presents the protocol. In the initial step, the server P sends to the local machines not only the coefficients at level L, but also three additional numbers for each reference time series $S_{qi} \in Q$, which will later allow the local sites to generate the similarity ranges: $\sum_{l=1}^{L-1}\sum_p [n_{(l,p)}^{(qi)}]^2$, $\sum_{l=1}^{L-1}\sum_p([n_{(l,p)}^{(qi)}]^2 \times 2^l)$ and $\sum_{l=1}^{L-1}\sum_p([n_{(l,p)}^{(qi)}]^2 \times 2^l)^2$. After receiving these values, the local machines can compute the similarity bounds of each candidate series to Q based on Eq. (9)–(14) according to different linkage distances. Then, each local machine sends back only the two bound values for each candidate series to the server. With the bounds of each candidate, the server P can then do corresponding pruning to tell the local machines which candidates cannot be in the final k list and can be discarded. Such procedure goes iteratively until the final results are produced. Note that the pruning strategy is the same as that done in MsWave-S.

C. Theoretical Analysis

Here we analyze the bandwidth consumption of MsWave-S and MsWave-L. Suppose there are total s time series distributed in m local machines, $|Q|$ reference time series, and the length of each time series is T. By Haar wavelet decomposition, there will be $L = \log_2 T$ levels of wavelet coefficients for each time series, and there are $2^{L-\rho+1}$ number of coefficients at some level ρ. As a result, we have the bandwidth saved by MsWave-L compared to MsWave-S as follows.

$$-3|Q|m + s(4|Q|-2) + \sum_{\rho} 2s_{\rho}(|Q|-1)$$

$$= \qquad |Q|(4s - 3m + 2\sum_{\rho} s_{\rho}) - 2s - 2\sum_{\rho} s_{\rho} \geq 0, \qquad (15)$$

where s_{ρ} is the number candidate time series left at level ρ. The term $-3|Q|m$ refers to the three more summation values sent by P to local machines in step 1 in MsWave-L. The term $s(4|Q|-2)$ is the bandwidth saved by transmitting only the 2 bounds for each $\{Q, S_x\}$ instead of all pair-wise reference-candidate level distances and the related summation values in step 2. The final summation term describes the bandwidth saved at the following steps when more and more levels of coefficients are disseminated.

It is possible to show that Eq. (15) is always greater than zero. Since in general cases $s \geq m$ (otherwise the data do not have to be distributed to m machines), we have $|Q|(4s - 3m) - 2s > (|Q|-2)s \geq 0$, because $|Q| > 1$ for multiple time series, and the final term $2s(|Q|-1)$ is also greater than zero.

In summary, as the size of Q is getting larger, MsWave-L shows more superiority in saving the total bandwidth consumption.

IV. Experiments

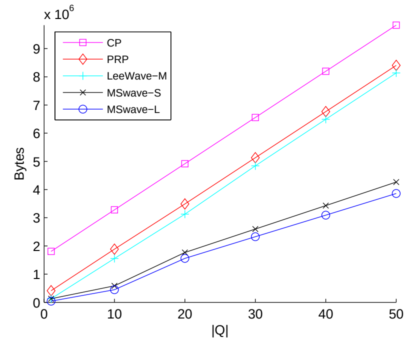
The goal for this section is twofold. First, we would like to compare the bandwidth consumption of our models with some baseline and state-of-the-art models. Second, we intend to provide some discussions on a variety of scenarios and configurations.

A. Data Description and Experiment Setup

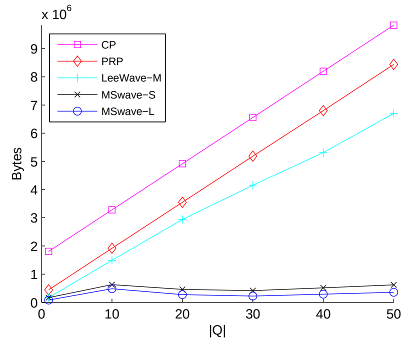
We use one real data set and one synthetic data set in our experiments. For the real data set, we choose a public dataset recording the daily average temperature of 300 cities around the world acquired from the temperature data archive of University of Dayton. The data from each city is considered as a time series with 2048 data points. For the sysnthetic data set, we use the same random walk data model used in [10]. Each time series is generated by the random walk whose every step size is a normal distributed random number with 0 mean and 1 standard deviation. There are 12,500 time sereis of length 12,500 generated. After picking $|Q|$ streams as the reference set, the candidate series were chosen from the rest all-$|Q|$ ones and are equally distributed to the m sites.

We compare the total bandwidth cost for MsWave-L, MsWave-S, LeeWave-M, CP and PRP in the distributed environment simulated in MATLAB. We stressed on the influences of the size of reference set $|Q|$, window size T, number of sites m, and k for kNN/kFN on the bandwidth cost. The total bandwidth cost is the summation of all data transmitted between the server and other local sites. Note that in this simulation framework, practical issues such as overhead of the headers, package loss, retransmission, etc were not considered.

There are two strategies we employed to choose the multiple reference time series for querying: the series with high similarity (denote as *analogous reference set*) and randomly chosen series (denote as *random reference set*). We produce the



(a) random reference set



(b) analogous reference set

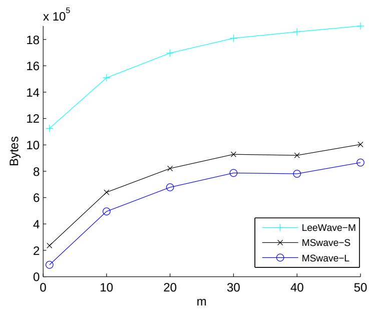
Fig. 6. Comparison between models given two kinds of queries, T=1024, k=10, m=20, d_{com}, kFN

former reference set by first choosing one time series randomly and then choose its closest —Q—-1 neighbors to form the multiple time series reference set.

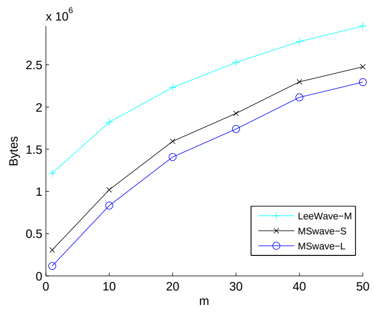
B. Model Comparison on Real Data

MsWave-L vs. MsWave-S vs. LeeWave-M vs. CP vs. PRP Here we show a comparison of all models in total transmission cost in Fig. 6. Both plots show that MSWAVE models outperform the other models significantly. We also found the slopes of MSWAVEs are less steep than the slopes of the others, highlighting the benefits of using Eq. (9)∼(14) for pruning. For analogous reference sets, we even find that the results of MsWave models do not increase significantly when $|Q|$ increases. It is due to that when reference time series are to some extend similar, the distance to any one of the reference time series would be very close to the distance to the whole reference set. This holds regardless of whether d_{avg}, d_{sin}, or d_{com} is chosen. Therefore, in the early level of transmission, we already obtain enough information to estimate the true distance, which leads to more effective pruning. On the other hand, for random reference set because the distances to each reference varied a lot, it is generally required to send many levels of coefficients to be able to accurately estimate the true distance to determine the kNN/kFN neighbors, thus consumes more bandwidth.

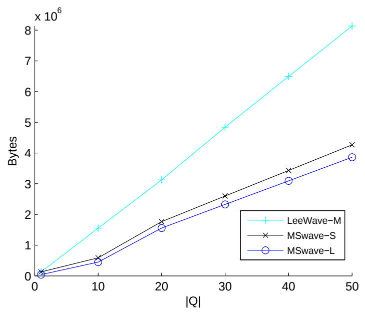
Comparing MsWave-S , MsWave-L and LeeWave-M. Here we particularly highlight the comparison of MsWave-S, MsWave-L, and LeeWave-M in total transmission cost measured in bytes. We choose kNN for d_{sin} (Fig. 7(b), 7(d), 7(f)) and kFN for d_{com} (Fig. 7(a), 7(c), 7(e)) measure because as have been shown Section 2.3 they are the



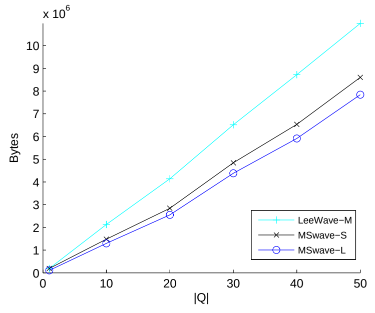
(a) k=10, $|Q|$=10, d_{com}, kFN



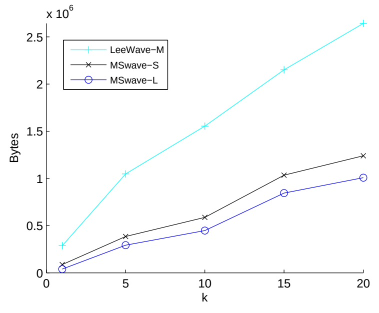
(b) k=10, $|Q|$=10, d_{sin}, kNN



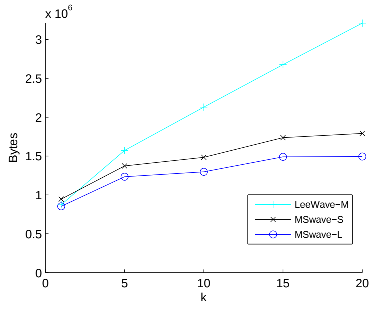
(c) k=10, m=20, d_{com}, kFN



(d) k=10, m=20, d_{sin}, kNN



(e) m=20, $|Q|$=10, d_{com}, kFN



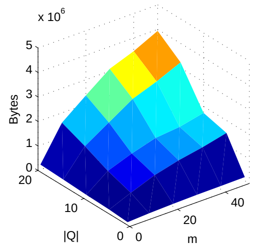
(f) m=20, $|Q|$=10, d_{sin}, kNN

Fig. 7. Comparison between models given the random reference set, T=1024.

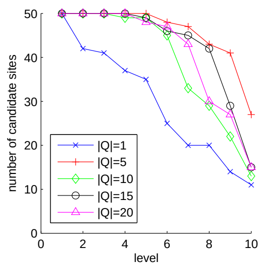
only two feasible cases for LEEWAVE-M. As shown in Fig. 7, we can see the performance of MSWAVE-L is clearly better than MSWAVE-S, while both are much better than LEEWAVE-M in all configurations.

Fig. 7(a) and 7(b) discusses the results while varying m. It shows increasing m does not change the difference of performance significantly. In the next experiment, we verified the impact of the size of reference set $|Q|$ which is varied from 1 to 50 (Fig. 7(c) and 7(d)). As we can see from the figures, the difference between LEEWAVE-M and MSWAVE-L increases as $|Q|$ increases. On the other hand, the difference between MSWAVE-L and MSWAVE-S also increases as $|Q|$ becomes larger since the larger the reference set is, the more values MSWAVE-S has to send from local sites to the server. The results verified our discussion in Section 3.3. Then, we examined the impact of k on the performance (Fig. 7(e) and 7(f)). It is not hard to reason that the difference between LEEWAVE-M and MSWAVE-L in transmission cost increases as k increases because the overhead grows linearly with k. We can also see that the difference between MSWAVE-L and LEEWAVE-M also increases as k becomes larger. The larger k is, the fewer candidate series are pruned, and based on our discussion in Section 3.3, the gap becomes larger when there are more candidate series left.

Comparing the bounds in Eq. (6) **and Eq.** (8) Table I shows the performance of pruning by comparing our new bounds in Eq. (8) and the bounds derived in LEEWAVE in Eq. (6). The parameters in this experiment are d_{avg} in kFN, random reference set, T=1024, k=10, m=150, $|Q|$=10. There is slight improvement of the new bounds as it is slightly tighter and have better chance to prune more candidates, though the improvement is minor.

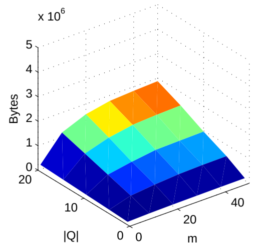


(a) Bandwidth consumption under different $|Q|$ and m

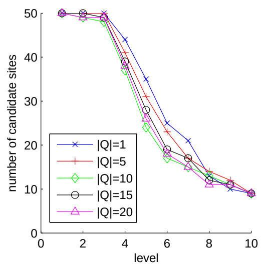


(b) Number of candidate sites in each level, m=50

Fig. 8. Results of kNN queries with d_{avg} given random reference set, T=1024, k=10



(a) Bandwidth consumption under different $|Q|$ and m



(b) Number of candidate sites in each level, m=50

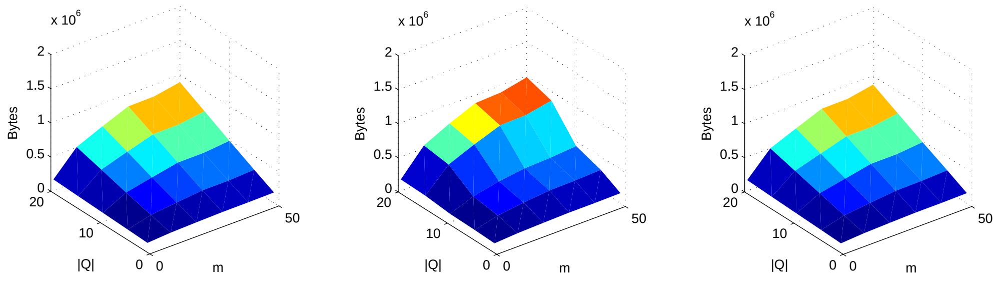
Fig. 9. Results of kNN queries with d_{avg} given analogous reference sets, T=1024, k=10.

C. Discussion

In this section, we discuss the sensitivity analysis in MSWAVE-L as it is the best models among all. We discuss how the transmission cost of MSWAVE-L would be influenced by variables such as the size of reference set, the type of reference set, and the distance metrics.

TABLE I. NEW BOUND (EQ. (6)) VS OLD BOUND (EQ. (8)) , T=1024, k=10, m=150, $|Q|$=10

step	1	2	3	4	5	6	7	8	9	10	11
new bound	150.0	150.0	148.9	137.3	53.9	32.5	24.8	19.8	15.2	11.5	8.0
old bound	150.0	150.0	150.0	139.1	62.4	33.7	26.0	20.3	15.6	11.5	8.0



(a) d_{avg} (b) d_{sin} (c) d_{com}

Fig. 10. Results for kFN queries with different linkage distances given analogous reference sets, T=1024, k=10

Random vs. analogous reference sets vs. Q: First, we examine the influence of the size of the reference set $|Q|$ on the transmission cost for kNN given $d_{avg}(Q, S_x)$ for the random reference set. In this experiment, we fixed k=10, T=1024 while $|Q|$ was varied from 1 to 20 and m was varied from 1 to 50. As is shown in Fig. 8(a) and Fig. 9(a), we can see that the ~~transmission cost~~ ~~almost~~ increased as $|Q|$ increased. From Fig. 8(b), we can see that when $|Q|$=10, the performance of pruning was better than others except $|Q|$=1. It is because for the set of randomly included references, at that moment (i.e., $|Q|$ increases to 10) certain reference time series that is close to many candidates was included in the set, and as a results, allows the system to quickly prune lots of candidates.

Fig. 9(b) also reveals that for analogous reference sets, MsWave-L does a desirable job as the bandwidth costs does not increase with $|Q|$ too much, which matches our discussion in the previous section.

Finally, we examined the influence of the different distance metrics. We compare three distance measures using analogous reference set to do kFN queries when k=10, T=1024, $|Q|$ was varied from 1 to 20 and m was varied from 1 to 50. As are shown in Fig. 10, we can see very little difference in terms of bandwidth consumption. The reason is that when the references in the reference set were very similar to each other, the distances between an arbitrary candidate series and each reference series shall be very close. Thus, their d_{avg}, d_{sin}, and d_{com} should be similar. As a result, the bounds obtained in every round are similar, and so are the final results. On the other hand, for random reference sets, we found no consistent pattern among the three distance metrics. The transmission cost indeed depends on the reference chosen in the set.

D. Model Comparison on Large-scale Synthetic Data

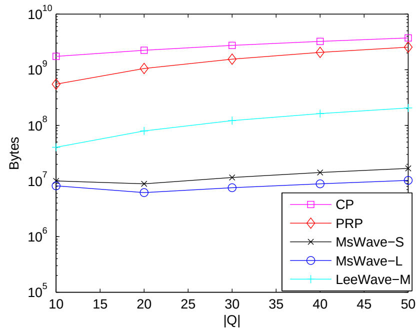
Before the end of the experiment section, we would like to show the performance of different methods on the large-scale synthetic data where the total number and the length of time series and the number of sites are all large.

Fig. 11(a) shows the bandwidth consumption of all methods using a logarithmic scale when T =12,500, m=500, and k=30 with the random reference set. We can see that both MsWave-L and MsWave-S saved significant bandwidth usage, about 1 to 2 orders of magnitude, compared to CP, PRP, and LeeWave-M. The saving here is much bigger than that on the real temperature data. This shows the superiority of MsWave-S and MsWave-L on a larger-scale data. In addition, we also observed the gap between MsWave-L and MsWave-S also increased as $|Q|$ increased, which was the same as our previous discussion on the real data set. Fig. 11(b) shows the performance of candidate site pruning of MsWave-L. The significant drop of candidate site number when only the coefficients of the top-half levels were passed was the main reason of its siginificant bandwidth saving. We also show the advantage of MsWave-L when m is big and much greater than k.

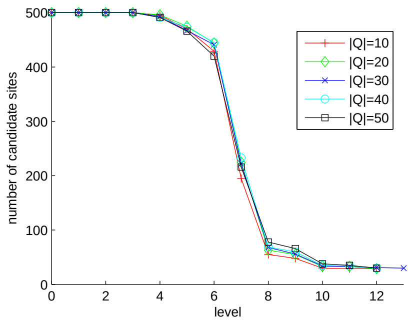
V. Related Work

Similarity search in time series databases, as an important feature in many applications, has drawn wide attention in recent decades. Among them, searching k-nearest neighbors is a well-studied topic in both fixed and streaming time series environment, such as the work in [11], [9], [12], [13], [3]. Given an error bound, Koudas et al. [12] approximated k-nearest neighbors search among stream snapshots. Liu et al. [13] proposed a new indexing technique based on scalar quantization to provide efficient nearest-neighbor search among multiple streams. Hung and Chen [9] provided an efficient approach to finding the k-nearest neighbors under an arbitrary range constraint based on the Haar wavelet synopses. Kashyap et al. [3] proposed an scalable kNN search method for vertically stored time series. Based on a multi-resolution transform on time series, the kNN search can be done by progressively pruning candidates efficiently in a stepwise sequential-scan manner. With different indexing and approximation methods, the general goal is to get the kNN as efficient as possible from a huge amount of time series. All these works assume that streams are collected and processed at a central site.

There are number of works studying issues for time series stored in a ditributed environment such as aggregation queries , burst detection, and frequent pattern mining with privacy



(a) Bandwidth consumption of all models



(b) Number of candidate sites in each level of MsWave-L.

Fig. 11. Experiments on synthetic data with random reference set, T=12500, k=30, m=500, d_{com}, kFN.

preserving [14], [15], [16]. However, only limited work discuss similarity search or nearest neighbor search for ditributed time series. For example, the work in [1] analyzed four schemes to tackle the k nearest neighbor queries. Later, the work in [2] considered to do the same k-nearest neighrbor queires on distributed streaming environment. Both works considered only single time series as the reference for queries. Also, they did not address the k-farthest neighbor query problem.

To the best of our knowledge, we are the first to deal with both kNN and kFN queries for a reference set of multiple time series in a distributed environment.

VI. Conclusions

Many believe distributed computation will become inevitable in big data era not just because we need to speed up the processing but also due to the fact that data are likely to be stored distributedly. Time series data are of no difference as they are generating in a very fast speed, usually from sensors in a Machine-to-Machine environment. As have been reported by McKinsey in a white paper about big data, a Boeing 737 generates 240 terabytes of data during a single cross-country flight. Such huge amount of time series data have to be stored in a distributed database. To efficiently handle ad hoc queries to the database, or to even design a search engine in such a distributed environment that to concurrently process large amount of complex queries while still guarantee the quality of the results without consuming too much bandwidth and transmission cost, MsWave seems to be a reasonable framework to be considered.

Technically speaking, compared with centralized nearest neighbor search for time series, distributed time-series matching has been addressed by much less literature, let along the consideration of more complex query patterns such as multiple time series. That says, although this paper advances the state-of-the-art by introducing the multiple-series query, we believe there are still many unresolved issues in this direction for future work. For instance, we would like to investigate how to improve the response time of such query; how the proposed distributed time series matching mechanism can be extended to perform time-series pattern discovery and supervised/semi-supervised learning in a distributed environment; how MsWave can be extended to other types of distances such as dynamic time warping; how other type of complex queries such as "finding instances similar to at least k reference instances" can be resolved, etc.

References

[1] A. N. Papadopoulos and Y. Manolopoulos, "Distributed processing of similarity queries," *Distributed and Parallel Databases*, vol. 9, 2001.

[2] M.-Y. Yeh, K.-L. Wu, P. S. Yu, and M.-S. Chen, "LeeWave: level-wise distribution of wavelet coefficients for processing $k$nn queries over distributed streams," *Proc. VLDB Endow.*, vol. 1, no. 1, 2008.

[3] S. Kashyap and P. Karras, "Scalable knn search on vertically stored time series," in *Proc. of ACM SIGKDD*, 2011.

[4] A. Haar, "Zur theorie der orthogonalen funktionensysteme," *Mathematische Annalen*, vol. 69, 1910.

[5] A. Bulut and A. K. Singh, "SWAT: Hierarchical stream summarization in large networks," in *Proc. of IEEE ICDE*, 2003.

[6] W.-G. Teng, M.-S. Chen, and P. S. Yu, "Resource-aware mining with variable granularities in data streams," in *Proc. of SIAM Data Mining*, 2004.

[7] Y. Zhu and D. Shasha, "Efficient elastic burst detection in data streams," in *Proc. of ACM SIGKDD*, 2003.

[8] Y. Matias, J. S. Vitter, and M. Wang, "Wavelet-based histograms for selectivity estimation," in *Proc. of ACM SIGMOD*, 1998.

[9] H.-P. Hung and M.-S. Chen, "Efficient range-constrained similarity search on wavelet synopses over multiple streams," in *Proc. of ACM CIKM*, 2006.

[10] T. Rakthanmanon, B. Campana, A. Mueen, G. Batista, B. Westover, Q. Zhu, J. Zakaria, and E. Keogh, "Searching and mining trillions of time series subsequences under dynamic time warping," in *Proc. of ACM SIGKDD*, ser. KDD '12. New York, NY, USA: ACM, 2012, pp. 262–270. [Online]. Available: http://doi.acm.org/10.1145/2339530.2339576

[11] L. Gao, Z. Yao, and X. S. Wang, "Evaluating continuous nearest neighbor queries for streaming time series via pre-fetching," in *Proc. of ACM CIKM*, 2002.

[12] N. Koudas, B. C. Ooi, K.-L. Tan, and R. Zhang, "Approximate NN queries on streams with guaranteed error/performance bounds," in *Proc. of VLDB*, 2004.

[13] X. Liu and H. Ferhatosmanoglu, "Efficient k-NN search on streaming data series," in *Proc. of SSTD*, 2003.

[14] V. Rastogi and S. Nath, "Differentially private aggregation of distributed time-series with transformation and encryption," in *Proc. of ACM SIGMOD*, 2010.

[15] L. Singh and M. Sayal, "Privacy preserving burst detection of distributed time series data using linear transforms," in *Proc. of IEEE CIDM*, 2007.

[16] J. C. da Silva and M. Klusch, "Privacy preserving pattern discovery in distributed time series," in *Proc. of IEEE ICDE Workshop*, 2007.