Communication-Efficient Distributed Multiple Reference Pattern Matching for M2M Systems

Jui-Pin Wang[†] Yu-Chen Lu[†] Mi-Yen Yeh[‡] Shou-De Lin[†] Phillip B. Gibbons[§]

[†]Department of Computer Science and Information Engineering, National Taiwan University, Taiwan
[‡]Institute of Information Science, Academia Sinica, Taiwan
[§]Intel Labs Pittsburgh, USA

Email: r01922165@ntu.edu.tw, b98902105@ntu.edu.tw, miyen@iis.sinica.edu.tw,
sdlin@csie.ntu.edu.tw, phillip.b.gibbons@intel.com

***Abstract*—In M2M applications, it is very common to encounter the ad hoc snapshot query that requires fast responses from many local machines in which all the data are distributed. In the scenario when the query is more complex, the communication cost for sending it to all the local machines for processing can be very high. This paper aims to address this issue. Given a reference set of multiple and large-size patterns, we propose an approach to identifying its k nearest and farthest neighbors globally across all the local machines. By decomposing the reference patterns into a multi-resolution representation and using 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 for large numbers of machines and large-sized reference patterns.**

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 subtly, 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).

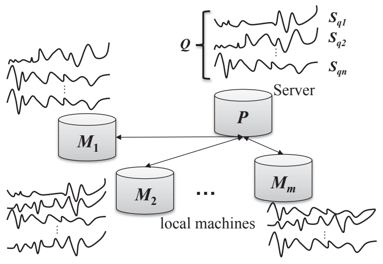


Fig. 1. The system model

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 communication, it is usually critical to design an approach that requires as little communication between machines 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 paper 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 sometimes be required to assemble the query at the server such as in the 3^{rd} scenario. Also, while we do not explicitly model response time, our solutions are highly parallel and fast.

Prior work has considered kNN search for the system model in Fig. 1, but restricted to a single reference pattern (i.e., $|Q| = 1$) [1], [2]. In a naive solution the server sends the query to each local machine, each local machine computes the kNN to the query from among the locally maintained measurement readings and sends them back to the server, and finally the server determines the overall kNN solution from among the results received. This solution, called *Concurrent Processing (CP)* [1], incurs a high bandwidth cost because each of the m machines sends back k results, out of which only k are in the overall solution. To address this, Papadopoulos and Manolopoulos [1] proposed the *Probabilistic Processing method (PRP)*, which reduces the amount of data required to be transmitted back to the server from the local machines. To further reduce bandwidth consumption, our earlier work [2] proposed LEEWAVE, which leverages the multi-resolution property of the Haar wavelet transformation of time series. However, none of this prior work considered multiple reference patterns or kFN search, and straightforward generalizations of LEEWAVE produce incorrect results (as we will show in Section II-C).

Our framework, called MSWAVE, is designed based on the following insights. First, to handle multiple reference patterns as queries, we propose three distance measurements, 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 are analogous to the single-link, average-link, and complete-link clustering models. Second, based on the characteristics of the Haar wavelet transform, MSWAVE pre-processes each series of measurement readings by decomposing it into a multi-resolution representation. Instead of sending the whole query set Q to the local machines, the server iteratively sends information on each query in Q in a level-wise manner starting with the coarsest resolution (fewest bytes sent) and continuing with increasingly finer resolutions (more bytes sent). We further derive and maintain certain similarity range bounds for each of the three distance measurements, such that the upper and lower bounds can be incrementally updated at each iteration/level. More importantly, we prove that these similarity range narrow as we move from one wavelet coefficients level to the next, enabling effective pruning of the candidate time series that reduces bandwidth consumption without causing any false dismissal. Although prior work has proposed wavelet level-wise pruning strategies [2], [3], we not only generalize it to multiple reference patterns but also further reduce bandwidth by shifting the similarity bounds calculations from the server to the local machines.

We 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 setups for searching both kNN and kFN.

Our main contributions can be summarized as follows:

- We present MSWAVE, a general communication-efficient framework to identify both kNN and kFN instances given multiple time series reference patterns in a distributed environment. To our knowledge, this is the first solution proposed for such purpose.
- Methodology-wise, we propose to use average, closest, and furthest neighbor distance to process multiple query (dis)similarity. We then take advantage of the multiple-resolution property of wavelet coefficients, and then for each distance measurement we derive upper and lower bounds of the similarity between each candidate time series to the query set. Such bounds can be exploited to prune candidates for more efficient search without compromising correctness. Moreover, in further contrast to prior approaches, we propose to shift the bounds computation from the server to the local machines to further reduce the bandwidth consumption.
- 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 bandwidth savings from shifting the bounds computations to the local machines. Finally, we conduct extensive experiments that demonstrate MSWAVE's significant 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.

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: 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,

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

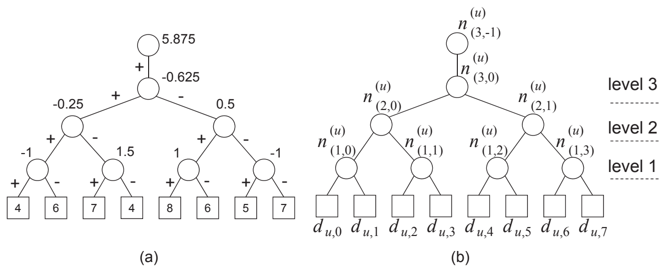


Fig. 2. (a) Error tree for a time series {4,6,7,4,8,6,5,7} (b) Error tree notation

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 [2], [3], [5]. In Haar wavelet decomposition, each time series is decomposed into multiple resolutions and can be represented using an error tree structure [6], 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 calculated directly from the coefficients themselves, in a top-down level-wise manner as suggested in LEEWAVE [2]:

$$\begin{aligned} Dst(S_{ref}, S_x) &= \sum_{l=1}^{L} Dst^l(S_{ref}, S_x) \\ &= accDst^{\ell}(S_{ref}, S_x) + \sum_{l=1}^{\ell-1} Dst^l(S_{ref}, S_x), \end{aligned} \tag{2}$$

where

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

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

$$\begin{aligned} & accDst^{\ell}(S_{ref}, S_x) \le Dst(S_{ref}, S_x) \le \\ & \quad accDst^{\ell}(S_{ref}, S_x) + \sum_{l=1}^{\ell-1} \sum_p ([n^{(ref)}_{(l,p)}]^2 + [n^{(x)}_{(l,p)}]^2) \times 2^l \\ & \quad + 2 \times \sqrt{\sum_{l=1}^{\ell-1} \sum_p [n^{(ref)}_{(l,p)} \times 2^l]^2 \times \sum_{l=1}^{\ell-1} \sum_p [n^{(x)}_{(l,p)}]^2}. \end{aligned} \tag{3}$$

[1]Note that when the length of a time series 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 distinct powers of two. In such cases, we still send coefficients from different sub-trees in a level-wise manner.

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

Note that the server can compute the lower and upper bounds in Eq. (3) from its $n^{(ref)}_{(l,p)}$ terms and three summation terms provided by a local machine. This saves considerable bandwidth compared to the local machine sending its complete time series. Also, our earlier work [2] proved 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 $Q = \{S_{q1}, \dots, S_{qn}\}$ 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 $Q = \{S_{q1}, \dots, S_{qn}\}$ are defined as:

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

These definitions are intended to be analogous to the single-link, average-link, and complete-link distances used in clustering. Intuitively, a time series S_x is considered close to a group of time series Q if either there exists one time series in Q that is very similar to S_x (i.e., d_{sin}), or most of the time series in Q are close enough to S_x to make their average similar to S_x (i.e., d_{avg}), or the most dissimilar time series in Q is still similar to S_x (i.e., d_{com}).

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

Definition 2: Given a server P with a reference time series set $Q = \{S_{q1}, \dots, S_{qn}\}$, each of length T, and a set of distributed local machines $M_1, \dots, M_m$, 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, considering the 6 cases (3 linkage distance measurements for 2 kinds of queries), LeeWave-M can guarantee correct solutions for only 2 of the 6, namely, for d_{sin} in kNN and d_{com} in kFN. To see this, consider the following simple counterexample 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 machines, 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 counterexamples for kFN under d_{sin} and d_{avg}.

Besides the limitation of being able to solve only 2 out of the 6 cases, 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 machine based on sum terms sent by the local machines; 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 an 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. (3), we can derive the upper bound UB and the lower bound LB of $Dst(S_{qi}, S_x)$ as soon as all the coefficients for S_{qi} from the highest level L to the current level ℓ have been sent to the local machines, as follows:

$$LB(qi,x) = accDst^{\ell}(S_{qi}, S_x). \tag{4}$$

$$\begin{aligned} UB(qi,x) &= accDst^{\ell}(S_{qi}, S_x) \\ &+ \sum_{l=1}^{\ell-1}\sum_{p}([n_{(l,p)}^{(qi)}]^2 + [n_{(l,p)}^{(x)}]^2) \times 2^l \\ &+ 2 \times \min\{\sqrt{\sum_{l=1}^{\ell-1}\sum_{p}[n_{(l,p)}^{(qi)} \times 2^l]^2 \times \sum_{l=1}^{\ell-1}\sum_{p}[n_{(l,p)}^{(x)}]^2}, \\ &\sqrt{\sum_{l=1}^{\ell-1}\sum_{p}[n_{(l,p)}^{(x)} \times 2^l]^2 \times \sum_{l=1}^{\ell-1}\sum_{p}[n_{(l,p)}^{(qi)}]^2}\}. \end{aligned} \tag{5}$$

Note that Eq. (5) is an enhanced version of the upper bound compared to that in Eq. (3). Because the roles of S_{qi} and S_x are interchangeable in the squared terms in Eq. (3), a tighter upper bound is obtained by choosing the minimum among the two choices. Our experiments will show that this subtle change noticeably 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 $Dst(S_{qi}, S_x)$ for all $S_{qi} \in Q$, we note that because the distance is non-negative, we can simply derive the new bounds as follows.

$$LB_{avg}(Q, S_x) = \frac{1}{|Q|}\sum_{i=1}^{|Q|} LB(qi, x) \tag{6}$$

$$UB_{avg}(Q, S_x) = \frac{1}{|Q|}\sum_{i=1}^{|Q|} UB(qi, x) \tag{7}$$

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

$$LB_{sin}(Q, S_x) = \min_{1 \le i \le |Q|} LB(qi, x) \tag{8}$$

$$UB_{sin}(Q, S_x) = \min_{1 \le i \le |Q|} UB(qi, x) \tag{9}$$

Finally, for $d_{com}(Q, S_x)$, the two bounds are:

$$LB_{com}(Q, S_x) = \max_{1 \le i \le |Q|} LB(qi, x) \tag{10}$$

$$UB_{com}(Q, S_x) = \max_{1 \le i \le |Q|} UB(qi, x) \tag{11}$$

Fig. 3(a) illustrates the bounds for $d_{sin}(Q, S_x)$. 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 $d_{sin}(Q,)$ 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).

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

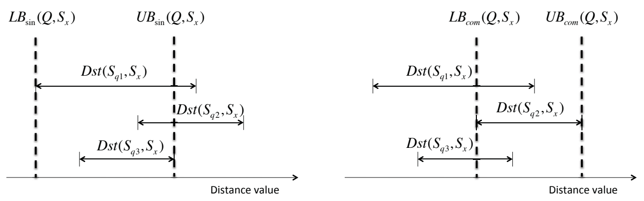
The server P:	*A local machine M_i:*
1. Send coefficients of each $S_{qi} \in Q$ at level L to all M local machines.	2. For each local candidate time series, S_x, compute and return $(Dst^L(S_{qi}, S_x)\ \ \forall S_{qi} \in Q$, $\sum_{l=1}^{L-1}\sum_p [n_{(l,p)}^{(x)}]^2$, $\sum_{l=1}^{L-1}\sum_p ([n_{(l,p)}^{(x)}]^2 \times 2^l)$, $\sum_{l=1}^{L-1}\sum_p ([n_{(l,p)}^{(x)}]^2 \times 2^l)^2)$ to P.
3. Compute the upper and lower bounds based on Eq. (4)-(5) 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. (6)-(11). 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 ids of any pruned candidate series to the appropriate 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. (4)-(5) for each candidate time series to each reference series. Then update the bounds of the linkage distance based on Eq. (6)-(11). Do corresponding pruning for kNN or kFN. Set **done** to **true** if there are k candidate time series left.	
8. }	
9. Ask the appropriate machines for the contents of the final k time series.	10. If asked, send back the corresponding full time series 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 measure (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. (4)-(5). Then compute and return the two linkage distances bounds based on Eq. (6)-(11).
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 ids of any pruned candidate series to the appropriate local machines.	6. Update the corresponding upper/lower bounds based on the computation defined in Eq. (6)-(11) 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 k candidate time series left.	
8. }	
9. Ask the appropriate machines for the contents of the final k time series.	10. If asked, send back the corresponding full time series contents.

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



(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 and lower bound computations for different linkage distance measures.

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 ℓ to level $\ell - 1$

Proof: As argued above, it readily follows from [2] that Eq. (4) is non-decreasing and Eq. (5) 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 ℓ to level $\ell - 1$

Proof: Let $l(\ell)$ and $u(\ell)$ 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 ℓ. That is,

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

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

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

where the first inequality holds because $l(\ell)$ is the arg min at level ℓ and the second inequality holds because Eq. (4) is non-decreasing. Thus, $LB_{sin}(Q, S_x)$ is non-decreasing.

Similarly, because Eq. (5) 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 ℓ to level $\ell - 1$.

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 the server P progressively prunes the candidates. We will present two schemes to solve this problem: MsWave-S and MsWave-L.

MsWave-S: Server computes the 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. (4) and Eq. (5) for each candidate time series.

It then does 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 non-increasing from level to level, we can guarantee that there are no false dismissals 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, P moves to the next level.

For any given level l, P sends the level-l coefficients of each $S_{qi} \in Q$ and the ids of any pruned time series to the appropriate local machines. The local machine returns two level-specific numbers for each (remaining) candidate time series: $Dst^l(S_{qi}, S_x)$ for $i = 1$ to $|Q|$ and $\sum_p [n_{(l,p)}^{(x)}]^2$. P then uses these values to update the upper/lower bounds, always making them tighter. With the bounds of each candidate series to each reference time series, P further computes the similarity range under the prespecified linkage distance of each series to the reference set Q based on Eq. (6)–(11). With increasingly tighter ranges, P can better prune the candidate list. The algorithm ends when there are k candidate time series remaining.

MsWave-L: Local machines compute the bounds. Note that with MsWave-S, the local machines consume bandwidth to send back the level distances of each time series 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 under 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, reducing bandwidth.

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 enable the local machines 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. (6)–(11) according to different linkage distance measures. 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 results and can be discarded. This procedure proceeds iteratively until the final results are produced. Note that the pruning strategy is the same as that done in MsWave-S.

C. *Analysis of Bandwidth Consumption*

We will now analyze the bandwidth consumption of MsWave-L relative to MsWave-S. Suppose there are a total of s time series distributed in m local machines, and $|Q|$ reference time series. Let s_ℓ be the number of candidate time series remaining at level ℓ. We have that the difference in bandwidth between MsWave-S and MsWave-L is: $-3|Q|m + s(4|Q| - 2) + \sum_\ell 2s_\ell(|Q| - 1)$, which equals

$$|Q|(4s - 3m + 2\sum_\ell s_\ell) - 2s - 2\sum_\ell s_\ell. \qquad (12)$$

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.

Note that Eq. (12) is always greater than zero. Because 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. Moreover, MsWave-L's bandwidth savings over MsWave-S increases linearly with $|Q|$.

IV. Experiments

The goal for this section is twofold. First, we would like to compare the bandwidth consumption of MsWave with some baseline and state-of-the-art approaches. 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 the University of Dayton. The data from each city is considered as a time series with 2048 data points. For the synthetic data set, we use the same random walk data model used in [7]. Each time series is generated by the random walk whose every step size is a normal distributed random number with mean=0 and standard deviation=1. There are 12,500 time series of length 12,500 generated. After selecting $|Q|$ time series as the reference set, the candidate time series are chosen from the remaining time series and are equally distributed to the m machines.

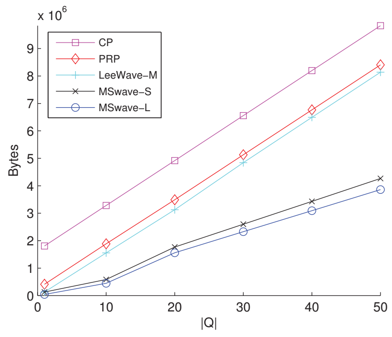
We consider five frameworks: (i) CP, which is the Concurrent Processing baseline [1] described in Section I, but generalized to $|Q| > 1$ by sending the whole query set at the beginning; (ii) PRP, which is the Probabilistic Processing method [1] mentioned in Section I, but again generalized to $|Q| > 1$ in a straightforward manner; (iii) LEEWAVE-M, as discussed in Section II-C; (iv) MSWAVE-S (Fig. 4); and (v) MSWAVE-L (Fig. 5).

We compare the total bandwidth cost for these five frameworks in a distributed environment simulated in MATLAB. We study the influences on the bandwidth cost of the size of the reference set $|Q|$, the time series length T, the number of machines m, and the k for kNN/kFN. The total bandwidth cost is the summation of all data transmitted between the server and other local machines. Note that in this simulation framework, practical issues such as the overheads of message headers, packet losses, retransmissions, etc. were not considered.

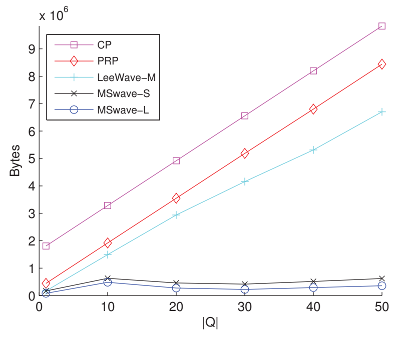
There are two strategies we employ to choose the time series that comprise the instances in a query set Q. For the *analogous reference set*, we choose one time series randomly and then choose its closest $|Q|$-1 neighbors to form Q; thus the queries in Q are highly similar. For the *random reference set*), we choose $|Q|$ time series at random; thus the queries in Q are likely to be dissimilar.

B. Comparison on Real Data

Comparing all five frameworks. Fig. 6 shows a comparison of the total bandwidth consumption of all five frameworks, for both the random reference set (Fig. 6(a)) and the analogous reference set (Fig. 6(b)). Both plots show that the MSWAVE frameworks outperform the other frameworks significantly. We also find the slopes of MSWAVE frameworks are less steep than the slopes of the others, highlighting the benefits of using Eq. (6)−(11) for pruning. For analogous reference sets, we even find that the bandwidth costs of the MSWAVE frameworks do not increase significantly when $|Q|$ increases. This can be explained as follows. When the reference time series are similar, the distance to any one of the reference time series is 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 levels of the protocol, we already obtain enough information to estimate the true distance, which leads to more effective pruning. On the other hand, for the random reference set, because the distances to each reference vary 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 consuming more bandwidth.



(a) random reference set

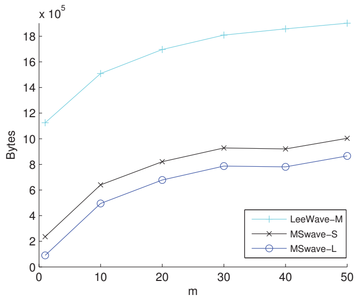


(b) analogous reference set

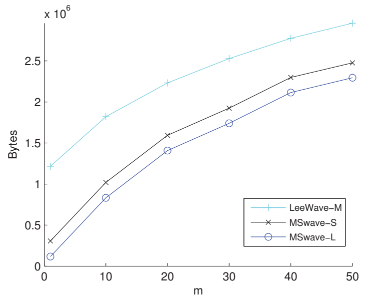
Fig. 6. Comparison between frameworks given two kinds of query sets, T=1024, k=10, m=20, d_{com}, and kFN

Comparing MSWAVE-L, MSWAVE-S and LEEWAVE-M. Next, Fig. 7 highlights the comparison of MSWAVE-S, MSWAVE-L, and LEEWAVE-M in total bandwidth consumption. 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)) because, as argued in Section 2.3, these are the only two feasible cases for LEEWAVE-M. The figure shows that the performance of MSWAVE-L is clearly better than MSWAVE-S, while both are much better than LEEWAVE-M in all configurations.

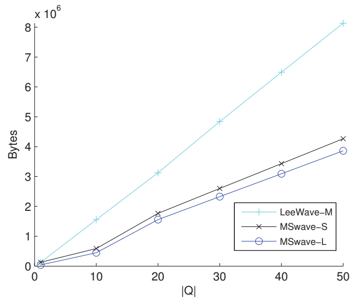
Figs. 7(a) and 7(b) present the results while varying the number of machines m from 1 to 50. The figures show that increasing m does not change the performance differences significantly. Figs. 7(c) and 7(d) study the performance impact of the size of reference set $|Q|$, which is varied from 1 to 50. The figures show that the MSWAVE-L's bandwidth savings over LEEWAVE-M increases as $|Q|$ increases. On the other hand, while MSWAVE-L's bandwidth savings over MSWAVE-S also increases as $|Q|$ increases, the increase is less significant. The savings increases because, the larger the reference set is, the more values MSWAVE-S has to send from the local machines to the server, as analyzed in Section III-C. Finally, Figs. 7(e) and 7(f) examine the impact of k on the performance, with k varying from 1 to 20. It is not hard to reason that the difference between LEEWAVE-M and MSWAVE-L



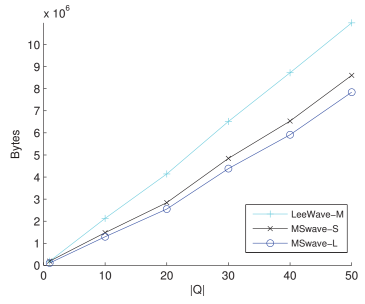
(a) k=10, $|Q|$=10, d_{com}, kFN



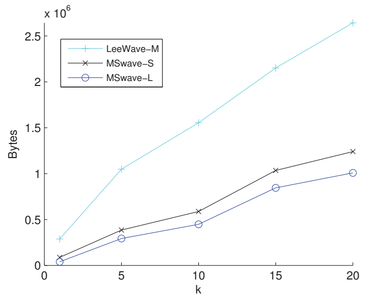
(b) k=10, $|Q|$=10, d_{sin}, kNN



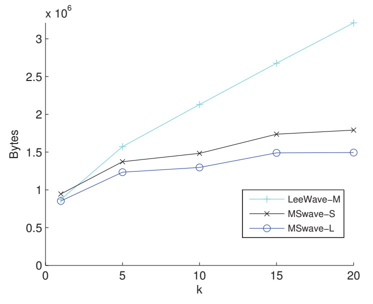
(c) k=10, m=20, d_{com}, kFN



(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 frameworks given the random reference set, T=1024.

in bandwidth consumption increases as k increases because LEEWAVE-M's overhead grows linearly with k. We can also see that the difference between MSWAVE-L and MSWAVE-S increases slightly as k becomes larger. The larger k is, the fewer candidate series are pruned early, and based on our discussion in Section III-C, the gap becomes larger when there are more candidate series still remaining.

Comparing the upper bounds in Eq. (3) **and Eq.** (5)**.** Table I compares our new upper bound in Eq. (5) to the upper bound derived in LEEWAVE in Eq. (3). The parameters in this experiment are d_{avg} in kFN, random reference set, T=1024, k=10, m=150, and $|Q|$=10. The new bounds are slightly lower, which makes them better for pruning, although the improvement is quite small.

C. Sensitivity Analysis of MSWAVE-*L on Real Data*

The previous results have shown that MSWAVE-L outperforms the other frameworks. To gain additional insights into its performance, we present a further sensitivity analysis for MSWAVE-L in this section.

Sensitivity to query set size and number of machines. First, we examine the impact of the size of the reference set $|Q|$ and the number of machines m on the bandwidth consumption for finding kNN under d_{avg}. In this experiment, we fix k=10 and T=1024, while $|Q|$ is varied from 1 to 20 and m is varied from 1 to 50, and we consider both the random and analogous reference sets. Figs. 8(a) (random) and 8(c) (analogous) show that the bandwidth consumption generally increases as $|Q|$ increases. An exception occurs for the random reference sets, where the bandwidth consumption of $|Q| = 10$ is smaller than $|Q|$=5. By looking into Fig. 8(b) we find that the pruning power of $|Q|$=10 was also better than that of $|Q| = 5$. This is because when the size of the reference set increases from $|Q|$=5 to $|Q|$=10, certain reference time series that are close to many candidates were included in the set, which enables MSWAVE-L to quickly prune many candidates. A similar effect can be seen in Fig. 8(d) for the analogous reference sets. In this case, the bandwidth consumption of MSWAVE-L does not increase too much with the growth of $|Q|$, which matches our discussion in the previous section.

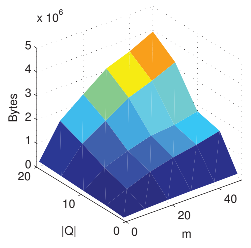
Sensitivity to distance measure. Finally, we examine the influence of the different distance measures. We compare the three distance measures using analogous reference sets to do kFN queries when T=1024, $k = 10$, $|Q|$ is varied from 1 to 20 and m is varied from 1 to 50. As shown in Fig. 9, we can see little difference between the measures in terms of bandwidth consumption. The reason is that when the patterns in the reference set are very similar to each other, the distances between an arbitrary candidate series and each reference series is 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 find no consistent patterns among the three distance measures. The bandwidth consumption indeed depends on how the reference set is chosen.

D. Comparison on Large-scale Synthetic Data

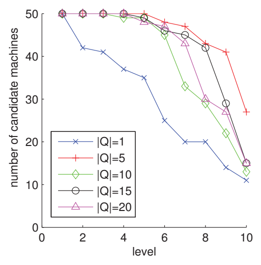
We conclude our experimental study with a comparison of the five frameworks on the large-scale synthetic data set discussed in Section IV-A. The purpose is to compare their bandwidth consumption when the number of time series and the length of each time series are much larger, namely, both are increased to 12,500. Recall that both MSWAVE and LEEWAVE-M can work when the time series pattern length is not a power of 2. We also increase the total number of machines to 500.

TABLE I. NEW BOUND (EQ. (5)) VS OLD BOUND (EQ. (3)) , T=1024, k=10, m=150, $|Q|$=10

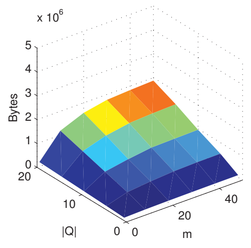
step	1	2	3	4	5	6	7	8	9	10	11
new bound	150.0	150.0	148.9	137.3	53.9	32.5	24.8	19.8	15.2	11.5	8.0
old bound	150.0	150.0	150.0	139.1	62.4	33.7	26.0	20.3	15.6	11.5	8.0



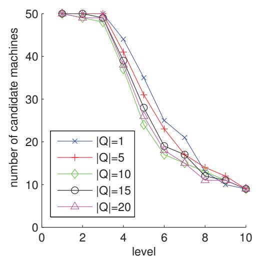
(a) Bandwidth consumption under different $|Q|$ and m



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

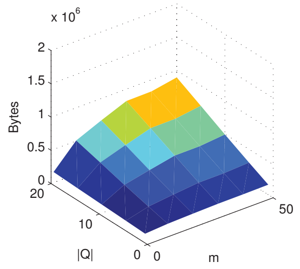


(c) Bandwidth consumption under different $|Q|$ and m

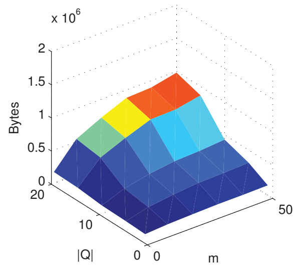


(d) Number of candidate machines in each level, m=50

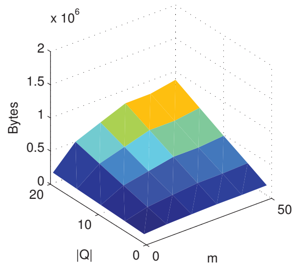
Fig. 8. Results of kNN queries using MSWAVE-L with d_{avg} for either random reference sets ((a),(b)) or analogous reference sets ((c),(d)), T=1024, k=10



(a) d_{avg}



(b) d_{sin}



(c) d_{com}

Fig. 9. Results for kFN queries with different linkage distances given analogous reference sets, T=1024, k=10

Fig. 10(a) shows the bandwidth consumption of all frameworks using a logarithmic scale under the parameter settings T =12,500, k = 30, m = 500, d_{com}, kFN, and the random reference set, varying $|Q|$ from 10 to 50. The bandwidth savings for MSWAVE-L and MSWAVE-S are even more dramatic, about 1 to 2 orders of magnitude, compared to CP, PRP, and LEEWAVE-M. MSWAVE's advantage is fairly consistent across the range of $|Q|$. In addition, we also observe the gap between MSWAVE-L and MSWAVE-S increases as $|Q|$ increases, again agreeing with the analysis in Eq. (12).

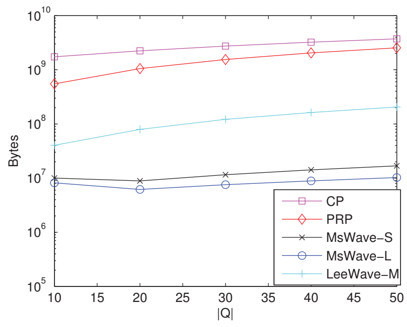
Finally, Fig. 10(b) focuses in on MSWAVE-L, the best framework, and shows the effectiveness of its candidate pruning at each level for the large-scale data set. The parameter settings are the same as the prior experiment. The significant drop in the number of candidate machines when only the coefficients of the top-half levels are passed is the main reason for its significant bandwidth savings. The results in this section demonstrate the added advantage of MSWAVE-L when m is large and much greater than k.

V. RELATED WORK

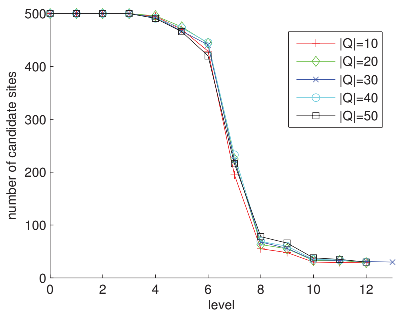
Similarity search in time series databases has drawn wide attention in recent years, due to its importance in many applications. In particular, k-nearest neighbor search is a well-studied topic in both fixed and streaming time series environments, such as the work in [8], [9], [10], [3]. Given an error bound, Koudas *et al.* [8] approximated k-nearest neighbors search among stream snapshots. Liu *et al.* [9] proposed a new indexing technique based on scalar quantization to provide efficient nearest-neighbor search among multiple streams. Hung and Chen [10] 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 a 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 efficiently as possible from a large number of candidate time series. All these works assume that streams are collected and processed at a central site.

There are a number of works studying issues for time series stored in a distributed environment, such as aggregation queries, burst detection, and frequent pattern mining while preserving privacy [11], [12], [13]. However, only a few works discuss similarity search or nearest neighbor search for distributed time series. For example, Papadopolous and Manolopoulos [1] analyzed four schemes to tackle kNN queries, and our earlier work proposed LEEWAVE [2]. Both studies considered only single time series as the reference for queries and did not address the kFN query problem.

Other work has studied queries containing multiple instances, but in different settings. For example, in multiple pattern matching in text search or bioinformatics applica-



(a) Bandwidth consumption of all frameworks



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

Fig. 10. Experiments on synthetic data with random reference set, T=12500, k=30, m=500, d_{com}, kFN.

tions [14], [15], the inputs are assumed to be multiple strings and the algorithms report all occurrences of the input strings. This is different from our goal of finding kNN/kFN in continuous time series while limiting bandwidth consumption. Meanwhile, the typical assumption of centralization in these settings aims to speed up the processing, in contrast to our setting where data are assumed to be coming from distributed 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.

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

VI. Conclusions and Future Work

Distributed computation is generally believed to be a reasonable and inevitable solution for M2M applications. It is not only because huge amounts of data such as sensor readings are being accumulated fast and distributedly, but also because of concerns of communication efficiency among thousands or more devices. MsWave provides the first framework for efficiently and correctly handling ad hoc kNN/kFN queries with multiple reference patterns over distributed time series data.

Technically speaking, compared with centralized nearest neighbor search for time series, distributed time-series matching has been studied by only a few prior works, none of which considered more complex query patterns such as multiple time series. Although this paper advances the state-of-the-art by introducing the multiple-series query, we believe there are still many unresolved issues to be explored. For example, we would like to investigate how to improve the response time of such queries, which is constrained by the current one-level-at-a-time approach; how to extend the proposed distributed time series matching mechanism to supervised/semi-supervised learning in a distributed environment; how to extend MsWave to other types of distance measures such as dynamic time warping; and how to resolve other types of complex queries such as "find instances similar to at least k reference instances." These and other open questions make for promising directions for future work.

Acknowledgment

This work was sponsored by AOARD grant number No. FA2386-13-1-4045 and National Science Council, National Taiwan University and Intel Corporation under Grants NSC100-2911-I-002-001, NSC101-2628-E-002-028-MY2, NSC101-2221-E-001-013 and NTU102R7501, and the Intel Science & Technology Center for Cloud Computing.

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

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

[7] 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*, 2012.

[8] 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.

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

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

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

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

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

[14] R. Kandhan, N. Teletia, and J. M. Patel, "Sigmatch: fast and scalable multi-pattern matching," *Proc. VLDB Endow.*, vol. 3, no. 1-2, 2010.

[15] S. Kim and Y. Kim, "A fast multiple string-pattern matching algorithm," in *Proc. of 17th AoM/IAoM Conference on Computer Science*, 1999.