Correlation Set Discovery on Time-series Data

Daichi Amagata and Takahiro Hara

Osaka University, Osaka, Japan
{amagata.daichi, hara}@ist.osaka-u.ac.jp

Abstract. Time-series data analysis is essential in many modern applications, such as financial markets, sensor networks, and data centers, and correlation discovery is a core technique for the analysis. In this paper, we address a novel problem that computes a k-sized time-series dataset where the minimum Pearson correlation of any two time-series in the set is maximized. This problem discovers a group of time-series, which are highly correlated with each other, from a given time-series dataset without any prior knowledge, thus helps many analytical applications. We show that this problem is NP-hard, and design an approximate heuristic solution that provides a high quality result with fast response time. Extensive experiments on real and synthetic datasets verify the efficiency, effectiveness, and scalability of our solution.

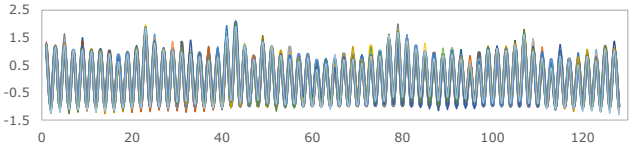
Keywords: Time-series · Correlation set

1 Introduction

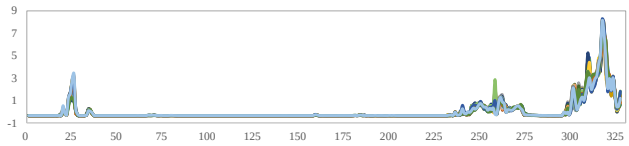
In the IoT era, many data can be represented as time-series, i.e., sequences of data points obtained by successive measurements. Time-series analysis is an important task in many applications, such as financial markets [14] and sensor networks [25]. In this paper, we focus on correlation discovery, which is also known to be an important tool for time-series analysis [6, 17, 20], and address a novel problem of correlation set discovery from a time-series dataset.

For many data mining and discovery tasks, it is interesting to discover an unknown pattern from a given time-series dataset [1, 8, 11], because such a pattern would be a rule and/or feature of the dataset. In this paper, we consider that a set of time-series, which are highly correlated with each other, indicates a pattern. Because of the large size of the dataset, it is infeasible to obtain the set by visual inspection. Efficient extracting such a set from a given time-series dataset is therefore an interesting problem. Besides, if the obtained set size is still large, it may be hard to analyze, which requires that user can limit the result size. Let $\rho(t, t')$ be the Pearson correlation between two time-series t and t'. Given a result size k, a user-specified threshold θ, and a set of time-series data T, our problem is to compute a set $A \subset T$ such that $|A| = k$, for $\forall t, t' \in A$, $\rho(t, t') \geq \theta$, and the minimum $\rho(t, t')$ is maximized.

Our problem can be used in many applications, e.g., pattern (rule) discovery, feature extraction, data exploration, and scientific observation. For example,



(a) Correlated time-series in Google



(b) Correlated time-series in GreenHouseGas

Fig. 1. Time-series datasets provided by our algorithm where $k = 25$ and $\theta = 0.8$

Figure 1 illustrates two sets of z-normalized time-series (identified by our algorithm).

Figure 1(a) illustrates a set of 25 time-series in Google dataset (the CPU rate of each machine in Google compute cells) [19]. To achieve high performance computing in data centers, it is important to take into account the correlation of resource utilization (correlated machines should be located in different servers) [9]. By discovering a correlation group (e.g., Figure 1(a)), administrators can know the machine group that should be divided for performance tuning. This example shows that our problem brings benefits to data center applications.

Environmental analysis is also an application of our problem. It has recently been found that greenhouse gas emissions are spatially correlated (e.g., industrial region) [10]. Investigating how far each emission affects others is also interesting from a scientific viewpoint. Correlation set discovery achieves this by identifying areas where correlated time-series have been observed (e.g., as in Figure 1(b)). This result is also important for policy makers to establish new environmental protection policies in those areas.

Challenge. In fact, this problem is NP-hard, so exact solutions are impractical, suggesting that approximate heuristic approaches are necessary. To design such a heuristic algorithm, we have to address the following challenges.

(1) *High quality result (effectiveness).* Since the optimal result is not obtained practically, a heuristic algorithm needs to have insights that can be used to discover a data space where time-series in the space are correlated. An intuitive approach is to explore such data spaces in offline pre-processing time. However, thresholds θ are normally different for each user, thus pre-processing for specific thresholds does not make sense.

(2) *Computational efficiency.* In the above applications, users may explore a correlation set with varying k and θ. To enable interactive explorations, an algorithm should provide a high quality result with fast response time.

Contributions. We overcome these non-trivial challenges and propose an efficient greedy algorithm. Our contributions are summarized as follows.

- We address the problem of computing a correlation set on time-series data (Section 2). To the best of our knowledge, we are the first to tackle this problem.
- We show that this problem is NP-hard, and propose a heuristic approximate algorithm (Section 3). Our greedy algorithm employs locality-sensitive hashing to obtain an approximate result with fast response time. Theoretical analysis shows that the algorithm has linear scalability with respect to $|T|$, l, and k, where T is the set of time-series and l is the time-series length. This result shows a better performance than that of a baseline which employs existing technique and incurs quadratic time w.r.t. $|T|$.
- The results of our experiments using real and synthetic datasets demonstrate the efficiency, effectiveness, and scalability of our solution (Section 4).

In addition to the above contents, we discuss related work in Section 2, and Section 5 concludes this paper.

2 Preliminary

2.1 Problem definition

A time-series t is described as $t = (t[1], t[2], ..., t[l])$, where $t[i]$ is a real value and l is the length of t. We assume that the length of each time-series in a given dataset T is the same [17] and all time-series in T are z-normalized[1] in advance like the real datasets in UCR time-series data archive[2]. (Note that normalizing time-series by z-normalization is currently common assumption to measure time-series similarity and obtain meaningful results [23, 24, 26].) Let $\|t, t'\|$ be the Euclidean distance between two z-normalized time-series t and t'. The Pearson correlation between t and t', $\rho(t, t')$, is obtained as follows [17].

$$\rho(t, t') = 1 - \frac{\|t, t'\|^2}{2l}$$

We here define correlation set T_θ.

Definition 1 (Correlation set). *Given a threshold θ and a set of time-series data T, a correlation set $T_\theta \subseteq T$ satisfies that $\forall t, t' \in T_\theta$, $\rho(t, t') \geq \theta$.*

The idea for selecting the Pearson correlation as similarity measure is twofold. First, its computational cost is linear to l, i.e., $O(l)$. The representative similarity measures for time-series are the Euclidean distance (which corresponds to the Pearson correlation) and dynamic time warping (DTW) [21]. Unfortunately, DTW incurs $O(l^2)$ time to measure the similarity between two time-series. Second, $\rho(t, t') \in [-1, 1]$, thereby specifying θ is not a difficult task (although DTW does not have such a bound).

It is desirable for users to be able to specify a result size k, in order to obtain a reasonable sized correlation set for easy data exploration and pattern

[1] `https://en.wikipedia.org/wiki/Standard_score`
[2] `https://www.cs.ucr.edu/~eamonn/time_series_data_2018/`

discovery. One of the most interesting correlation set T_θ^* is the one of size k that maximizes the minimum Pearson correlation between time-series in the set, which is formally described as:

$$T_\theta^* = \underset{T_\theta \subseteq T,\, |T_\theta|=k}{\operatorname{argmax}} f(T_\theta) \tag{1}$$

$$f(T_\theta) = \min_{t,t' \in T_\theta} \rho(t, t') \tag{2}$$

where T_θ is a correlation set. If there is no correlation set of size k in a given time-series set T, it is reasonable to provide the correlation set of the largest size, i.e.,

$$T_\theta^* = \underset{T_\theta \subseteq T}{\operatorname{argmax}} |T_\theta|. \tag{3}$$

Ties are broken by selecting the correlation set that maximizes Equation (2). Now we are ready to define the problem in this paper and its hardness.

Definition 2 (Correlation set discovery problem). *Given a set of time-series data T, a result size k, and a threshold θ, this problem is to discover the correlation set A that follows Equation (1) if there is a correlation set of size k in T. Otherwise, this problem is to discover the correlation set A that follows Equation (3).*

Theorem 1 (Hardness). *The correlation set discovery problem is NP-hard.*

Proof. We first assume that there is at least a correlation set of size k in T. We show that our problem corresponds to the k-dispersion problem [18] in this case. The k-dispersion problem is defined as follows: Given a node set $V = \{v_1, v_2, ..., v_{|V|}\}$, this problem is to find a subset V' of V with $|V'| = k$ such that $\min_{v,v' \in V'} dist(v, v')$ is maximized. This problem is shown to be NP-hard. In our problem, each time-series t and the Pearson correlation $\rho(t, t')$ respectively correspond to a node v and $dist(v, v')$. This concludes that computing Equation (1) is NP-hard. Next, we assume that there is no k-sized correlation set in a given T. In this case, we have to compute Equation (3). Consider that a time-series t is a node v and if $\rho(t, t') \geq \theta$, there is an edge between v and v'. Now this problem corresponds to finding the maximum clique in a graph, which is also well known to be NP-hard. Theorem 1 therefore holds. □

Due to Theorem 1, it is not feasible to obtain the optimal answer. Hence, we need to design a heuristic algorithm that can efficiently provide an approximate answer set A with high $f(A)$. Note that it is impossible to know in advance whether or not there is a correlation set of size k in a given T. We therefore focus on designing an algorithm that can obtain a result set A *incrementally* to guarantee that A is a correlation set.

2.2 Related Work

The Pearson correlation is a core similarity function, thereby correlation discovery on time-series data has been extensively studied. Literatures [2, 6, 17, 20, 27] tackled the problem of discovering (all) correlation pairs. Among them, the most

similar to our problem is [17], so we extend the algorithm proposed in [17] for our problem. We compare our algorithm with the extended algorithm, and confirm that computing all correlation pairs does not support efficient correlation set discovery. Our experimental results show that our algorithm significantly outperforms the extended algorithm.

One of other related works is motif discovery. The motif of a given time-series is the most correlated pair of subsequences. Efficient motif discovering algorithms have been proposed for in-memory data [11, 16, 22] and disk-resident data [15]. Matrix profile project, e.g., [12, 23, 17], achieves fast motif discovery. However, these works focus only on a single pair, thereby we do not consider their solutions. This discussion also suggests that our problem is different from finding some similar, e.g., kNN, time-series to a given query time-series [7].

3 Proposed Algorithm

This section presents our proposed algorithm Greedy-L. This algorithm employs a novel approach, i.e., greedy heuristic combined with locality sensitive hashing.

3.1 Greedy Heuristic Framework

First, we introduce the framework of the greedy heuristic, and we use the notations in the proof of Theorem 1. Given k and a node set V, this greedy heuristic computes a result set V' as follows.

1. Insert the pair of nodes (v, v') with the maximum distance into V'.
2. Consider an objective function $f(V', v) = \min_{v' \in V'} dist(v, v')$. Insert the node $v \in V \backslash V'$ into V such that v maximizes $f(V', v)$.
3. Iterate the above operation until $|V'|$ becomes k.

This approach can provide an approximate answer in polynomial time, and existing experimental results show that it provides a high quality result in practice [4]. However, straightforward adaptation of this approach to our problem is not efficient. This is because the first operation needs $O(l|T|^2)$ time and each iteration needs $O(kl|T|)$ time, so the straightforward approach incurs $O(l(|T|^2 + k^2|T|))$ time.

3.2 Locality Sensitive Hashing

The above approach incurs quadratic time cost. *We break this quadratic barrier* by optimizing LSH (locality sensitive hashing) usage. We here define LSH.

Definition 3 (Locality-sensitive hashing). *Given a distance r, an approximate ratio c ($c > 1$), and two probabilities p_1 and p_2 ($p_1 > p_2$), a hash function h is (r, cr, p_1, p_2)-sensitive, if it satisfies the following both conditions:*

- *If* $\|t, t'\| \leq r$, *then* $\Pr[h(t) = h(t')] \geq p_1$;
- *If* $\|t, t'\| \geq cr$, *then* $\Pr[h(t) = h(t')] \leq p_2$.

The LSH function commonly used in the Euclidean space is shown below [3].

$$h(t) = \lfloor \frac{a \cdot t + bw}{w} \rfloor \tag{4}$$

Note that a is a random vector with each dimension independently chosen from the standard normal distribution $\mathcal{N}(0, 1)$, and its length is l. b is a real number randomly chosen from $[0,w)$, and w is a real number that represents the width of h. Recall that we are interested in time-series t and t' satisfying $\rho(t, t') \geq \theta$, thus their hash values should be the same (or very close). To this end, we set

$$w = \sqrt{2l(1-\theta)}. \tag{5}$$

Let $\theta_E = \sqrt{2l(1-\theta)}$, and let $d = \|t, t'\|$. [3] shows that $\Pr[h(t) = h(t')]$ can be obtained as follows:

$$\begin{aligned} p(d) &= \Pr[h(t) = h(t')] \\ &= \int_0^{\theta_E} \frac{1}{d} f_2(\frac{x}{d})(1 - \frac{x}{\theta_E})dx \\ &= 2norm(\frac{\theta_E}{d}) - 1 - \frac{2}{\sqrt{2\pi}} \frac{d}{\theta_E}(1 - e^{-\frac{\theta_E^2}{2d^2}}) \end{aligned} \tag{6}$$

where $f_2(z) = \frac{2}{\sqrt{2\pi}} e^{-\frac{z^2}{2}}$ and $norm(\cdot)$ is the cumulative distribution function of a random variable following $\mathcal{N}(0, 1)$. Note that $h(t)$ has the following lemma [5].

LEMMA 1. *The LSH obtained from Equation (4) is* (θ_E, $c\theta_E$, $p(\theta_E)$, $p(c\theta_E)$)*-sensitive.*

Because $h(\cdot)$ provides the same (or similar) hash values if two time-series are very similar, it is intuitive that we do not need to compare two time-series with totally different hash values. However, it is important to note that using a single $h(\cdot)$ cannot avoid unnecessary computation well, because many time-series with far distance (i.e., low Pearson correlation) may have the same hash values. To avoid this, a compound LSH function $G(t) = (h_1(t), h_2(t), ..., h_m(t))$ is employed, where each component of $G(t)$ is $h(t)$ and independently generated [3]. We consider that $G(t)$ is a key of t, and two time-series with high Pearson correlation would have the same or similar keys.

It is important to note that existing studies utilize LSH as indices, i.e., offline processing, but we utilize LSH for online processing to deal with arbitrary θ, see Equation (5).

3.3 Main Techniques

Assume that each time-series $t \in T$ is assigned its key K and is inserted into the bucket with key K, B_K. One may consider the following simple combination of the greedy heuristic and LSH. We compute a pair of two time-series $\langle t_i, t_j \rangle$, which is firstly added to A, by using LSH. In other words, if we compute the pair

with the highest Pearson correlation for $\forall B_K \in B$, where B is the set of buckets, we can obtain $\langle t_i, t_j \rangle$. Then we compute $t^* = \text{argmax}_{t \in T \setminus A} f(A, t)$, where

$$f(A, t) = \min_{t' \in A} \rho(t, t'),$$

by scanning T, and t^* is inserted into A. This operation is iterated until $|A|$ becomes k.

Although this seems to reduce computational cost, it is not sufficient. In each iteration, we compute t^* *based on the intermediate* A, so the pair $\langle t_i, t_j \rangle$, which is firstly added to A, has a large influence on the final quality and size of A. Due to this property, $\langle t_i, t_j \rangle$ has to satisfy the following requirements.

- $\rho(t_i, t_j)$ is high as much as possible: Because $f(A)$, which is described in Equation (2), has submodularity, i.e., $f(A) \geq f(A \cup \{t\})$, the first pair should have high Pearson correlation. Otherwise, the quality of the final result becomes low.
- $\langle t_i, t_j \rangle$ exists in a large group of time-series which are correlated with each other: This requirement is necessary to provide A such that $|A| = k$.

We below elaborate how to discover such a pair. Assume that each time-series t in T is assigned a key K by $G(t)$. For each bucket $B_K \in B$, we compute the highest Pearson correlation in B_K denoted by ρ_K. Recall that higher ρ_K is better due to the submodularity of $f(A)$. We next consider the size of the *adjacent buckets* which are defined below.

Definition 4 (Adjacent bucket). *Given a set of buckets B and a bucket $B_K \in B$, each bucket $B_{K'}$ which is an adjacent bucket of B_K, satisfies that $|\{i \,|\, h_i^K = h_i^{K'}\}| = m - 1$ where h_i^K ($h_i^{K'}$) is the i-th hash value of K (K').*

Let B_θ be the set of buckets B_K such that $\rho_K \geq \theta$. We retrieve the adjacent buckets of B_K in B_θ and compute s_K which is the summation of their sizes (the number of time-series in the buckets) and $|B_K|$. More formally,

$$s_K = |B_K| + \sum |B_{K'}|,$$

where $B_{K'}$ is an adjacent bucket of B_K. Recall that time-series in the same bucket or buckets with similar keys tend to be correlated. Therefore, if s_K is large, time-series in B_K would exist in a large group of time-series which are correlated with each other. Based on the above idea, we select the pair of two time-series with the highest Pearson correlation in B_K where $\rho_K \cdot \frac{s_K}{|T|}$ is the maximum among B_θ. (Because $\rho_K \in [\theta, 1]$, s_K has to be normalized and $|T|$ is used to achieve this.) The complexity of this operation is as follows.

Lemma 2. *We can select the first two time-series with $O(ml|T|)$ time.*

Proof. Computing $G(\cdot)$ for each time-series needs $O(ml)$ time, thus the hashing incurs $O(ml|T|)$ time. Let β be the number of buckets $\in B$ where $|B_K| \geq 2$, and let n be the average number of time-series in B_K. To obtain $\langle t, t' \rangle$, we need $O(\beta n^2)$. However, by setting a sufficiently large constant as m, n can be very small, so we have $O(\beta n^2) \ll O(ml|T|)$. We can compute the first two time-series

by scanning B_θ, and $|B_\theta| \leq |T|$. Then, we can conclude that the time complexity is $O(ml|T|)$. □

Next, we consider how to efficiently find a time-series which has high Pearson correlation with each time-series in an intermediate result A. Our idea is simple yet effective. Because two time-series with high Pearson correlation share the same or similar key, promising time-series, which can be the next result t^*, exist in the adjacent buckets of the buckets in which the time-series $\in A$ exist. We compute t^* from the set of the buckets denoted by S, and its time complexity is $O(l|S|)$.

LEMMA 3. *We can obtain* $t^* = \mathrm{argmax}_{t \in S \backslash A} f(A, t)$ *with* $O(l|S|)$ *time.*

PROOF. Assume that a time-series t is in A and $A = \{t\}$, and for $\forall t_i \in S \backslash \{t\}$, we compute $\rho(t, t_i)$. Assume further that $t^* = t'$ and each time-series t_i caches $f(A, t)$. When we find the next t^*, we can obtain the exact $f(A, t_i)$ of a given $t_i \in S \backslash A$ by comparing $\rho(t', t_i)$ with the cached value, which needs only $O(l)$ time. Thus we can obtain t^* with $O(l|S|)$ time. □

Besides, a lower-bound of the existing probability of a time-series t, which satisfies that $f(A, t) \geq \theta$, in S is obtained as follows.

LEMMA 4. *We have*

$$\Pr[t \in S, f(A, t) \geq \theta] \geq p(\theta_E)^m + p(\theta_E)^{m-1}(1 - p(\theta_E))m.$$

PROOF. From Equation (6) and Definition 4. □

3.4 Algorithm Description

Algorithm 1 details Greedy-L. Greedy-L first obtains the key of each time-series (lines 1–2). Then Greedy-L computes $\langle t, t' \rangle$, where $t, t' \in B_K$ and $\rho(t, t') \cdot \frac{s_K}{|T|}$ is the maximum in B_θ (lines 3–17). The pair $\langle t, t' \rangle$ is inserted into A. Greedy-L retrieves the next result t^* from S which is the union of B_K such that $t, t' \in B_K$ and the adjacent buckets of B_K. Also, for each iteration (lines 20–26), after Greedy-L inserts $t^* = \mathrm{argmax}_{t \in S \backslash A} f(A, t)$ into A, B_K, where $t^* \in B_K$, and its adjacent buckets are inserted into S (line 24). This is repeated until $|A|$ becomes k or Greedy-L identifies that $\nexists t \in S$ such that $f(A, t) \geq \theta$.

Now we show our main result: the time complexity of Greedy-L is linear to each parameter and breaks the quadratic barrier.

THEOREM 2. *Greedy-L needs* $O(ml|T| + kl|S|)$ *time to provide* A*, where* $S \subseteq T$.

PROOF. From Lemma 2, lines 1–17 need $O(ml|T|)$ time. To obtain s_K, we need to find the adjacent buckets of B_K. We cache the value range of each LSH h_i, and z-normalization provides the fact that the range is very small as shown in Figure 1. Thus s_K is obtained by $O(m)$ time, i.e., lines 14–17 incurs $O(m|B_\theta|)$ time, and $O(m|B_\theta|) \ll O(ml|T|)$. Lines 19, 21, and 24 respectively need $O(l|S|)$ time. As a result, the time complexity of Greedy-L is $O(ml|T| + kl|S|)$. □

Discussion. We exploit the adjacent buckets to effectively select buckets for the candidates of the result. One may consider about employing *near* buckets that

Algorithm 1: Greedy-L

1 **for** $\forall t \in T$ **do**
2 $\quad B_K \leftarrow B_K \cup \{t\}$ where $K = (h_1(t), h_2(t), ..., h_m(t))$
3 $B_\theta \leftarrow \varnothing$, $P \leftarrow \varnothing$
4 **for** $\forall B_K \in B$ *where* $|B_K| \geq 2$ **do**
5 $\quad t_K, t'_K \leftarrow \varnothing$, $\rho_K \leftarrow -1$
6 $\quad$ **for** $\forall t_i \in B_K$ **do**
7 $\quad\quad$ **for** $\forall t_j \in B_K$ **do**
8 $\quad\quad\quad$ **if** $\rho_K < \rho(t_i, t_j)$ **then**
9 $\quad\quad\quad\quad \rho_K \leftarrow \rho(t_i, t_j)$, $\langle t_K, t'_K \rangle \leftarrow \langle t_i, t_j \rangle$
10 $\quad$ **if** $\rho_K \geq \theta$ **then**
11 $\quad\quad B_\theta \leftarrow B_\theta \cup B_K$
12 $\quad\quad P \leftarrow P \cup \langle \rho_K, t_K, t'_K \rangle$
13 $t, t' \leftarrow \varnothing$, $\mu = 0$
14 **for** $\forall B_K \in B_\theta$ **do**
15 $\quad s_K \leftarrow |B_K| + \sum |B_{K'}|$ where $B_{K'} \in B_\theta$ is the nearest bucket of B_K
16 $\quad$ **if** $\rho_K \cdot \frac{s_K}{|T|} > \mu$ **then**
17 $\quad\quad \mu \leftarrow \rho_K \cdot \frac{s_K}{|T|}$, $\langle t, t' \rangle \leftarrow \langle t_K, t'_K \rangle$
18 $A \leftarrow \langle t, t' \rangle$
19 $S \leftarrow B_K \cup B^N_{K'}$ where $t, t' \in B_K$ and $B^N_{K'}$ is the set of the nearest bucket of B_K in B
20 **while** $|A| < k$ **do**
21 $\quad t^* \leftarrow \underset{t \in S \setminus A}{\text{argmax}}\ f(A, t)$
22 $\quad$ **if** $f(A, t^*) \geq \theta$ **then**
23 $\quad\quad A \leftarrow A \cup \{t^*\}$
24 $\quad\quad S \leftarrow S \cup B_K \cup B^N_{K'}$ where $t^* \in B_K$ and $B^N_{K'}$ follows line 19
25 $\quad$ **else**
26 $\quad\quad$ **break**

share $(m - m')$ LSHs with a given bucket. If we employ this, Greedy-L loses its efficiency significantly due to large increase of S (the number of near buckets of a given bucket is $\binom{m}{m'}$). Besides, specifying an appropriate m' is not trivial. Greedy-L therefore employs the adjacent buckets.

We next show that Greedy-L is a parallel-friendly framework. Recall that each LSH in $G(\cdot)$ is independently generated. This computation can be parallelized. Also, it can be seen that computing ρ_K, s_K, and t^* is parallelized by dividing B, B_θ, and $|S|$ into some pieces.

4 Experiments

We present our empirical study that evaluates the performance of Greedy-L.

4.1 Setting

Datasets. In our experiments, we used two real datasets and a synthetic dataset introduced below.

- GreenHouseGas [13]: This dataset has 46,736 time-series, and each time-series consists of 327 green house gas concentrations.
- Google: This dataset consists of 10,380 time-series (CPU rates of machines in Google compute cells) with length 128.
- Rand: This dataset is generated by a random walk technique. When generating a time-series t, we randomly choose the first value ($t[1]$) in $\{-1, 1\}$. The subsequent value is generated by $t[i+1] = t[i] + \mathcal{N}(0, 1)$ [16]. We set $|T| = 100,000$ and $l = 1,000$ by default.

(We conducted experiments on other datasets but omit their results because they are consistent.) When we use a dataset, all time-series in the dataset are memory-resident.

Algorithms. We evaluated the following algorithms.

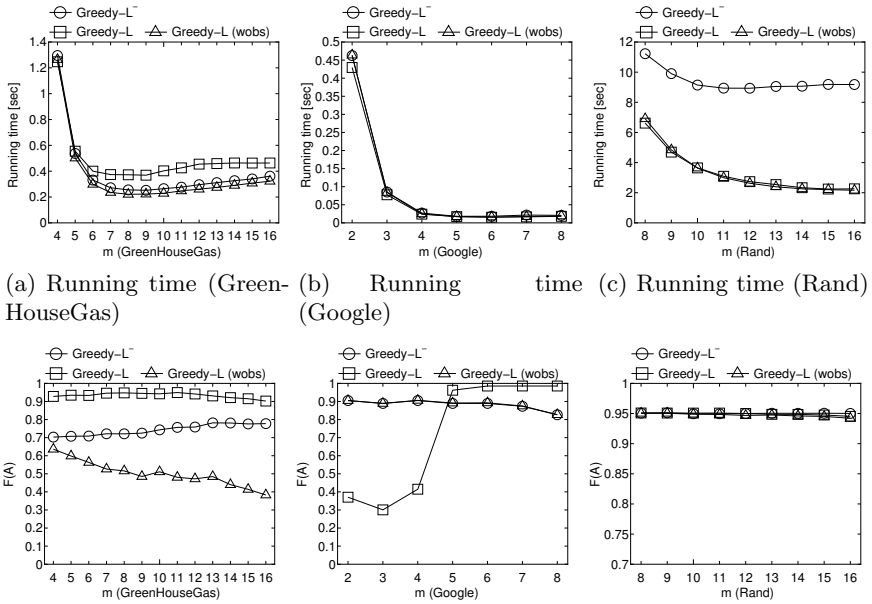
- Greedy-M: this is an extended version of [17], which is a state-of-the-art online algorithm to compute all time-series pairs whose Pearson correlation satisfies θ. Greedy-M employs this technique and the greedy heuristic introduced in the beginning of the proposed algorithm section, to compute A.
- Greedy-L: the proposed algorithm in this paper.
- Greedy-L$^-$: this algorithm utilizes LSH only to obtain the first two time-series. The greedy heuristic is also employed in each iteration to update the result set.
- Greedy-L (wobs): this algorithm normally executes the same operations as those in Greedy-L, but selects the first bucket B_K such that ρ_K is the highest among B.

All algorithms were implemented in C++, and all experiments were conducted on a PC with Intel Xeon E5-2687W v4 processors (3.0GHz) and 512GB RAM.

Criteria. We measured the average of each metric introduced below. We run the algorithms 50 times for each experiment.

- Running time (efficiency). This metric is defined as the time to provide a correlation set A.
- $F(A) = f(A) \cdot \frac{|A|}{k}$ (effectiveness). Although the above algorithms guarantee that A is a correlation set, they do not guarantee that $|A| = k$. It is unfair to compare algorithms based on $f(A)$, since the algorithms may provide different result size. We therefore normalize $f(A)$ by $\frac{|A|}{k}$.

Recall that, as shown in Theorem 1, the exact answer A^* is not obtained practically, so comparing $F(A)$, where A is provided by our solution, with $F(A^*)$ is impossible.



(a) Running time (GreenHouseGas) (b) Running time (Google) (c) Running time (Rand)

(d) $F(A)$ (GreenHouseGas) (e) $F(A)$ (Google) (f) $F(A)$ (Rand)

Fig. 2. Impact of m

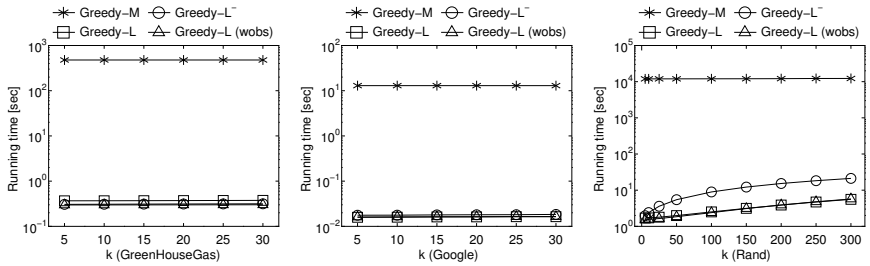
Table 1. Tuning m for each algorithm

Algorithm	GreenHouseGas	Google	Rand
Greedy-L	8	6	13
Greedy-L$^-$	13	5	11
Greedy-L (wobs)	6	4	13

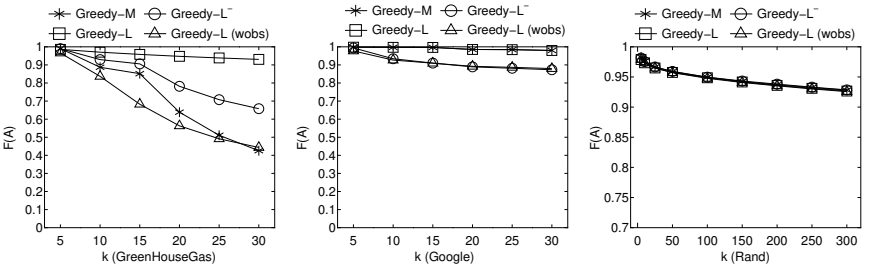
4.2 Result

By default, $\theta = 0.8$, $k = 20$ in the cases of GreenHouseGas and Google, and $k = 100$ in the case of Rand.

Varying m. We first tune m (the number of $h(\cdot)$ in the compound LSH function) of Greedy-L$^-$, Greedy-L, and Greedy-L (wobs) for each dataset by using the default parameter setting. Figure 2 illustrates the impact of m. We can see that m affects the performances of the three algorithms. For example, when m is small, there is a large number of time-series in the same bucket, so computing the first two time-series which will be in A needs long time. Figures 2(d) and 2(f) show that Greedy-L$^-$ and Greedy-L provide stable $F(A)$ (but Greedy-L (wobs) does not). On the other hand, Figure 2(e) shows that Greedy-L provides bad result quality when m is small. When m is small, there are many non-correlated time-series in the same bucket. In this case, s_K cannot reflect data distribution. Based on the result, we set m as shown in Table 1. ($F(A)$ is prioritized.)



(a) Run time (GreenHouseGas) (b) Run time (Google) (c) Run time (Rand)



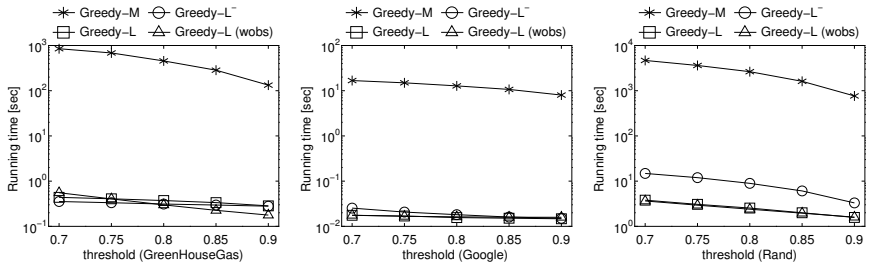
(d) $F(A)$ (GreenHouseGas) (e) $F(A)$ (Google) (f) $F(A)$ (Rand)

Fig. 3. Impact of k

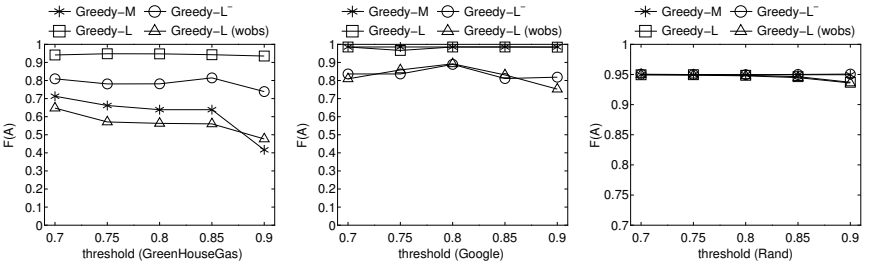
Varying k. Figures 3(a) and 3(b) show that all the algorithms are not affected by k. Since Greedy-M incurs the overhead from computing all pairs Pearson correlation, i.e., $O(l|T|^2)$, it is reasonable. The other algorithms have two main computational overheads: hashing and iteration. When k is small, hashing becomes a dominant factor, thereby the result is obtained. When k is large, on the other hand, the running time of the algorithms except Greedy-M increases as shown in Figure 3(c). We see that Greedy-L scales better than Greedy-L$^-$, because Greedy-L$^-$ scans the whole dataset in each iteration. Note that Greedy-L runs up to 1,500 times faster than Greedy-M.

Let us focus on result quality, and Figures 3(d) and 3(e) show that Greedy-L provides the best result among the four algorithms. (Because Rand has many correlated time-series, the four algorithms provide almost the same result, as shown in Figure 3(f).) In particular, Greedy-M, Greedy-L$^-$, and Greedy-L (wobs) fail to return a good result in the case of GreenHouseGas. In this dataset, the pair of two time-series with the highest Pearson correlation exists in a very small group. The three algorithm (often) return this set, but Greedy-L can avoid this situation and provides a larger group by exploiting LSH, which verifies the effectiveness of our approach. (Recall that the result obtained by Greedy-L is illustrated in Figure 1.)

Varying θ. Figure 4 shows the impact of threshold. As shown in Figures 4(a)–4(c), as θ increases, running time of each algorithm decreases. Even in this case,

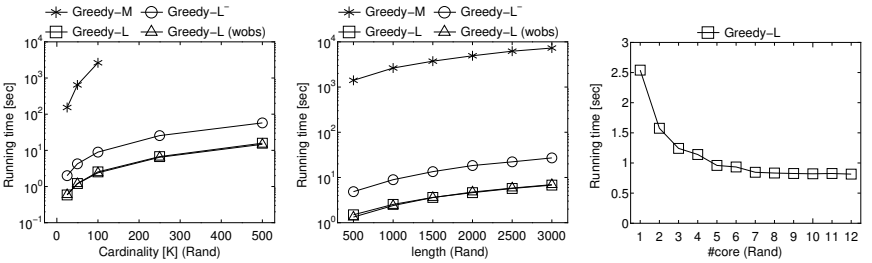


(a) Running time (GreenHouseGas) (b) Running time (Google) (c) Running time (Rand)



(d) $F(A)$ (GreenHouseGas) (e) $F(A)$ (Google) (f) $F(A)$ (Rand)

Fig. 4. Impact of θ



(a) Varying $|T|$ (b) Varying l (c) Varying the number of cores

Fig. 5. Scalability test

Greedy-M is very slow and the other algorithms keep outperforming Greedy-M significantly.

Figures 4(d) and 4(e) show that the Greedy-L (wobs) often returns a worse result than the other algorithms. As well as Greedy-L, Greedy-L (wobs) finds the next result (i.e., t^*) only from a subset of T, and the subset is also dependent on the first two time-series of A. This result implies that ignoring s_K misses identifying a group of time-series, and Greedy-L (wobs) cannot be robust.

Varying $|T|$, l, and the number of cores. We also investigate the scalability to the size of a given dataset,the length of a time-series, and the number of available CPU cores by using Rand. (We used OpenMP to support parallelization.) The results are respectively shown in Figures 5(a), 5(b), and 5(c). (We omit the results of $F(A)$ because they are almost consistent like Figure 4(f).) Recall that the time complexity of Greedy-M is $O(|T|^2 l)$, so its running time is significantly large, which is shown in Figure 5(a) (we omit the result of Greedy-M in the cases of $|T| = 250,000$ and $|T| = 500,000$). Since the time complexities of the other algorithms are linear to $|T|$, the experimental results follow this fact. Impact of l also has this case.

Figure 5(c) shows that Greedy-L reduces its running time with increase of available cores. For example, by using 8 cores, its running time becomes approximately 3 times faster than the case of using only 1 core.

Remark. As Theorem 2 also argues, Greedy-L significantly outperforms the approach using existing techniques. In addition, Greedy-L provides a high quality result, i.e., A with high $f(A)$, in practice, meaning that Greedy-L satisfies the two important requirements, effectiveness and efficiency.

5 Conclusion

In this paper, we addressed a novel problem of discovering a correlation set on time-series data. We showed that this problem is NP-hard, and proposed an efficient greedy heuristic algorithm, Greedy-L. Greedy-L employs locality sensitive hashing to reduce running time. In particular, we devised a novel technique that exploits locality-sensitive hashing to discover a large group of time-series which are correlated with each other. The experimental results demonstrate the efficiency, effectiveness, and scalability.

Acknowledgment. This research is partially supported by JSPS Grant-in-Aid for Scientific Research (A) Grant Number 18H04095, JSPS Grant-in-Aid for Young Scientists (B) Grant Number JP16K16056, and JST CREST Grant Number J181401085.

References

1. Amagata, D., Hara, T.: Mining top-k co-occurrence patterns across multiple streams. TKDE **29**(10), 2249–2262 (2017)
2. Cole, R., Shasha, D., Zhao, X.: Fast window correlations over uncooperative time series. In: KDD. pp. 743–749 (2005)
3. Datar, M., Immorlica, N., Indyk, P., Mirrokni, V.S.: Locality-sensitive hashing scheme based on p-stable distributions. In: SoCG. pp. 253–262 (2004)
4. Drosou, M., Pitoura, E.: Diversity over continuous data. IEEE Data Eng. Bull. **32**(4), 49–56 (2009)
5. Gan, J., Feng, J., Fang, Q., Ng, W.: Locality-sensitive hashing scheme based on dynamic collision counting. In: SIGMOD. pp. 541–552 (2012)

6. Guo, T., Sathe, S., Aberer, K.: Fast distributed correlation discovery over streaming time-series data. In: CIKM. pp. 1161–1170 (2015)
7. Huang, Q., Feng, J., Zhang, Y., Fang, Q., Ng, W.: Query-aware locality-sensitive hashing for approximate nearest neighbor search. PVLDB **9**(1), 1–12 (2015)
8. Kato, S., Amagata, D., Nishio, S., Hara, T.: Monitoring range motif on streaming time-series. In: DEXA. pp. 251–266 (2018)
9. Kim, J., Ruggiero, M., Atienza, D., Lederberger, M.: Correlation-aware virtual machine allocation for energy-efficient datacenters. In: DATE. pp. 1345–1350 (2013)
10. Li, L., Hong, X., Tang, D., Na, M.: Ghg emissions, economic growth and urbanization: A spatial approach. Sustainability **8**(5), 462 (2016)
11. Li, Y., Yiu, M.L., Gong, Z., et al.: Quick-motif: An efficient and scalable framework for exact motif discovery. In: ICDE. pp. 579–590 (2015)
12. Linardi, M., Zhu, Y., Palpanas, T., Keogh, E.: Matrix profile x: Valmod-scalable discovery of variable-length motifs in data series. In: SIGMOD. pp. 1053–1066 (2018)
13. Lucas, D., Kwok, C.Y., Cameron-Smith, P., Graven, H., Bergmann, D., Guilderson, T., Weiss, R., Keeling, R.: Designing optimal greenhouse gas observing networks that consider performance and cost. Geoscientific Instrumentation, Methods and Data Systems **4**(1), 121 (2015)
14. Marti, G., Andler, S., Nielsen, F., Donnat, P.: Clustering financial time series: How long is enough? In: IJCAI. pp. 2583–2589 (2016)
15. Mueen, A., Keogh, E., Bigdely-Shamlo, N.: Finding time series motifs in disk-resident data. In: ICDM. pp. 367–376 (2009)
16. Mueen, A., Keogh, E., Zhu, Q., Cash, S., Westover, B.: Exact discovery of time series motifs. In: SDM. pp. 473–484 (2009)
17. Mueen, A., Nath, S., Liu, J.: Fast approximate correlation for massive time-series data. In: SIGMOD. pp. 171–182 (2010)
18. Ravi, S., Rosenkrantz, D.J., Tayi, G.K.: Facility dispersion problems: Heuristics and special cases. In: Algorithms and Data Structures, pp. 355–366 (1991)
19. Reiss, C., Wilkes, J., Hellerstein, J.L.: Google cluster-usage traces: format+ schema. Google Inc., White Paper pp. 1–14 (2011)
20. Tsytsarau, M., Amer-Yahia, S., Palpanas, T.: Efficient sentiment correlation for large-scale demographics. In: SIGMOD. pp. 253–264 (2013)
21. Vlachos, M., Hadjieleftheriou, M., Gunopulos, D., Keogh, E.: Indexing multidimensional time-series. The VLDB Journal **15**(1), 1–20 (2006)
22. Yankov, D., Keogh, E., Medina, J., Chiu, B., Zordan, V.: Detecting time series motifs under uniform scaling. In: KDD. pp. 844–853 (2007)
23. Yeh, C.C.M., Kavantzas, N., Keogh, E.: Matrix profile vi: Meaningful multidimensional motif discovery. In: ICDM. pp. 565–574 (2017)
24. Yeh, C.C.M., Zhu, Y., Ulanova, L., Begum, N., Ding, Y., Dau, H.A., Silva, D.F., Mueen, A., Keogh, E.: Matrix profile i: All pairs similarity joins for time series: A unifying view that includes motifs, discords and shapelets. In: ICDM. pp. 1317–1322 (2016)
25. Yi, X., Zheng, Y., Zhang, J., Li, T.: St-mvl: Filling missing values in geo-sensory time series data. In: IJCAI. pp. 2704–2710 (2016)
26. Zhu, Y., Zimmerman, Z., Senobari, N.S., Yeh, C.C.M., Funning, G., Mueen, A., Brisk, P., Keogh, E.: Matrix profile ii: Exploiting a novel algorithm and gpus to break the one hundred million barrier for time series motifs and joins. In: ICDM. pp. 739–748 (2016)
27. Zhu, Y., Shasha, D.: Statstream: Statistical monitoring of thousands of data streams in real time. In: VLDB. pp. 358–369 (2002)