# 基于动态时间规整的时序数据相似连接

周宁南 张 孝 刘城山 王 珊
(教育部数据工程与知识工程重点实验室(中国人民大学) 北京 100872)
(中国人民大学信息学院 北京 100872)

**摘 要** 由于蕴含事物发展规律,时序数据上的数据挖掘正成为大数据决策的重要组成部分.作为时序数据挖掘的一种基本操作,时序数据相似连接可以找出给定相似度度量下的所有相似时序数据对.研究表明,动态时间规整(Dynamic Time Warping,DTW)正在文本挖掘、趋势预测等越来越多的科学与社会应用领域中成为时序数据上目前最佳的相似性度量方法.该文首次提出采用 DTW 作为相似性度量方法的时序数据相似连接问题.特别地,该文首次提出了基于阈值和基于 Top-$k$ 的两种 DTW 度量上的时间序列相似连接任务.除了服务于进一步的时序数据挖掘算法,这两个任务还具有机器翻译、关联检测等广泛的直接应用.但是,直接的相似连接方法因为时序数据的规模大、DTW 计算复杂性高而不能在实际中工作.尽管存在很多基于 DTW 的索引和上下界计算方法,这些工作主要关注 DTW 度量上的快速检索而非相似连接.因此,这些方法都假设存在一个固定的时序数据作为查询,并根据查询使用时间和空间复杂度很高的方法构建索引或进行预计算.但在文中的相似连接问题中,所有时序数据都是查询,因此这些方法的构建索引和预计算的时间比直接的相似连接方法需要的处理时间还长.为此,该文针对两种相似连接任务提出了两个基于 DTW 上下界的剪裁框架用于减少准确 DTW 相似性的计算次数.基于划分,该文为 DTW 度量设计了新颖的上下界计算方案.由于细粒度的划分带来上下界接近准确的 DTW 相似性但需要更长的计算时间,而粗粒度的划分需要更短的计算时间和与准确 DTW 相似性有较大差距的上下界,该文设计了基于二分查找的机制来自动找到合适的划分粒度,实现了整体的高处理性能.面对单机不能容纳全部时序数据和运行时间长的情况,该文将提出的两种相似连接处理框架利用 MapReduce 并行计算框架扩展到了分布式环境.该文在两个真实数据集上验证了文中提出的 DTW 相似连接在实际应用中的效果,并在真实与合成数据集上进行了充分的实验,验证了文中方法的高效性.

**关键词** 动态时间规整;时序数据;相似连接;划分剪枝;分布剪枝
**中图法分类号** TP311 **DOI 号** 10.11897/SP.J.1016.2018.01798

# Similarity Join on Time Series under Dynamic Time Warping

ZHOU Ning-Nan ZHANG Xiao LIU Cheng-Shan WANG Shan
(*Key Laboratory of Data Engineering and Knowledge Engineering of the Ministry of Education (Renmin University of China), Beijing* 100872)
(*Department of Information, Renmin University of China, Beijing* 100872)

**Abstract** Revealing evolution insights of things, time series mining is becoming an indispensable component of big data driven decision making. As a fundamental operation in time series mining, given a similarity measure, similarity join gathers all pairs of similar time series. It is demonstrated that DTW (Dynamic Time Warping) has served as the best measure in disparate domains ranging from scientific to social fields such as text mining or tendency prediction. In this paper, we for the first time propose to join similar time series with DTW as the similarity measure. Specifically,

收稿日期:2016-12-13;在线出版日期:2017-04-19.本课题得到国家重点研发计划项目(2016YFB10007002)和国家自然科学基金重点项目(61432006)资助.**周宁南**,男,1990 年生,博士,主要研究方向为数据库、数据挖掘.E-mail:zhouningnan@gmail.com.**张 孝**(通信作者),男,1972 年生,博士,副教授,中国计算机学会(CCF)会员,主要研究方向为数据库、大数据.E-mail:zhangxiao@ruc.edu.cn.**刘城山**,男,1992 年生,硕士,主要研究方向为数据库.**王 珊**,女,1944 年生,教授,博士生导师,中国计算机学会(CCF)会士,主要研究领域为高性能数据库、知识工程、大数据.

we for the first time define two tasks, the threshold based and the Top-$k$ based similarity join under DTW. Besides to serve time series further mining tasks such as stock prediction, these two tasks can be directly applied to a wide spectrum of applications such as machine translation and delay-correlation detection. Unfortunately, trivial solutions suffer from the large scale nature of time series and high computational complexity of DTW. Numerous indexing techniques and various lower and upper bounds of DTW have been proposed. However, these works aim at similarity search rather than similarity join under DTW. In concrete, they assume that a fixed time series serves as a query and index or precomputation is performed on the query time series. It is time and space-consuming to construct index and precompute for the fixed time series. However, under our similarity join task, all time series serve as the fixed query and thus the index construction or precomputation time for all the time series is even beyond the execution time of the trivial solution and thus these techniques become impractical. To tame similarity join under DTW, we first propose two pruning based processing frameworks for the threshold-based and Top-$k$ based similarity join tasks respectively. These two frameworks prune unnecessary calculation of accurate DTW similarity between time series by leveraging the cheap upper and lower bound of DTW measure. In this way, we further devise novel upper and lower bounds for DTW measure. Both bounds are developed on top on time series partition. Since fine-grained partition enables more accurate DTW similarity but consumes more execution time while coarse-grained partition results in less accurate DTW similarity but consumes less execution time, we develop a mechanism based on binary search to quickly tune the granularities of partitions automatically and thus enable the overall practical performance. When single machine cannot meet the requirement of performance or cannot hold the massive time series, we extend our processing frameworks to distributed environment. Specifically, we design a MapReduce implementation to our pruning based similarity join framework. We conduct extensive experiments to demonstrate the effectiveness and efficiency of our methods. First, we apply the two proposed similarity join tasks on two real world datasets to demonstrate that the threshold-based similarity join task can be used to find correlated power supplement sources and find the same entities in different languages. Then, we use both real world and synthetic datasets to demonstrate that our methods outperform existing solutions consistently under various lengths and volume of time series.

**Keywords**　dynamic time warping; time series; similarity join; partition-based pruning; distribution-based pruning

## 1　引　言

随着大数据时代的到来，人们获得了诸如供电负荷走势、文本序列等社会和科学各领域的大规模时序数据. 例如，电网中各个供电设备的输出功率每15 min 由传感器记录一次，这样一个设备一年就积累长度超过 60/15 h×24 h/d×365 d≈35000 的时间序列. 由于时序数据蕴含了事物发展的规律或特点，将具有相似变化趋势的时序数据聚集在一起可以挖掘其中的相关性、主旨等规律[1-4]，进而为基于大数据的决策提供依据.

DTW(Dynamic Time Warping，动态时间规整)在时序数据的大部分应用领域中被证明是最佳的相似性度量方法[1,5]，其普适性得到超过 800 篇论文的验证，没有其它度量方法显著优于它[6,7]. DTW 在下面两个实际数据集上的应用说明了本文将要提出两类具体任务.

**例 1.**　图 1 是 3 个供电设备从 1 月 1 日到 1 月 14 日每天采样一次供电功率得到的供电功率变化曲线. 可以看出的规律是，设备 3 的曲线向右平移三天就能够和设备 1 的曲线基本重合. 这种时间上滞后的关联性可以被 DTW 捕捉到. 这样，给一个较小的阈值(比如 5)，设备 1 和设备 3 将成为 DTW 相似连接的结果. 利用此结果，电网公司可以通过比较某个时刻设备 1 和三天前设备 3 的供电功率判断是否

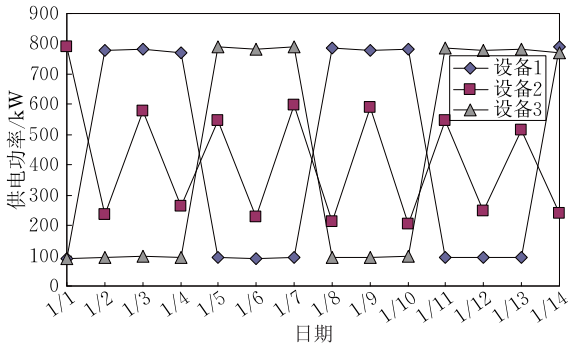
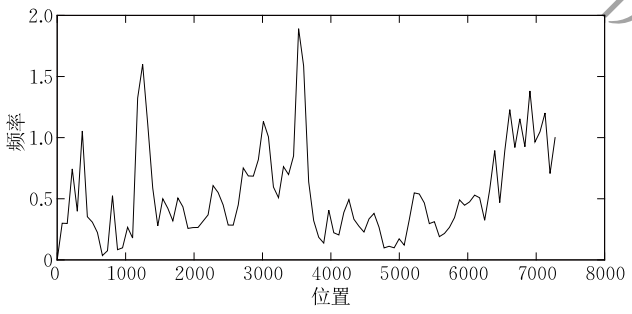


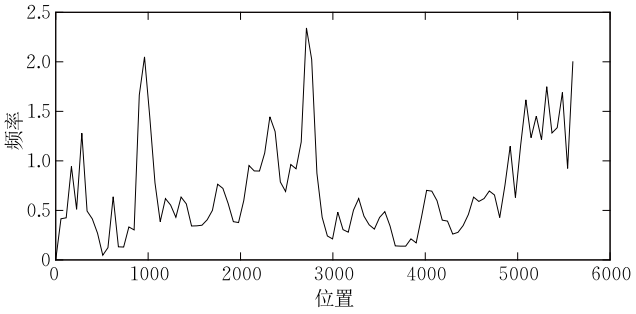

图 1 3 个供电设备 14 天内供电功率的时序数据

供电异常.

**例 2.** 图 2 是具有相同意义的词汇在不同语言的同一著作中出现频度的时序数据. 我们将著作以 100 个单词为单位切割成连续的片段,并记录给定单词在每个片段中出现的次数. 尽管著作的不同语言的版本具有不同的语序、词汇数,DTW 可以捕捉到图 2(b)与时间轴上被压缩了的图 2(a)十分相似. 如果我们将每个词汇与它们最相似(Top-1)的时间序列相连接,连接结果将显示每对对应不用语言中相同意义的词汇,这为利用统计的机器翻译中进行双语词汇匹配提供了帮助[8].




(a) 英文版“God”一词在每100个词汇中出现的频度, 横坐标表示给定单词在切割后的单词片段中的位置




(b) 中文版“上帝”一词在每100个词汇中出现的频度, 横坐标表示给定单词在切割后的单词片段中的位置

图 2 中英文词汇在同一著作中出现频度的时序数据

上面的两个例子体现了对时序数据进行基于阈值和 Top-$k$ 的相似连接在数据挖掘中具有重要应用. 总结这两类任务,本文首次提出了基于 DTW 的相似连接这样一个新的查询问题,并主要考虑基于阈值和基于 Top-$k$ 的 DTW 相似连接这两种具体任务.

计算 DTW 的时间复杂度较高,为 $O(n^2)$,其中 $n$ 是时序数据的长度,而我们面对的时序数据通常很长,通常超过 $10^4$,这为基于 DTW 的时序数据相似连接带来了极大的挑战. 现有的相似连接和 DTW 相似搜索两方面工作都具有不足.

相似连接问题通常采用“过滤-验证”框架[9,10],即过滤阶段将每个时序数据与和它可能成功连接的若干时序数据组成时序数据对加入候选集合;验证阶段为每对候选集合中的时序数据对计算 DTW 值,确定最终的连接结果. 然而,此框架通常针对字符串相似连接,现有的基于前缀过滤或分段过滤等字符串相似连接技术在 DTW 相似连接问题中存在两个缺陷:(1) 时序数据的长度通常超过 $10^4$,验证阶段用时较长;(2) 由于具有不同的数据类型和相似性度量方式,现有技术不能直接应用在 DTW 相似连接上.

另一方面,DTW 相似搜索对查询时序数据进行变换,而在相似连接时对全部时序数据进行变换会导致较高的时间和空间代价. 其他工作通过牺牲 DTW 计算上的准确度,以随时序数据长度线性增长的计算代价得到给定时序数据对的近似 DTW. 这样不仅造成连接结果不准确,而且由于时序数据较长,对全部时序数据对计算近似 DTW 依然需要较长时间.

为了克服上述困难,本文采取使用较少时间为每对时序数据计算 DTW 相似度的上下界的方法避免计算所有时序数据对的准确 DTW,从而提高相似连接的性能. 特别地,本文提出了两个新颖的算法分别用于快速计算任意一对时序数据的 DTW 上界和下界. 具体地,我们采用基于划分的方法比准确 DTW 计算的时间成倍地减少了计算时序数据 DTW 上界的时间. 我们还考虑时序数据的取值分布以 $O(n\log n)$ 的时间复杂度计算一对时序数据的 DTW 下界. 这既减少了需要计算准确 DTW 的时序数据对个数,又实现了比计算近似 DTW 还低的计算代价.

本文贡献如下:

(1) 首次提出基于 DTW 的时序数据相似连接问题,并分别定义基于阈值和基于 Top-$k$ 的两个 DTW 时序数据相似连接任务.

(2) 提出两个新颖的 DTW 上界和下界计算方法,高效地解决了本文提出的两种基于 DTW 的时

序数据相似连接问题.

(3)使用真实的电网数据和文本数据验证了基于 DTW 的时序数据相似连接问题的实际应用需求.

(4)在真实和合成数据上进行的充分实验验证了本文提出的基于 DTW 相似连接的性能和所提出的两种上下界计算方法的效果.

本文第 1 节描述问题的背景和意义;第 2 节介绍问题定义;第 3 节介绍相关工作;第 4 节给出基于 DTW 相似性度量方法的相似连接算法;第 5 节使用实验来验证所提问题的实际应用效果和算法效率;第 6 节总结全文以及提出未来工作.

## 2　问题定义

我们首先定义本文所处理的时间序列数据,然后定义 DTW 相似度的一个特例:欧几里得距离,最后定义 DTW 相似度和基于 DTW 的相似连接.

**定义 1.**　一个时间序列 $T$ 是一个长度为 $m$ 的有序列表 $T=[t_0,t_1,\cdots,t_{m-1}]$,这里 $|T|=m$.

例如,表 1 分别展示了图 1 中设备 1、设备 2 和设备 3 的时间序列 $T_1$,$T_2$ 和 $T_3$.

**表 1　设备 1、设备 2 和设备 3 的时序数据**

| | 长度=14 的时间序列 | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 设备 1($T_1$) | 121 | 779 | 781 | 769 | 121 | 103 | 131 | 786 | 779 | 781 | 122 | 121 | 103 | 789 |
| 设备 2($T_2$) | 789 | 235 | 679 | 262 | 647 | 228 | 697 | 212 | 691 | 206 | 645 | 246 | 613 | 240 |
| 设备 3($T_3$) | 121 | 103 | 131 | 131 | 789 | 781 | 788 | 121 | 103 | 131 | 786 | 779 | 781 | 769 |

**定义 2.**　时间序列 $T_i$ 和 $T_j$($|T_i|=|T_j|$)上的欧氏距离被定义为

$$ED(T_i,T_j)=\sqrt{\sum_{k=0}^{m-1}(T_i[k]-T_j[k])^2}.$$

根据定义 2,如图 3(a)所示,长度相同的时间序列 $T_1$ 和 $T_3$ 间的欧式距离是按黑色虚线将两个序列一一对应后,相应差值(黑色虚线投影到 $Y$ 轴的长度)的平方和的算数平方根.可以看出,$T_1$ 和 $T_3$ 在欧式距离上较大的差异主要来自于 12 个数值差异较大的数据点对.但这样的数据点一一对应得到的两条时间序列上的距离并不能真实反应实际情况中两条时间序列的相似性.例如,时间序列 $T_1$ 和 $T_3$ 实际表示在时间上有延迟的相同的变化趋势.当我们将 $T_3$ 向前移动三天,我们可以发现 $T_1$ 和 $T_3$ 是重合的.这样在时间上具有延迟效应的相似变化趋势在实际中比较常见,除了例 1 中供电负载变化由于供电线路造成的具有延迟 3 天的相关性外,例 2 中不同词汇由于中英文表达相同文本所用单词量不同,也造成含义相同的词汇得到的时序数据具有延迟相似的效应.为了发现时序数据上具有延迟效应的相似变化趋势,DTW 相似度被广泛采用.

与欧式距离将两个时间序列一一对应不同,DTW 相似度允许序列中多个点对应到另一个序列的一个点,再将两个序列最优对应下的差值平方和的算数平方根作为相似度.

为了将长度分别为 $n$ 和 $m$ 的时间序列 $T_i$ 和 $T_j$ 进行对应,DTW 构建一个 $n\times m$ 的矩阵 $\boldsymbol{w}$,其每个元素 $\boldsymbol{w}[r,c]=(T_i[r]-T_j[c])^2$ 表示 $T_i[r]$ 与 $T_j[c]$ 对应时的欧式距离.

这样,如图 3(b)中实心方块所示,两个序列的一个匹配就对应于矩阵元素 $\boldsymbol{w}_p=\boldsymbol{w}[r_p,c_p]$ 的一条路径是 $[w_1,w_2,\cdots,w_p,w_{p+1},\cdots,w_q]$,其中 $\max\{n,m\}\le q\le n+m$. DTW 中路径需要满足的条件:

(1)路径必须以矩阵的左下角元素和右上角元素作为起始和终止,即 $\boldsymbol{w}_1=\boldsymbol{w}[0,0]$ 和 $\boldsymbol{w}_q=\boldsymbol{w}[n,m]$.

(2)路径中元素必须向相邻元素转移,因此路径中相邻的元素 $\boldsymbol{w}_p=\boldsymbol{w}[r_p,c_p]$ 和 $\boldsymbol{w}_{p+1}=\boldsymbol{w}[r_{p+1},c_{p+1}]$ 要满足 $c_p=c_{p+1}$,$r_p+1=r_{p+1}$,或 $c_p+1=c_{p+1}$,$r_p=r_{p+1}$,或 $c_p+1=c_{p+1}$,$r_p+1=r_{p+1}$.

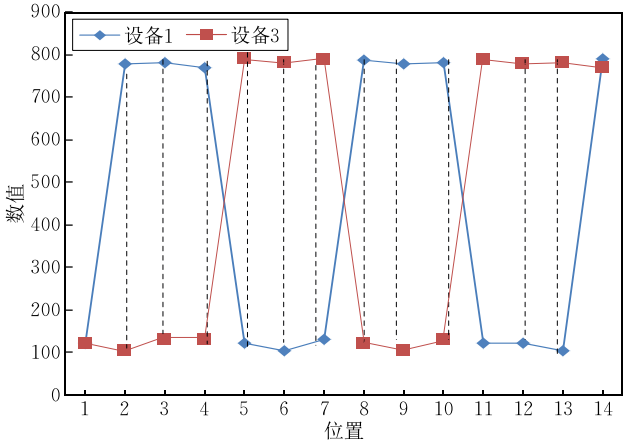
**定义 3.**　给定时序数据 $T_i$ 和 $T_j$,它们的 DTW 相似度是所有满足条件的路径的差值平方和的算数平方根中的最小值,即

$$\mathrm{DTW}(T_i,T_j)=\min_{path}\left\{\sqrt{\sum_{w_p\in path}\boldsymbol{w}_p}\right\}.$$

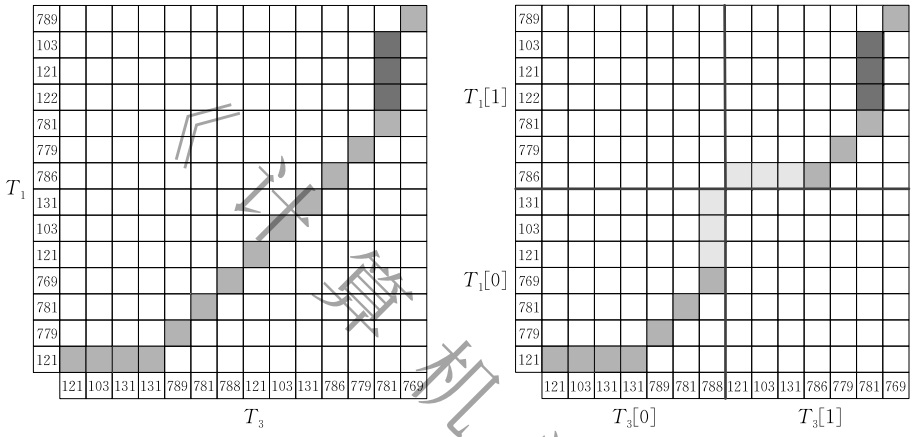
对于时序数据 $T_i$ 和 $T_j$,动态规划的转移方程

$$\gamma(r,c)=W[r,c]+\min\{\gamma(i-1,j),\gamma(i,j-1),\gamma(i-1,j-1)\}$$

可以得到差值平方和最小的路径对应的 DTW 值 $\gamma(m,n)$.值得注意的是,越小的 DTW 相似度意味着两条时序数据越相似.图 3(b)中实心方块表示了时序数据 $T_1$ 和 $T_3$ 间的最优对应.对比图 3(a)的欧氏距离,$T_1$ 和 $T_3$ 间较小的 DTW 相似度只由 3 个黑色方块表示的具有较大差异的数据对构成.这与图 1 中 $T_1$ 和 $T_3$ 具有相似的变化趋势相吻合.




(a) 设备1与设备3之间的欧式距离




(b) $T_1$和$T_3$间的DTW距离　　(c) $T_1$和$T_3$的DTW上界

图 3　设备 1 与设备 3 间的 DTW 相似度

下面分别定义例 1 和例 2 中的相似连接任务.

**定义 4**(基于阈值的 DTW 相似连接).　给定时序数据集合 $S_1$,$S_2$和阈值 $\theta$,基于阈值的相似连接返回所有时序数据对$(T_i,T_j)$,其中 $T_i\in S_1$,$T_j\in S_2$,$\mathrm{DTW}(T_i,T_j)\leq\theta$.

**定义 5**(基于 Top-$k$ 的DTW相似连接).　给定时序数据集合$S_1$,$S_2$和整数 $k$,基于 Top-$k$ 的相似连接对$\forall T_q\in S_1$返回一个子集$S\subseteq S_2$,当$|S|=k$,对$\forall T_i\in S$,$\forall T_j\in S_2-S$,有 $\mathrm{DTW}(T_q,T_i)\leq\mathrm{DTW}(T_q,T_j)$.

例如,阈值 $\theta=5$ 时,$S_1$和 $S_2$都是例 1 中配电曲线集合的基于阈值的相似连接将具有延迟三天关系的数据对〈设备 1,设备 3〉作为查询结果;$k=1$ 时,$S_1$和 $S_2$分别是例 2 所示的英文和中文词汇的时序数据的 Top-1 相似连接将两种语言中具有相同含义的词汇对作为查询结果.

## 3　相关工作

本文首次提出基于 DTW 的相似连接问题,其相关工作主要包括两类:基于 DTW 的搜索问题和针对其他数据类型或度量方式上的相似连接问题.

### 3.1　基于 DTW 的搜索问题

基于 DTW 进行相似搜索的工作主要提出了计算两条时间序列的 DTW 相似度下界的方法[11-16]和近似计算 DTW 的方法[17-20].其中,Kim 下界计算给定时序数据与待匹配子时间序列间首尾数值以及最大值之间的差异之和,并以此作为 DTW 相似性的下界[9].Yi 下界首先找出两条序列中最大值较小的序列,并对另一序列中所有高于这个值的数据与其差值作为两条时序数据 DTW 相似度的差值[12].最新的工作 Keogh 下界则将长度为 $m$ 的时间序列通过循环置换重写成 $m$ 个长度为 $m$ 的时间序列,并得到紧的 DTW 下界[13].

近似计算 DTW 的方法是通过避免构建完整的矩阵 $\boldsymbol{w}$ 来高效计算近似 DTW 相似度.其中,基于封装的方法将时序数据映射成近似的向量,并基于向量构建矩阵上的路径[18,21].FastDTW 只计算 $\boldsymbol{w}$ 中偏离对角线较小的区域内的路径[17],Local DTW 只

允许计算以 $w$ 的对角线为对角线的窄平行四边形范围内的路径[18].

这些计算 DTW 相似性下界的方法和近似计算 DTW 的方法并不能直接应用于本文的问题. 对于计算 DTW 相似性下界的方法来说,尽管理论上我们可以对每条时序数据用计算 DTW 相似性下界的方法裁剪掉不可能成为相似连接结果的时序数据,实际上现有方法中得到的紧的 DTW 下界的方法依赖 DTW 查询中查询序列较短的假设,这在相似连接问题中是不适用的. 例如,在实验数据集上目前最好的 Keogh 下界[13]将长度为 $m$ 的时间序列重写为 $m$ 个长度为 $m$ 的时间序列所需的空间远远超出整个实验数据集的大小. 此外,其他如 Kim 下界和 Yi 下界只考虑了若干时序数据中的特征,而本文提出的下界考虑了更为全面的数值分布特征,因此得到了更好的下界. 计算近似 DTW 的方法由于牺牲了准确性,可以作为本文的补充,在对 DTW 相似度准确性不敏感的情况下使用.

### 3.2　其它数据类型上的相似连接问题

本小节主要涉及字符串上的相似连接问题和集合上的相似连接问题.

针对集合上的相似连接问题,文献[22]提出了将集合元素排序后进行前缀过滤的方法. 文献[23]在此方法的基础上使用位置过滤和后缀过滤改进了前缀过滤的效果. 特别地,它们发现如果两个集合对在基于阈值的相似连接结果中存在,这两个集合的不同元素不应超过由阈值导出的一个数值. 文献[24]考虑了更多因素进一步增强了前缀过滤的效果. 文献[25]为基于 Jaccard 相似性和编辑距离等不同相似性度量方式的集合连接问题提出了一般的解决方法. 文献[26]为数据库增加了划分和枚举两个操作符实现了数据库中的集合相似连接操作. 文献[27]扩展了前缀过滤技术实现了基于 Top-$k$ 的集合相似连接. 文献[28,29]在分布式计算框架 MapReduce 上实现了集合相似连接.

针对字符串上的相似连接问题[30],文献[26,31]提出了一种新的操作符,并使用 SQL 语句在关系数据库中进行字符串相似连接操作. 文献[32]提出一种可以过滤掉不可能作为相似连接结果的字符串对的签名方法. 文献[33]提出了基于转化的框架来定义不同相似字符串来完成相似连接的方法. 最近的工作通常使用"过滤-验证"机制[34],并应用前缀过滤[35]、分段过滤[36]等技术进行字符串上的相似连接.

尽管我们可以将时序数据视为字符串甚至集合,但由于两两集合元素或两两字符串之间只有相同和不同关系,而时序数据间由于不同的数值间有差距大小的关系,因此基于集合或字符串的相似连接方法并不能直接应用于本文定义的基于阈值或基于 Top-$k$ 的相似连接问题.

### 3.3　其它度量方式上的相似连接问题

除了基于集合和字符串上的相似连接使用 Jaccard 相似性、编辑距离等于 DTW 不同的相似性度量方法外,基于时序数据的其它工作还有利用欧式距离等相似性度量方法进行相似连接的方法.

文献[37-39]使用 R 树或空间哈希等方法为数据库中的时序数据的子序列建立索引,然后利用索引进行连接操作. 特别地,文献[36]为每个数据库建立一棵 R 树,使用深度优先的方法搜索 R 树,最后得到叶子节点的相似子序列对. 文献[38]使用广度优先的方法提升了这种方法的性能. 文献[39]将子序列哈希到不同数据桶中并使用不同桶中的子序列作为连接候选结果. 这些方法一般用于度量方式是准确连接的时序数据连接任务. 由于大体相似但有细微不同的时间序列会被映射到不同的 R 树叶子节点或不同的哈希值,这些方法不能直接应用于本文的相似连接任务.

还有部分工作考虑流数据上如何使用少量内存对最近的时序数据进行连接[40]和图数据上基于编辑距离的相似连接问题[41]. 但前者是基于欧式距离的相似连接方法[40],后者使用节点间路径的相似性作为度量方法,也不能应用于本文基于 DTW 的相似连接任务.

最近,为了提高处理时序数据上相似搜索的效率,近似的时序数据表示方法被相继提出. 最直接的表示方法是将时序数据切成片段并使用每段中的平均值代表每一个片段. 例如,PAA 将时序数据均匀切片[42],也有其他方法采用自适应的切片方式[43]. 此外,还有部分工作采用将时序数据进行离散傅里叶变换,奇异值分解或离散小波变换进行表示的方法[44,45]. 最新的工作使用基于集合的方法表示时序数据[46]. 这些方法在数据类型和相似性度量方式上均与本文有较大区别.

## 4　相似连接查询处理

本文基于 DTW 上下界的计算统一处理基于阈

值和基于 Top-$k$ 的两种时序数据上的相似连接问题.4.1 节首先介绍 DTW 上下界在两种相似连接问题中的应用,4.2 和 4.3 节分别介绍本文提出的 DTW 上界和下界计算方法和相应参数的设置方法.

### 4.1 基于 DTW 上下界的相似连接

本节介绍两种相似连接任务如何利用 DTW 上下界减少计算数据对间的准确 DTW.

在基于阈值的相似连接中,算法 1 利用以下两个事实减少了准确 DTW 相似度的计算.如果时序数据 $T_i$ 和 $T_j$ 的 DTW 上界小于给定阈值,无需计算它们的 DTW 相似度就可以知道它们属于相似连接结果集合(第 3～4 行);如果它们的 DTW 下界大于给定阈值,无需计算它们的 DTW 相似度就可以知道它们不属于相似连接结果集合(第 5～6 行).

**算法 1.** 基于阈值的相似连接算法.

输入:时序数据集合 $S_1$,$S_2$,阈值 $\theta$

输出:连接结果时序数据对的集合 $ret$

1. $ret=\varnothing$
2. For each $T_i\in S_1$
3. For each $T_j\in S_2$
4. If ($DTW_{UB}(T_i,T_j)\leq\theta$)
5. $ret=ret\cup\{(T_i,T_j)\}$
6. Else If ($DTW_{LB}(T_i,T_j)>\theta$)
7. continue
8. Else If ($DTW(T_i,T_j)\leq\theta$)
9. $ret=ret\cup\{(T_i,T_j)\}$
10. End If
11. End For
12. End For
13. return $ret$

在基于 Top-$k$ 的相似连接中,算法 2 利用以下两个事实减少了准确 DTW 相似度的计算:

(1)针对任意时序数据 $T_p\in S_1$,如果另一个时序数据 $T_p$ 与 $T_q$ 的 DTW 相似度的下界大于其它 $k$ 个时序数据与 $T_q$ 间 DTW 相似度的上界,$T_p$ 一定不会存在于 $T_q$ 的 Top-$k$ 相似连接结果中(第 7～8 行);(2)否则,如果 $T_p$ 与 $T_q$ 的 DTW 相似度的上界小于 $k$ 个时序数据中的一个时序数据 $T_e$ 与 $T_q$ 的 DTW 相似度下界,则 $T_e$ 一定不会存在于 $T_q$ 的 Top-$k$ 相似连接结果中(第 11～16 行).

**算法 2.** 基于 Top-$k$ 的相似连接算法.

输入:时序数据集合 $S_1$,$S_2$,整数 $k$

输出:连接结果时序数据对的集合 $ret$

1. $ret=\varnothing$
2. For each $T_q\in S_1$
3. $H=\varnothing$
4. For each $T_q\in S_2$
5. If ($|H|<k$)
6. $H=H\cup\{T_p\}$
7. Else If ($DTW_{LB}(T_p,T_q)>\max\limits_{t'\in H}DTW_{UB}(t',T_q)$)
8. continue
9. Else
10. $H=H\cup\{T_p\}$
11. End If
12. While ($|H|>k$)
13. $T_e=\arg\max\limits_{t'\in H}DTW_{LB}(t',T_q)$
14. If ($DTW_{UB}(T_p,T_q)<DTW_{LB}(T_e,T_q)$)
15. $H=H-\{T_e\}$
16. Else
17. break
18. End If
19. End While
20. End For
21. For each $T_p\in H$
22. calculate DTW($T_p$,$T_q$)
23. End For
24. sort $H$ by DTW in ascending order
25. $ret=ret\cup(H[0:k-1]\times\{T_q\})$
26. End For
27. return $ret$

### 4.2 基于 DTW 的上界计算

我们看到,上述两种算法均需要快速计算时序数据 $T_i$ 和 $T_j$ 的 DTW 的上界.我们提出基于划分的 DTW 上界计算方法.具体地,我们将 $T_i$ 和 $T_j$ 分别平均切成 $g_1$ 份,并依次计算 $T_i$ 的第 $k$ 份 $T_i[k]$ 与 $T_j$ 的第 $k$ 份 $T_j[k]$ 之间的准确 DTW.如图 3(c)所示,$T_1$ 和 $T_3$ 被切成均等的 $g_1=2$ 份,得到 $T_1$ 和 $T_3$ 的子序列 $T_1[0]$,$T_1[1]$,$T_3[0]$ 和 $T_3[1]$.其中 $T_1[0]$ 和 $T_1[1]$ 之间以及 $T_3[0]$ 和 $T_3[1]$ 之间的 DTW 距离分别按图 3(c)中左下角和右上角的正方形所示计算.我们可以看到,由于分割后 $T_1[0]$ 中后 3 个时序数据 121,103 和 131 无法像图 2(b)中那样与 $T_3[1]$ 中的前 3 个时序数据 121,103 和 131 匹配,分割后两对子序列的 DTW 之和相比 $T_1$ 和 $T_3$ 间的准确 DTW 增加了 6 对标记为最浅色的具有较大差异的数值间的欧氏距离.下面的定理 1 证明了这样得到的子序列 DTW 之和是完整序列的 DTW 上界.

**定理 1.** 给定时序数据 $T_i$ 和 $T_j$,如果将它们均匀分割成 $g_1$ 份,则 $\mathrm{DTW}(T_i,T_j)\leq\sum\limits_{g=0}^{g_1}\mathrm{DTW}(T_i[g],T_j[g])$.

证明.　$T_i$和$T_j$划分得到的各个子序列依次计算 DTW 时形成的匹配可以形成 $T_i$和 $T_j$上进行匹配的一个路径，而它们的 DTW 之和正是这个路径对应的欧式距离之和. 由于 DTW($T_i$,$T_j$)被定义为 $T_i$和 $T_j$上所有匹配形成路径对应的最小欧式距离，定理 1 得证.　　证毕.

由于时序数据被划分成 $g_1$份，每一份计算 DTW 的时间复杂度为 $O[(n/g_1)^2]$，$g_1$个 DTW 的和所需时间复杂度为 $O[(n/g_1)^2\times g_1]=O(n^2/g_1)$. 因此，通过增加划分份数，我们可以成倍减少计算时间.

较大的 $g_1$会有较好的加速效果，但 DTW 计算的任务启动开销增加，计算每对子序列上 DTW 所占的比重也增加，并且会导致每份子序列较短，引起序列间隔附近的时序数据无法匹配而增大 DTW 上界与真实值的偏差，进而减弱相似连接的裁剪效果. 为了权衡任务启动开销的时间，运行时间和加速效果，我们采用如下方法确定 $g_1$：假设任务的启动时间是 $t_s$，系统计算一个欧式距离的时间是 $t_c$，我们首先要求每个任务中进行 DTW 计算的时间多于任务的启动时间，则每个子时间序列长度至少是 $\sqrt{t_s/t_c}$. 这样，初始的 $g_1=\sqrt{t_s/t_c}$. 为了避免过大的 $g_1$造成 DTW 上界与真实值偏差过大，我们开始不断迭代计算 $g_1=g_1\div 2$ 的情况. 因为 $g_1$较大时运行速度很快，我们可以得到 DTW 上界随运行时间增加而减少的曲线，当减少速度下降时，即曲线的二阶导数非正时，迭代停止.

### 4.3　基于 DTW 的下界计算

现有的紧的 DTW 下界计算方法[12]需要对每条时序数据进行所需空间为 $O(n^2)$的预处理，远远超出整个数据集大小，因此只适用于序列较短且较少的 DTW 相似查询；其它常用的 DTW 下界计算方法是假设两条时序数据的最大值不同. 如图 1 所示，这种方法在 $T_1$，$T_2$和 $T_3$的最大值相同时得到的 DTW 下界为 0，没有效果.

为了克服现有方法存在的缺陷，我们考虑两条时序数据在数值分布上的差异，而不仅仅考虑时序数据对最大值间的差异. 具体地，我们用等宽直方图刻画时序数据的数值分布，将时序数据的值域划分成 $g_2$个区间，并统计每条时序数据在每个区间上的频数. 表 2 显示了当 $g_2=4$ 时，时序数据 $T_1$，$T_2$和 $T_3$的值域(0,800]被等宽直方图均匀划分为(0,100]，(200,300]，(500,600]和(700,800]这 4 个区间. 其中 $T_1$和 $T_3$都只在(0,100]和(700,800]这两个区间上具有数值分布，而 $T_2$则在(200,300]，(500,600]和(700,800]这 3 个数值范围具有数值分布.

**表 2　设备 1、设备 2 和设备 3 在数值上的分布**

| 设备/区间 | 设备 1($T_1$) | 设备 2($T_2$) | 设备 3($T_3$) |
|---|---|---|---|
| (0,100] | 7 | 0 | 7 |
| (200,300] | | 7 | |
| (500,600] | | 6 | |
| (700,800] | 7 | 1 | 7 |

为了计算 DTW 下界，我们假设相同区间内的时序数据都可以匹配且差异为 0，不同区间之间如果进行匹配则按距离最近的端点计算距离. 这样，$T_1$和 $T_3$之间在分布上的差异为 0，而 $T_2$在(200,300]这个区间里的 7 个数值只能和 $T_1$和 $T_3$的距离(200,300]最近的(0,100]中的数值进行匹配以取得可能的最小差异. 这样，当 $T_2$在(200,300]中的数值都取为 200，$T_1$和 $T_3$在(0,100]中的数值都取为 100 时，我们得到最少的 DTW 差异为 $7\times(200-100)$. 类似地，$T_2$在(500,600]这个区间里的 6 个数值为了取得最小差异只能与它最接近的 $T_1$和 $T_3$的(700,800]区间中的数据进行匹配，差异最小是 $6\times(700-600)$. 这样，我们可以看到，尽管 3 个时序数据的最大值相同，但它们的 DTW 下限是不同的.

算法 3 描述了这个计算方法. 首先，全部时序数据的取值范围被等分为 $g_2$份(第 1～3 行)，并分别计算每条时序数据落在每个区间中的数值的个数(第 4～7 行). 对于任意时序数据 $T_i$和 $T_j$，我们针对 $T_i$的每个区间，首先使用二分查找寻找 $T_j$的区间比这个区间小和大的最接近的非空区间(第 9～11 行). 若 $T_j$中存在相同的非空区间，则下界不增加(第 12～13 行)，否则选择距离最近的区间计算差值并加入下界(第 14～20 行). 抛去预计算时划分区间的时间，算法 3 计算 DTW 下界复杂度是 $O(m+g_2\log g_2)$. 下面的定理 2 证明这样得到了 DTW($T_i$,$T_j$)的一个下界.

**算法 3.**　动态时间规整下界算法.

输入：时间序列 $T_i$和 $T_j$，整数 $g_2$

输出：$T_i$和 $T_j$的下界

1. max=max$_k$\{$T_i[k]$,$T_j[k]$\}
2. min=min$_k$\{$T_i[k]$,$T_j[k]$\}
3. $gap=(\max-\min)/g_2$
4. For each $T_i[k]\in T_i$
5. 　$D_i[(T_i[k]-\min)/gap]$++
6. End For
7. For each $T_j[k]\in T_j$

8. $D_j[(T_j[k]-\min)/gap]++$
9. End For
10. $DTW_{LB}(T_i,T_j)=0$
11. For each $D[k]>0$
12. $D_j[k_{less}]=$bSearchLargest$(D_j,D_i[k]_{left})$
13. $D_j[k_{more}]=$bSearchLeast$(D_j,D_i[k]_{right})$
14. If $(D_j[k_{less}]_{left}==D_i[k_{more}])$
15. continue
16. End If
17. $left_{gap}=D_i[k]_{left}-D_j[k]_{left}$
18. $right_{gap}=D_j[k_{more}]_{left}-D_i[k]_{right}$
19. If $(left_{gap}>right_{gap})$
20. $DTW_{LB}(T_i,T_j)=D_i[k]\times right_{gap}^2$
21. Else
22. $DTW_{LB}(T_i,T_j)=D_i[k]\times left_{gap}^2$
23. End If
24. End For
25. return $DTW_{LB}(T_i,T_j)$

**定理 2.** 给定时序数据 $T_i,T_j$,将它们的取值等分为 $g_2$ 份,则算法 3 得到的距离之和不大于 DTW$(T_i,T_j)$.

证明. 对于 $T_i$ 中的任意数值 $a$,假设在 DTW 中它与 $T_j$ 中的数值 $b$ 进行匹配,假设 $a$ 与 $b$ 落在算法 3 得到的相同的区间中,则它们的距离被计算为 $0\leq(a-b)^2$. 假设 $a$ 与 $b$ 落在不同的区间$[I_1,I_2]$,$[I_3,I_4]$中,则算法 3 中它们的距离为$(I_3-I_4)\leq(a-b)^2$. 注意到不等号左侧为算法 3 得到的距离,右侧为 DTW 得到的距离,我们知道累加左侧算法 3 得到的距离不大于累加右侧 DTW 得到的距离. 定理 2 得证. 证毕.

较小的 $g_2$ 将使得实际差异较大的数值落在一个区间中进而得到与真实值相差更大的 DTW 下界,进而减弱相似连接的裁剪效果;由于计算 DTW 下界复杂度是 $O(m+g_2\log g_2)$,较大的 $g_2$ 会导致区间数增加,增加 DTW 下界的计算时间,减少分组开销 $m$ 所占比重. 为了权衡 DTW 下界计算时间,分组开销和裁剪效果,我们采用如下方法确定 $g_2$:假设对每个时序数据中的元素进行分组开销是 $t_o$,分组后每组计算时间是 $t_c$,则 $g_2$ 应该满足 $mt_o/t_c\leq g_2\log g_2$. 注意到等号右边随 $g_2$ 的增加而单调增加,我们令 $g_2$ 的初始值为 1,然后使用二分法高效地得到最大的整数 $g_2$ 使得不等式成立. 得到 $g_2$ 的初始值使得分组代价较小后,我们开始不断迭代计算 $g_2=2\times g_2$ 的情况. 因为 $g_2$ 较小时运行速度很快,我们可以得到 DTW 下界随运行时间增加而减少的曲线,当减少速度下降时,即曲线的二阶导数非正时,迭代停止.

### 4.4 扩展到 MapReduce 框架

由于相似连接仍然涉及双重循环计算,当数据量较大,查询处理依然需要较多时间时,我们可以将上下界计算框架在分布式计算框架 MapReduce 下实现. 当数据被存储在分布式文件系统,如 HDFS 中以后,我们只需要一个 MapReduce 任务进行一轮计算就可以得到相似连接查询结果. 在一轮 MapReduce 计算中,每个 map 任务负责找到一条时序数据的相似连接结果. 例如,每条时序数据 $T_i\in S_1$ 被分配给一个 map 任务,这个 map 任务从分布式文件系统中得到 S2 的所有时序数据,并使用算法 1 对第 3 行到第 11 行的处理过程得到 $T_i$ 的基于阈值的相似连接结果,或使用算法 2 对第 3 行到第 22 行的处理过程得到 $T_i$ 的基于 Top-$k$ 的相似连接结果. 这些连接结果是时序数据对$\langle T_i,T_j\rangle$的集合. 我们将每个 map 任务得到的数据对$\langle T_i,T_j\rangle$按照 $i$ 和 $j$ 的字典顺序组成键值映射到 reduce 任务,这样等价的时序数据对会被映射到同一个 reduce 任务. 例如,如果$\langle T_i,T_j\rangle$是相似连接结果,那么负责 $T_i$ 的 map 任务和负责 $T_j$ 的 map 任务会产生两个重复的相似连接结果,而这两个重复的结果由于具有相同的键值会被映射到一个 reduce 任务中,这样 reduce 任务去掉重复的查询结果后就得到了整个查询的结果. 这样基于并行计算框架 MapReduce 的解决方案扩展了我们的方法在大数据领域的应用.

## 5 实 验

本节中我们同时使用真实和合成数据验证本文方法的性能与效果. 实验说明了下述问题:(1)基于 DTW 的时间序列相似度连接在相关性分析和双语名词匹配方面的实际应用;(2)本文提出的相似连接算法与上下界计算方法是有效的.

### 5.1 实验环境与数据集描述

我们使用 C++实现本文方法,实验环境是一台 Linux 服务器,英特尔至强 E5645 2.4 GHz 处理器,8 GB 内存,1 TB SATA 硬盘. 我们实现了原本的 DTW 算法[5]、广泛应用的高效近似 DTW 计算方法 FastDTW[17]、广泛应用的 DTW 下界高效计算方法 Kim 下界[11]和 Yi 下界[12]以及当前效果最好的 Keogh 下界[13].

我们使用两个真实数据集以及合成数据集. 第

一个真实数据集包含某地电力部门 10 年的配电设备每 15 min 采集一次的实时功率，这样一共具有时间序列 103 条，其中每条的长度均经过分词后，我们统计每个词汇依次出现在每 100 个词汇中的频率. 这两个数据集分别包含了较长的时间序列和较短的时间序列. 合成数据生成不同长度的时间序列，并采用正态分布生成方差在(0,1]内随机取值的时序数据用于相似连接的性能测评. 各数据集的时间序列长度和个数情况见表 3.

**表 3　数据集**

| 数据集 | 时间序列长度 | 时间序列个数 |
|---|---|---|
| 配电网 | 350 400 | 103 |
| 英文文本 | 7500 | 2000 |
| 中文文本 | 5700 | 2000 |
| 合成数据 | $10^3 \sim 10^6$ | $10^2 \sim 10^4$ |

## 5.2　基于 DTW 相似连接的实际应用

配电网数据集除包含各配电设备的输电功率时序数据外，还提供了各配电设备所处区域的信息. 图 4 用相同的颜色显示了两个不同区域间的供电设备在时序数据正规化到[0,1]区间内，阈值 $\theta=5$ 时相似连接下的匹配情况. 我们可以看出，处于各集镇中心区域的供电设备被连接到一起，这揭示了这些区域的供电设备具有相同的功率变化趋势，例如在工作日白天输电功率上升，夜晚输电功率下降等. 利用 DTW 相似连接的结果，匹配在一起的供电设备的输电功率产生较大差异往往意味着输电线路出现异常，并触发电力公司的输电线路异常预警.

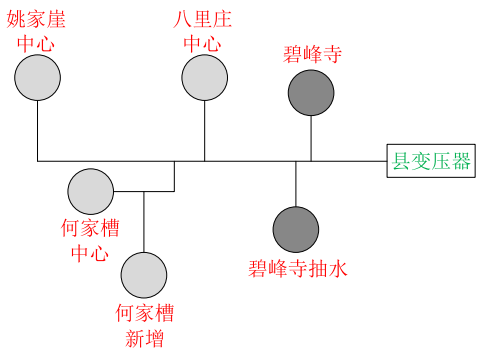



图 4　配电网数据集在 $\theta=5$ 时的相似连接结果

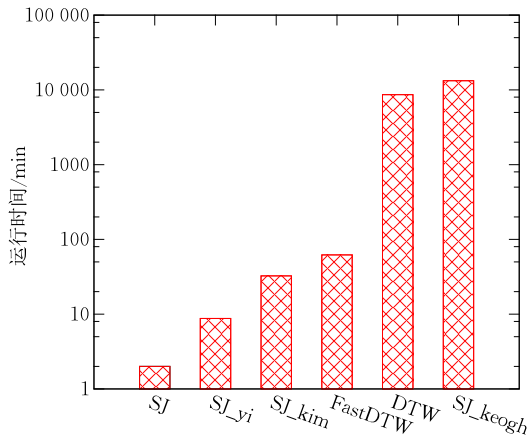
在第二份文本数据中，图 2 显示了英文的“God”一词和中文的“上帝”一词在每依次 100 个词中出现的频率的变化情况. 表 4 显示了 8 个 Top-1 相似度的匹配结果. 可以看出，“上帝”和“God”等不同语言下具有相同含义的名词实体很容易地被自动连接起来，这为双语实体匹配等任务提供了便利.

**表 4　Rest 数据集上的准确性比较**

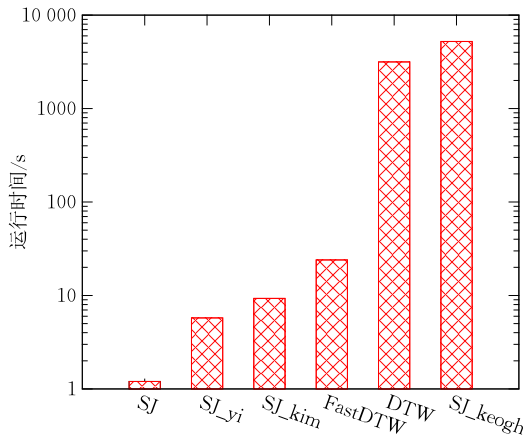
| 英文词汇 | 中文词汇 | 英文词汇 | 中文词汇 |
|---|---|---|---|
| God | 上帝 | People | 人们 |
| Father | 父亲 | King | 王 |
| House | 房屋 | Israel | 以色列 |
| Say | 说 | Land | 陆地 |

## 5.3　性能比较

图 5 分别显示了配电数据集和文本数据集上现有方法完成任务所需的时间. 对比图 5(a)和图 5(b)，我们可以看出，虽然文本数据集下时间序列的个数是电网数据集下时间序列个数的 20 倍，即不经过裁剪时需要比电网数据集多计算近 400 倍的 DTW 准确值，但文本数据集下相似连接所需的运行时间依然不到电网数据集下的六十分之一. 这是因为文本数据集中时序数据相对较短，而 DTW 计算时间是时序数据长度的三次方函数，而需要比较的时序数据对数是时序数据个数的二次方程函数，所以所花费时间更长. 这验证了我们使用 DTW 上下界裁剪需要计算 DTW 的数据对的合理性.




(a) 配电网数据集

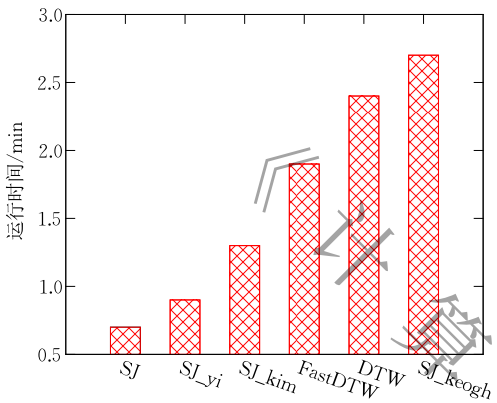



(b) 文本数据集

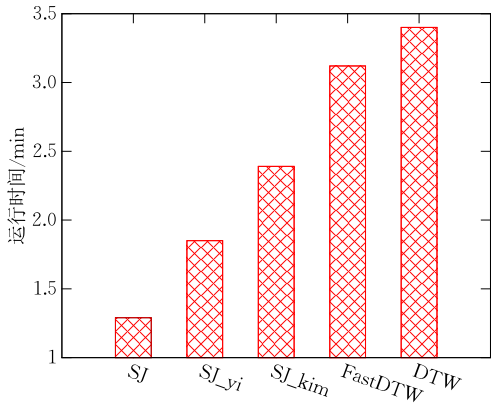
图 5　真实数据集上的性能对比

另一方面，我们可以看出，SJ，SJ_yi 和 SJ_kim 所需时间都小于基于 FastDTW 和 DTW 的方法的所需时间. 这是因为 DTW 和 FastDTW 不能避免对全部数据对进行相似度的计算，而前三种针对相似连接的方法尽管对需要计算 DTW 的数据对裁剪比例不同，但都大规模减少了 DTW 的计算. 特别地，本文方法在两个数据集中一致地比最新的方法快接近一个数量级以上. 特别值得注意的是，目前最好的 DTW 下界 Keogh 下界的运行时间是最慢的，比原始 DTW 方法还慢近一倍. 这是因为 Keogh 下界需要对每条曲线进行空间复杂性为 $O(n^2)$ 的处理. 在以后的实验中除非特殊说明，我们不包括 Keogh 下界的结果.

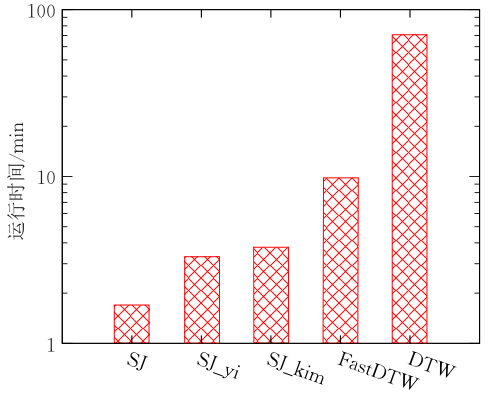
图 6 显示了合成数据集中不同长度的时间序列上各个方法的性能对比情况. 图 6(a)、(b)显示了当时序数据长度是 $10^2$ 时，各种方法在 $\theta=5$ 和 $k=10$ 的效果. 我们可以看到因为时序数据较短，DTW 的计算很快，各个上下界计算方法并没有数量级上的优势. 图 6(c)～(f)显示了当时序数据长度是 $10^3$ 和 $10^4$ 时，各种方法在 $\theta=5$ 和 $k=10$ 的效果. 我们可以看到当时序数据较长时各种基于计算上下界的方法的优劣和图 5(a)类似.



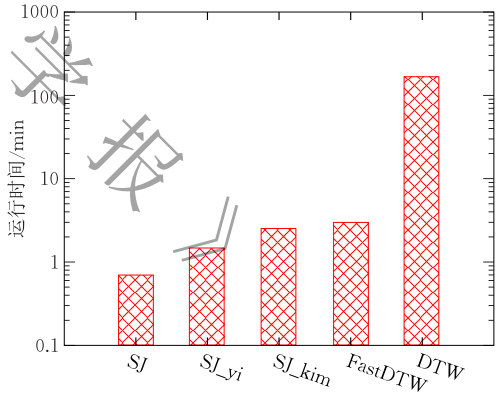

(a) 长度=$10^2$时$\theta=5$



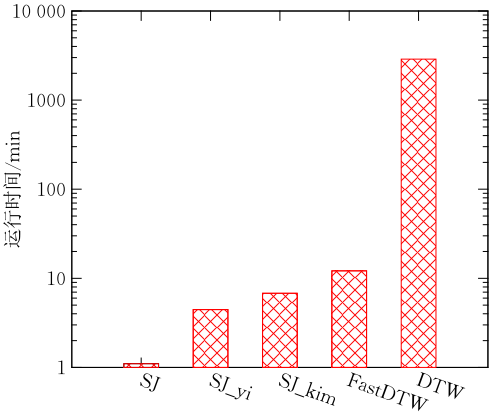

(b) 长度=$10^2$时$k=10$



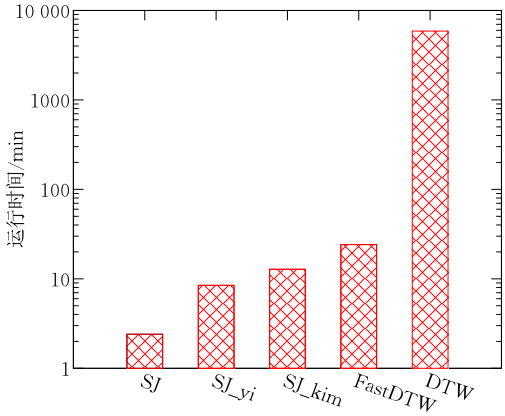

(c) 长度=$10^3$时$\theta=5$




(d) 长度=$10^3$时$k=10$




(e) 长度=$10^4$时$\theta=5$

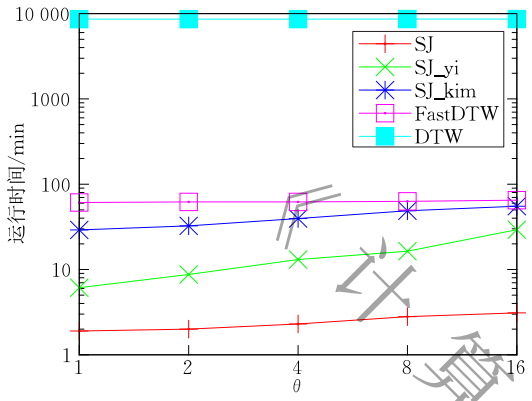



(f) 长度=$10^4$时$k=10$

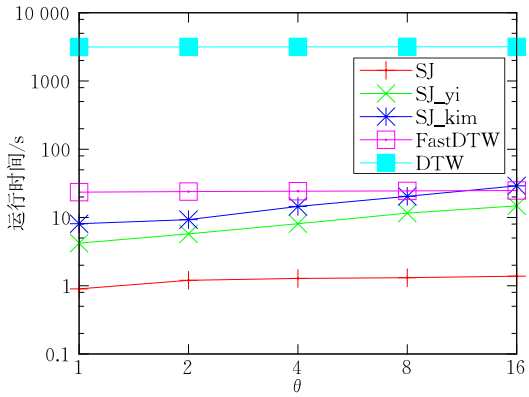
图 6 不同时序数据长度下的性能对比

图 7 显示了不同数据集上调整阈值 $\theta$ 后的相似连接运行时间. 我们可以看出，当 $\theta$ 增加时，由于结果集增大，各种方法的运行时间均有所增加. 特别地，Kim 下界和 Yi 下界的运行时间随 $\theta$ 增加显著变长，而我们的方法的运行时间增加不明显. 这是因为 $\theta$ 增加时其他方法由于下界不准确，需要进行验证的候选结果集较大，导致需要计算更多时序数据对的 DTW. 对比图 7(c)、(d)，我们可以看出，当 $\theta$ 确定而时序数据个数增加时，本文的 SJ 方法的运行时间没有显著增加. 这是因为本文的下界随着数据集长度的增加显著增加，裁剪掉了更多的时序数据对.

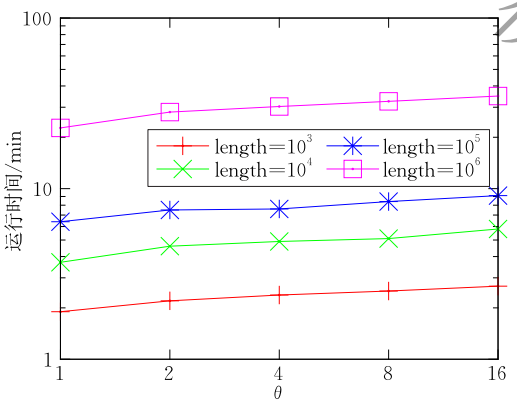
图 8 显示了不同 $k$ 值下的 Top-$k$ 相似连接的运行时间. 我们可以看出，当 $k$ 增加时，我们的 Top-$k$ 的 SJ 方法和基于 FastDTW 及原始 DTW 的方法的运行时间没有显著增加，而基于 Yi 下界和 Kim 下界的两种方法的运行时间显著增加. 一方面，这是因为基于 FastDTW 和 DTW 的方法需要对全部数据对求 DTW 值，因此 $k$ 的增加并不增加这两种方法的计算量；另一方面，基于 Yi 下界和 Kim 下界的两种



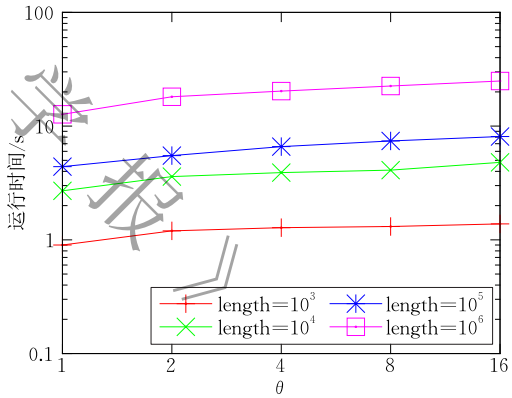

(a) 配电网数据集




(b) 文本数据集

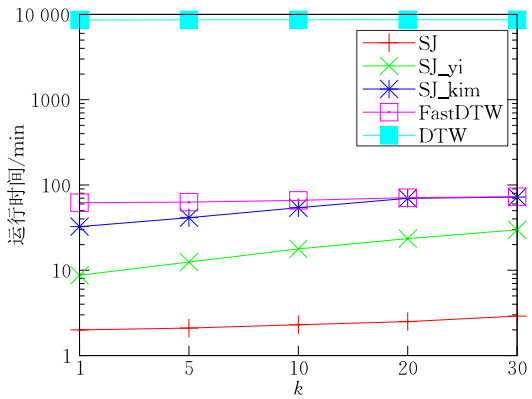



(c) $10^4$条合成时序数据

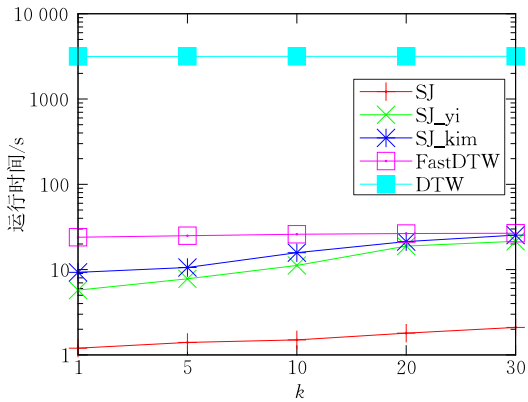



(d) $10^2$条合成时序数据

图 7　$\theta$ 对相似连接性能的影响




(a) 配电网数据集




(b) 文本数据集

图 8　$k$ 对相似连接性能的影响

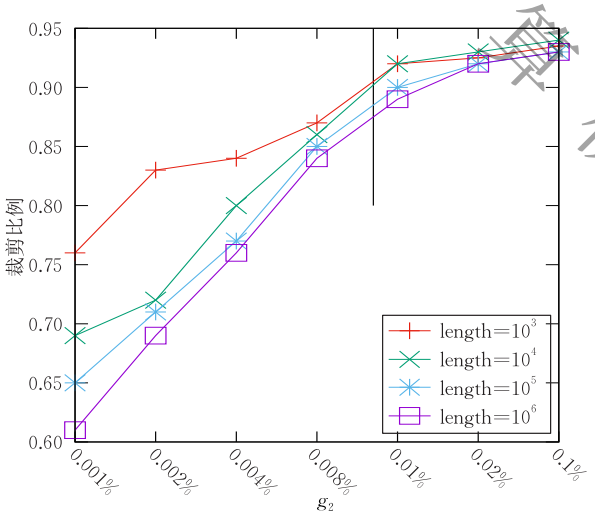
方法由于 $k$ 值增大,需要进行 DTW 计算的候选数据对显著变多,因此增加了验证过程中计算 DTW 的数据对个数.而我们的方法由于具有高效的上下界计算方法,在 $k$ 增加时需要计算 DTW 的数据对个数并没有显著增加,因此运行时间没有随 $k$ 的增加显著增长.

### 5.4 上下界计算效果与性能

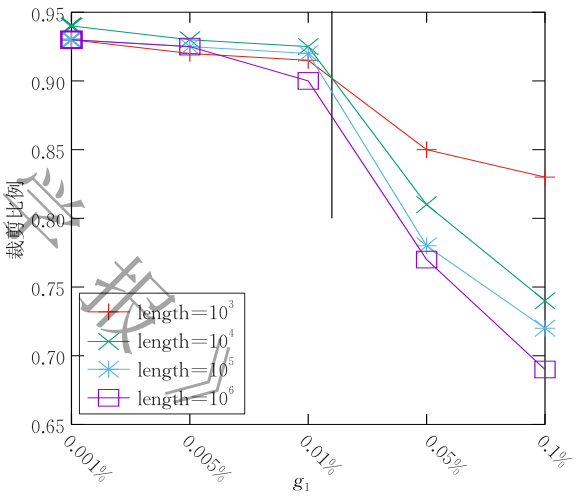
我们通过计算被裁剪的数据对占全部数据对比例衡量上下界计算方法的效果.其中,裁剪掉的数据比例越高说明上下界计算效果越好.

图 9 展示了不同 $g_1$ 和 $g_2$ 对性能的影响,同时用黑色竖线标出了计算 DTW 上界和下界的迭代结束时得到的 $g_1$ 和 $g_2$ 的位置.在固定 $g_1$ 为时序数据长度的 0.01%时,图 9(a)显示随着 $g_2$ 的增加,裁剪比例的增加速度在 $g_2$ 大于时序数据长度的 0.01%时逐渐变慢.这是因为当 $g_2$ 增加到数据长度的 0.01%时,大部分不能匹配的情况已落在不相邻的数据间隔中成为 DTW 下界的组成部分.当 $g_2$ 继续增加时,新的变化主要体现在较大的数值间隔被拆成相邻的两个数值间隔,这并不能显著增加 DTW 下界.固定 $g_2$ 为时序数据长度的 0.01%时,图 9(b)显示,裁减比例在 $g_1$ 小于 0.01%时增加缓慢,而在 $g_1$ 大于 0.01%时增加迅速.这是因为当 $g_1$ 为时序数据长度的 0.01%时,对于不同长度的时序数据,每个时序子序列的长度都在 10 000 左右,再增加子序列的长度得到的上界准确程度的增益有限.同时,我们可以看到迭代结束时得到的 $g_1$ 和 $g_2$ 具有较高的裁剪比例.

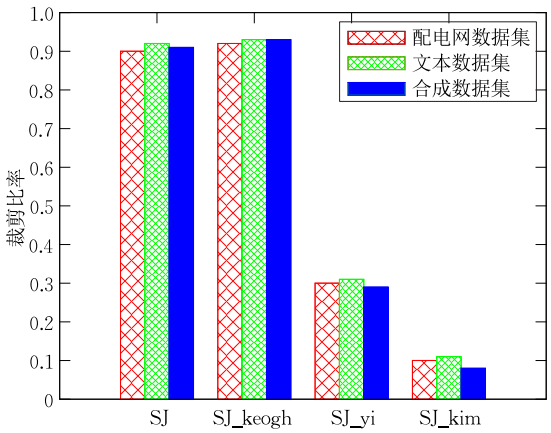
图 9(c)显示了 $g_1$ 和 $g_2$ 为时序数据长度的 0.01%时,我们的方法与最新方法 Keogh 下界以及常用的 Yi 下界和 Kim 下界效果的比较.结果显示,和已被证明是紧的下界 Keogh 下界相比,本文提出的方法的裁剪比例与其相当.此外,本文方法的裁剪比例是其他两种方法的 3 倍以上.图 9(d)验证了我们的上下界计算方法是高效的:我们的方法和 Kim 下界的



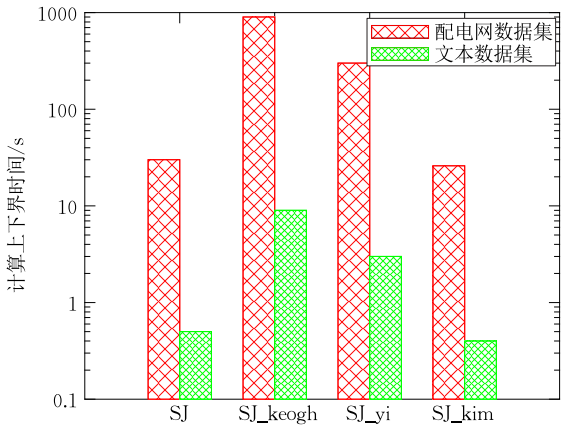

(a) 裁剪比例随$g_2$的变化




(b) 裁剪比例随$g_1$的变化




(c) 裁剪比例比较




(d) 上下界计算代价比较

图 9 上下界计算效果与代价

计算代价相当，远小于 Yi 下界和 Keogh 下界的计算代价.

## 6 结束语

本文提出了时间序列数据上一种新的查询类型：基于 DTW 的相似连接. 这种查询可以为时间序列上的数据挖掘任务（如跨语言词汇匹配、相关性分析等）提供重要的数据处理结果. 本文还首次定义了基于阈值和基于 Top-$k$ 这两种新颖的 DTW 相似连接任务. 为了提高查询处理效率，本文分别提出了时间序列间的 DTW 上界和下界计算方法. 本文在真实的配电网数据以及本文数据上验证了本文提出的两种 DTW 相似连接任务在实际中的应用，并进行了充分的实验证明本文提出的基于 DTW 的相似连接方法和 DTW 上下界计算方法是高效的.

本文的相似连接涉及两重循环计算，对大数据而言计算复杂度仍然偏大. 我们将在后续工作中关注如何支持某种索引机制，利用前缀过滤、分段过滤等字符串相似连接技术取得更高的裁剪效果，进一步加快大数据集下的连接操作.

**致　谢**　审稿专家和编辑老师提出了宝贵的修改意见和建议，在此表示衷心的感谢！

### 参　考　文　献


[1] Thanawin R, Bilson C, Abdullah M, et al. Searching and mining trillions of time series subsequences under dynamic time warping//Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. Beijing, China, 2012: 262-270

[2] Begum N, Keogh E J. Rare time series motif discovery from unbounded streams. Proceedings of the VLDB Endowment, 2014, 8(2): 149-160

[3] Petitjean F, Forestier G, Webb G I et al. Dynamic time warping averaging of time series allows faster and more accurate classification//Proceedings of the IEEE International Conference on Data Engineering. Shenzhen, China, 2014: 470-479

[4] Rakthanmanon T, Keogh E J. Data mining a trillion time series subsequences under dynamic time warping//Proceedings of the 23rd International Joint Conference on Artificial Intelligence. Beijing, China, 2013: 3047-3051

[5] Sakoe H, Chiba S. Dynamic programming algorithm optimization for spoken word recognition//Waibel A, Lee K-F eds. Readings in Speech Recognition. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1990: 159-165

[6] Keogh E, Kasetty S. On the need for time series data mining benchmarks: A survey and empirical demonstrations//Proceedings of the 8th International Conference on Knowledge Discovery and Data Mining. Edmonton, Canada, 2003: 349-371

[7] Ratanamahatana C A, Keogh E J. Three myths about dynamic time warping data mining//Proceedings of the SIAM International Conference on Data Mining. Houston, Texas, USA, 2005: 506-510

[8] Adams N, Marquez D, Wakefield G. Iterative deepening for melody alignment and retrieval//Proceedings of the 6th International Conference on Music Information Retrieval. London, UK, 2005: 199-206

[9] Chuitian R, Wei L, Xiaoli W, et al. Efficient and scalable processing of string similarity join. IEEE Transactions on Knowledge and Data Engineering, 2013, 25(10): 2217-2230

[10] Pang Jun, Gu Yu, Xu Jia, Yu Ge. Research progress on similarity join query technology. Computer Science and Exploration, 2013, 7(1): 1-13(in Chinese)
(庞俊，谷峪，许嘉，于戈. 相似性连接查询技术研究进展. 计算机科学与探索，2013，7(1)：1-13)

[11] Kim S, Park S, Chu W. An index-based approach for similarity search supporting time warping in large sequence databases//Proceedings of the 17th International Conference on Data Engineering. Washington, USA, 2001: 607-614

[12] Yi B, Jagadish K, Faloutsos H. Efficient retrieval of similar time sequences under time warping//Proceedings of the 14th International Conference on Data Engineering. Orlando, Florida, 1998: 201-208

[13] Faloutsos C, Lin K. FastMap: A fast algorithm for indexing of traditional and multimedia datasets//Proceedings of the 1995 ACM SIGMOD International Conference on Management of Data. California, USA, 1995: 163-174

[14] Zhou M, Wong M H. Boundary-based lower-bound functions for dynamic time warping and their indexing. Information Sciences, 2001, 181(19): 4175-4196

[15] Lemire D. Faster retrieval with a two-pass dynamic-time-warping lower bound. Pattern Recognition, 2009, 42(9): 2169-2180

[16] Fu A, Keogh E, Lau L, et al. Scaling and time warping in time series querying. The VLDB Journal, 2008, 17(4): 899-921

[17] Sakurai Y, Yoshikawa M, Faloutsos C. FTW: Fast similarity search under the time warping//Proceedings of the 24th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems. New York, USA, 2005: 326-337

[18] Zhu Y, Shasha D. Warping indexes with envelope transforms for query by humming//Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data. California, USA, 2003: 181-192

[19] Papapetrou P, Athitsos V, Potamias M et al. Embedding-based subsequence matching in time-series databases. ACM Transactions on Database Systems, 2011, 36(3): 1-39

[20] Lee H R, Chen C, Jang J R. Approximate lower-bounding functions for the speedup of DTW for melody recognition//Proceedings of the 9th IEEE International Workshop on Cellular Neural Networks and Their Applications. Taoyuan, China, 2005: 178-181

[21] Panagiotis P, Paul D, Vassilis A. Towards faster activity search using embedding-based subsequence matching//Proceedings of the 2nd International Conference on Pervasive Technologies Related to Assistive Environments. Corfu, Greece, 2009: 1-8

[22] Vassilis A, Panagiotis P, Michalis P, et al. Approximate embedding based subsequence matching of time series//Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data. Vancouver, Canada, 2008: 365-378

[23] Xiao C, Wang W, Lin X, Yu J X. Efficient similarity joins for near duplicate detection//Proceedings of the 17th International Conference on World Wide Web. Beijing, China, 2008: 131-140

[24] Wang J, Li G, Feng J. Can we beat the prefix filtering?: An adaptive framework for similarity join and search//Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data. Scottsdale, Arizona, USA, 2012: 85-96

[25] Sarawagi S, Kirpal A. Efficient set joins on similarity predicates//Proceedings of the 2004 ACM SIGMOD International Conference on Management of Data. Paris, France, 2004: 743-754

[26] Arasu A, Ganti V, Kaushik R. Efficient exact set-similarity joins//Proceedings of the 32nd International Conference on Very Large Data Bases. Seoul, Korea, 2006: 918-929

[27] Xiao C, Wang W, Lin X, Shang H. Top-*k* set similarity joins//Proceedings of the 25th International Conference on Data Engineering. Shanghai, China, 2009: 916-927

[28] Vernica R, Carey M J, Li C. Efficient parallel set-similarity joins using MapReduce//Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data. Indianapolis, Indiana, USA, 2010: 495-506

[29] Deng D, Li G, Hao S, et al. MassJoin: A MapReduce-based method for scalable string similarity joins//Proceedings of the 2014 IEEE 30th International Conference on Data Engineering. Chicago, USA, 2014: 340-351

[30] Jiang Y, Li G, Fu J. String similarity joins: An experimental evaluation. Proceedings of the VLDB Endowment, 2014, 7(8): 625-636

[31] Gravano L, Ipeirotis P G, Jagadish H V, et al. Approximate string joins in a database (almost) for free//Proceedings of the 27th International Conference on Very Large Data Bases. San Francisco, USA, 2001: 491-500

[32] Chaudhuri S, Ganti V, Kaushik R. A primitive operator for similarity joins in data cleaning//Proceedings of the 22nd International Conference on Data Engineering, Washington, USA, 2006: 5-16

[33] Arasu A, Chaudhuri S, Kaushik R. Transformation-based framework for record matching//Proceedings of the 24th International Conference on Data Engineering. Cancun, Mexico, 2008: 40-49

[34] Bayardo R J, Ma Y, Srikant R. Scaling up all pairs similarity search//Proceedings of the 16th International Conference on World Wide Web. Banff, Canada, 2007: 131-140

[35] Wang J, Li G, Deng D, et al. Two birds with one stone: An efficient hierarchical framework for Top-*k* and threshold-based string similarity search//Proceedings of the IEEE International Conference on Data Engineering. Seoul, Korea, 2015: 519-530

[36] Deng D, Li G, Feng J. A pivotal prefix based filtering algorithm for string similarity search//Proceedings of the ACM Conference on Management of Data. Snowbird, USA, 2014: 673-684

[37] Brinkhoff T, Kriegel H-P, Seeger B. Efficient processing of spatial joins using R-trees//Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data. Washington, USA, 1993: 237-246

[38] Huang Y W, Jing N, Rundensteiner E A. Spatial joins using R-trees: Breadth-first traversal with global optimizations//Proceedings of the 23rd International Conference on Very Large Data Bases. San Francisco, USA, 1997: 396-405

[39] Lo M L, Ravishankar C V. Spatial hash-joins. ACM Conference on Management of Data, 1996, 25(2): 247-258

[40] Xiang Lian, Lei Chen. Efficient similarity join over multiple stream time series. IEEE Transactions on Knowledge and Data Engineering, 2009, 21(11): 1544-1558

[41] Zhao X, Xiao C, Lin X, et al. Efficient graph similarity joins with edit distance constraints//Proceedings of the 2012 IEEE 28th International Conference on Data Engineering. Washington, USA, 2012: 834-845

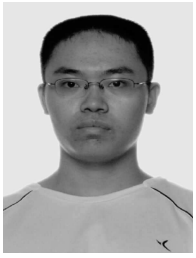
[42] Keogh E, Chakrabarti K, Pazzani M, Mehrotra S. Dimensionality reduction for fast similarity search in large time series databases. Knowledge and Information Systems, 2001, 3(3): 263-286

[43] Keogh E, Chakrabarti K, Pazzani M, Mehrotra S. Locally adaptive dimensionality reduction for indexing large time series databases. ACM Transactions on Database Systems, 2002, 27(2): 188-228

[44] Faloutsos C, Ranganathan M, Manolopoulos Y. Fast subsequence matching in time-series databases//Proceedings of the 1994 ACM SIGMOD International Conference on Management of Data. Minneapolis, Minnesota, USA, 1994: 419-429

[45] Struzik Z R, Siebes A. Wavelet transform in similarity paradigm. Lecture Notes in Computer Science (LNCS). Berlin, Germany: Springer, 1998: 295-309

[46] Peng Jinglin, Wang Hongzhi, Li Jianzhong, Gao Hong. Set-based similarity search for time series//Proceedings of the 2016 International Conference on Management of Data. San Francisco, USA, 2016: 2039-2052



**ZHOU Ning-Nan**, born in 1990, Ph. D. His research interests include database system and data mining.

**ZHANG Xiao**, born in 1972, Ph. D., associate professor. His research interests include database system and big data.

**LIU Cheng-Shan**, born in 1992, M. S. His research interests focus on database.

**WANG Shan**, born in 1944, professor, Ph. D. supervisor. Her research interests include high performance database knowledge engineering and big data.

**Background**

Time series data is pervasive across almost all human endeavors, including medicine, finance, science and entertainment. As such, it is hardly surprising that time series data mining has attracted significant attention and research effort. Most time series data mining algorithms require similarity join as a subroutine, and in spite of the consideration of dozens of alternatives, there is increasing evidence that the classic Dynamic Time Warping (DTW) measure is the best measure in most domains.

Currently, due to the high computationally complexity of DTW, none works has considered the proposed problem in this paper, namely the similarity join on time series under DTW. Works on DTW suffers from high pre-computation overhead when they are applied to similarity join while works on similarity join has not considered the DTW measure. This paper for the first time defines and investigates the novel problem of similarity join on time series under DTW measure and develops two novel upper and lower bounds on DTW to speed up the query evaluation. Real world datasets demonstrate the applications of our proposed problem and extensive experiments on real world and synthetic datasets validate the effectiveness of our solution and the pruning power of our proposed upper and lower bounds.

This work is partially supported by the National Key Research and Development Program (Project No. 2016YFB10007002), and the State Key Program of National Natural Science of China (Grant No. 61432006).