Finding and Visualizing Time Series Motifs of All Lengths using the Matrix Profile

(blinded)

***Abstract*— Many time series analytic tasks can be reduced to discovering and then reasoning about conserved structures, or *time series motifs*. Recently, the Matrix Profile has emerged as the state-of-the-art for finding time series motifs, allowing the community to efficiently find time series motifs in large datasets. The matrix profile reduced time series motif discovery to a process requiring a single parameter, the length of time series motifs we expect (or wish) to find. In many cases this is a reasonable limitation as the user may utilize out-of-band information or domain knowledge to set this parameter. However, in truly exploratory data mining, a poor choice of this parameter can result in failing to find unexpected and exploitable regularities in the data. In this work, we introduce the Pan Matrix Profile, a new data structure which contains the nearest neighbor information for *all* subsequences of *all* lengths. This data structure allows the first truly parameter-free motif discovery algorithm in the literature. The sheer volume of information produced by our representation may be overwhelming; thus, we also introduce a novel visualization tool called the motif-heatmap which allows the users to discover and reason about repeated structures at a glance. We demonstrate our ideas on a diverse set of domains including seismology, bioinformatics, transportation and biology.**

Keywords—Time series, Motif discovery, Anomaly detection

I. INTRODUCTION

In recent years, the Matrix Profile (MP) has emerged as a promising data structure to support time series data mining. The MP is a simple data structure that contains the nearest neighbor information for all subsequences of a given length in a time series. In the past three years it has been shown that the MP can be used to facilitate the discovery of motifs [21], discords (anomalies) [25], chains (evolving patterns) [24], shapelets [21], snippets [7], regimes [25], and more. However, we argue that the MP has a strong assumption that limits it practicality by requiring the user to specify the subsequence length ahead of time. A data scientist may have a good intuition as to what this subsequence length should be, based on their experience or a first principles model of the system being examined. However, in many cases, particularly for exploratory data mining, the user may have no idea as to the subsequence lengths at which patterns are conserved in the data necessitating the need for *variable-length* motif discovery.

Consider the one-hundred second excerpt of an EOG (Electro-oculogram; the movement of an eye) dataset from a 66-year old healthy male recorded during a sleep study show in Fig. 1. Here we are tasked with identifying regions corresponding to the "blinking of the eye" in an attempt to remove these regions from a companion EEG dataset (not shown). Because eye blinks are not only unique to the individual but also sensitive to the sensor placement, we cannot use a single "one-size-fits-all" template. But, given that blinks are typically well-conserved, at least during a single sleep session, we can perform motif discovery to identify an appropriate template. However, the suggested subsequence length for motif discover is not readily apparent. We may attempt to rely on the current sleep study literature in which case [16] suggests "*a duration of 1.5 to 2.5 seconds*" as the subsequence length. In Fig 1.*bottom.left* we show that using 2.5 seconds does indeed discover a highly conserved motif that corresponds to an eye blink. Moreover, searching for more examples of this pattern in the full night of sleep data, we find hundreds of additional examples of this shape.

A sleep technician might very well be justified in terminating her search. However, as Fig 1.*bottom.right* shows, this time series has a second type of eye-blink artifact with a subsequence length of five seconds which may not have been considered by the sleep technician due to the high frequency of the 2.5 second motif. The fact that eye blinks can be polymorphic seems underappreciated, but [1] cautions EOG signals can have " *more than one category… classified by shape.*" Missing this second blink artifact would have drastically corrupted the downstream analytics performed on this dataset.

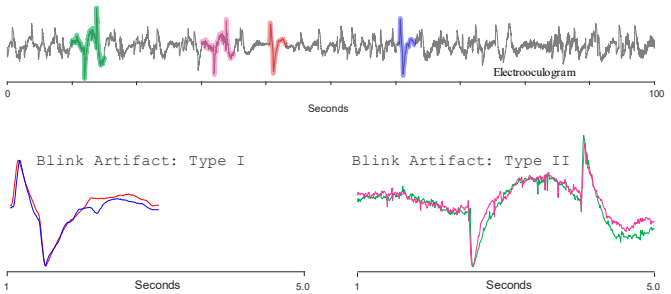

Fig. 1. *top*) One hundred seconds of EOG data. *bottom.left*) A search for the top 2.5-second motif reveals a highly conserved pattern eye-blink-artifact. *bottom.right*) A search for the top 5-second motif reveals another highly conserved pattern, that has no overlap with the first.

This very basic problem exemplified in an EOG dataset is ubiquitous in nearly all domains as the user's choice limits what regularities can be found in the dataset. In the EOG dataset shown in Fig. 1, the two motif patterns differ by a factor of two; however, as this paper will show, other datasets may contain motifs which can differ by up to two orders of magnitude.

A possible, yet inelegant, solution to this problem is trial and error over different lengths. Beyond being frustratingly time consuming and awkward for the user, there is still a real danger of missing an interesting pattern. Though the definition of a time series motif is fairly robust to minor changes in length [13][14][25], there *will* be some length at which there is a "phase change", that is, the location of the motif will "jump" to a different place in the time series. For example, in Section III.A

we introduce a dataset with a maximal motif for $m_1 = 68$, and a non-related/nonoverlapping maximal motif for $m_2 = 610$.

In this work we solve this motif-length sensitivity problem by introducing the Pan Matrix Profile (PMP), a data structure that contains *all* MP information of a time series with length n for all lengths in a fixed range r. In addition, we introduce SKIMP (**S**calable **KI**netoscopic[1] **M**atrix **P**rofile), an algorithm to compute the PMP with time complexity $O(n^2 r)$ and space complexity $O(nr)$. Though untenable for large datasets which require an *exact* solution, SKIMP is computed in an *anytime* fashion allowing for fast *approximate* solutions [26]. In almost all cases, running SKIMP to even one one-hundredth of its full convergence time will produce results that are almost indistinguishable from the final product.

Using SKIMP, we believe that all algorithms that exploit the MP could be made length-agnostic, that is to say, we can have length-agnostic chains [24], snippets, regimes, etc. However, for clarity and concreteness, in this work we confine our claims to motif and anomaly discovery and leave all other considerations for future work.

The practical application of SKIMP is in facilitating interactive time series analytics on practical problems in bioinformatics, seismology, medicine and industry.

The rest of this paper is organized as follows. In Section II we introduce the relevant notation, background material, and define the PMP data structure. Section III introduces a family of algorithms to compute the PMP and several algorithms to exploit and visualize it. We conduct an extensive empirical evaluation in Section IV. We defer a discussion of related work to V, so the readers intuitions for the issues at hand are more fully developed, before offering conclusions and directions for future work in Section VI.

II. Notation and Background

A. Time Series Notation

We begin by introducing all the necessary definitions, starting with the data type of interest, *time series*:

Definition 1: A time series T is a sequence of real-valued numbers t_i: $T = t_1, t_2, ..., t_n$ where n is the length of T:

We are typically interested not in *global*, but *local* properties of a time series. A local region of a time series is called a *subsequence*:

Definition 2: A subsequence $T_{i,m}$ of a time series T is a continuous subset of the values from T of length m starting from position i. Formally, $T_{i,m} = t_i, t_{i+1}, \ldots, t_{i+m-1}$, where $1 \leq i \leq n\text{-}m+1$.

Given a query subsequence $T_{i,m}$ and a time series T, we can compute the distance between $T_{i,m}$ and *all* the subsequences in T with length m. We call this a *distance profile*:

Definition 3: A *distance profile* D_i corresponding to query $T_{i,m}$ and time series T is a vector of the Pearson correlation between a given query subsequence $T_{i,m}$ and each subsequence in time series T with length m. Formally, $D_i = [d_{i,1}, d_{i,2}, ..., d_{i,n-m+1}]$, where $d_{i,j}$ ($1 \leq j \leq n\text{-}m+1$) is the distance between $T_{i,m}$ and $T_{j,m}$.

We assume that the distance is measured by Euclidean distance between z-normalized subsequences [25][21]. Once we obtain D_i, we can extract the nearest neighbor of $T_{i,m}$ in T. Note that if the query $T_{i,m}$ is a subsequence of T, the i^{th} location of distance profile D_i is zero (i.e., $d_{i,i} = 0$) and close to zero just to the left and right of i. This is called a *trivial match* in the literature (See Definition 7). Most of the community follow the suggestion in [4] to avoid such matches by ignoring an "exclusion" zone of length $m/2$ before and after i, the location of the query [21].

We wish to find the nearest neighbor of every subsequence in T. The nearest neighbor information for subsequences with length m is stored in two meta time series, the *matrix profile,* and the *matrix profile index*:

Definition 4: A *matrix profile* P of time series T is a vector of the Euclidean distances between every subsequence of $T_{i,m}$ and its nearest neighbor $T_{j,m}$ in T. Formally, $P_m = [\min(D_1), \min(D_2), ..., \min(D_{n-m+1})]$, where D_i ($1 \leq i \leq n\text{-}m+1$) is the distance profile D_i corresponding to query $T_{i,m}$ and time series T.

The i-th element in the matrix profile P tells us the Euclidean Distance from subsequence $T_{i,m}$ to its nearest neighbor in time series T. However, it does not tell us the *location* of that nearest neighbor; this is stored in the companion *matrix profile index*:

Definition 5: A *matrix profile index* I of time series T is a vector of integers: $I=[I_1, I_2, \ldots I_{n-m+1}]$, where $I_i=j$ if $d_{i,j} = \min(D_i)$.

Fig. 2 illustrates the relationship between distance matrix, distance profile (Definition 3) and matrix profile (Definition 4). Each element of the distance matrix $d_{i,j}$ is the distance between $T_{i,m}$ and $T_{j,m}$ for $1 \leq i$ and $j \leq n - m + 1$ of time series T.

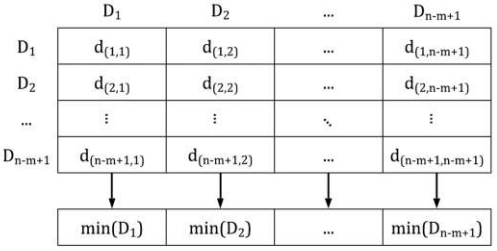

Fig. 2. The relationship between the distance matrix, distance profile, and matrix profile. A distance profile D_i is a column (also a row) of the distance matrix. The matrix profile stores the minimum (off diagonal) value of each column of the distance matrix; the location of the minimum value within each column is stored in the companion matrix profile index.

Fig. 3 shows a visual example of a distance profile and a matrix profile created from the same time series T. Note that as we presented it above, the matrix profile uses the z-normalized Euclidean distance [21]. However, this is logically equivalent to the Pearson correlation, and we can convert between them with ease. Some communities prefer to work with Pearson correlation

[1] A *kinetoscope* is a sequence of images. As we will show, SKIMP can be visualized as producing a sequence of motif-heatmaps.

(especially seismologists [11]) while our work remains agnostic to such considerations.

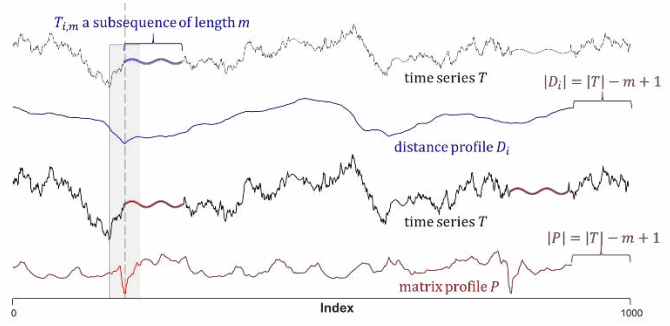

Fig. 3. *top*) A distance profile D_i created from $T_{i,m}$ shows the distance between $T_{i,m}$ and all the subsequences in T. The values in the dark zone are ignored to avoid trivial matches. *bottom*) The matrix profile P is the element-wise minimum of all the distance profiles (D_i is one of them). Note that the two lowest values in P are at the location of the 1st motif in T.

Definition 6: A *Pan Matrix Profile* (PMP) of a time series T is a matrix whose rows are the matrix profiles P_i of some time series T. The PMP is accompanied by a PMP index, recording the location of the nearest neighbor for each MP in the PMP.

To avoid extracting redundant motifs we must understand the issue of *trivial matches*:

Definition 7: (Trivial matches)**:** Given a time series T of length n containing subsequence $T_{p,m}$, if $T_{p,m}$ scores highly on any scoring function, then $T_{j,m}$ where $j\epsilon[\min(1, p - m/2), \max(p + m/2, n)]$ will almost certainly score high on the same function. These spurious high scoring subsequences are *trivial matches*.

To avoid the false positives of trivial matches when finding the top-K matches to a query, we discard some of the patterns using the concept of an *exclusion zone,* a standard practice [4].

III. Computing the Pan Matrix Profile

Before introducing algorithms to *compute* the PMP, we introduce motif-heatmaps, a technique to *visualize* the PMP.

A. Visualizing the PMP with Motif Heatmaps

While many algorithms treat the classic MP as a "black box" [24], it can be very helpful to visualize the MP for exploratory data analysis. At a quick glance, the MP can be used to visualize the *frequency* and *fidelity* (how well-conserved), and the *location* of motifs in a time series (Fig. 3.*bottom*).

We would like to achieve a similar visualization for the PMP. To achieve this, we propose mapping each MP to a one-dimensional row of a bitmap image, recording y-axis heights as a color gradient using a heatmap.

We illustrate this, in Fig. 4 using a text string analog. Given the text string:

T: `d3icdmy19qicdmnu19a`

we compute its "Matrix Profile" at every subsequence length from 1 to 5. Here the colors are discrete because subsequences $T_{i,m}$ and $T_{j,m}$ either match or they do not. Corresponding to locations at the apex of each dark triangle, we have maximal motifs of length four beginning at location 3 (`icdm`) and length two beginning at location 8 (`19`) which correspond to locations 11 and 17, respectively.

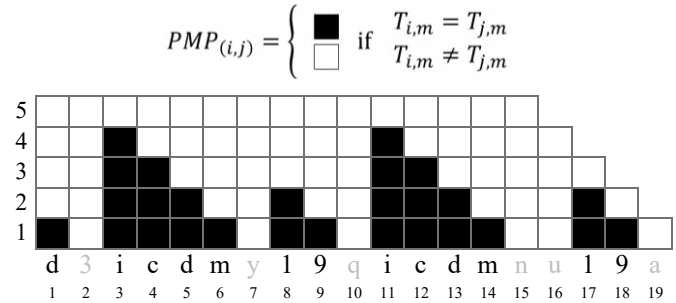

Fig. 4. A binary valued visualization of the PMP where subsequences either match (black) or don't match (white) for subsequence lengths $m = 1,2,\ldots,5$.

In a real-valued time series case, the colors of the PMP can take on subtle graduations of color or grayscale to indicate the degree of similarity.

Though motif-heatmaps perform a similar role as the ubiquitous *dot-plots* used in bioinformatics, they are not directly comparable, as dot-plots are only defined for discrete data, although a handful of papers have suggested discretizing real-valued time series to avail a dot-plot.

B. Computing the PMP

We begin with a concrete statement of the problem we wish to address:

Problem Definition: Given a time series T of length n, and a fixed range of subsequence lengths i with lower bound L, upper bound U, and step size S, we wish to produce the pan matrix profile PMP whose rows consist of matrix profiles P_i:

$$\text{PMP} = [P_L \quad P_{L+S} \quad \cdots \quad P_U]^T$$

In addition, we wish to produce a matrix PMPI whose rows consist of the matrix profile index I_i:

$$\text{PMPI} = [I_L \quad I_{L+S} \quad \cdots \quad I_U]^T$$

Before outlining our solution to this problem, we dismiss two apparently *promising* directions. Since the matrix profiles P_i and P_{i+1} will be highly related, we may attempt to "cache" some calculations used to compute one in order to reduce the number of computations required to compute the other; however, to produce meaningful results we use z-normalized Euclidean distance (or equivalently, Pearson correlation) [4][14][25][21] which makes such caching impossible. In addition, given a matrix profile P_i, it is impossible to predict or even produce an upper or lower bound for matrix profile P_{i+1} since $\max(D_j)$ for P_{i+1} may be significantly greater than its value in P_i as shown in Fig. 5 for a toy dataset with embedded noisy sine waves.

Thus, we believe there is no direct way to exploit the redundancy of adjacent matrix profiles to reduce computation. However, as we will show, we do exploit this redundancy to *order* our calculations, and achieve a faster convergence in the early stages of our anytime algorithm [21][26].

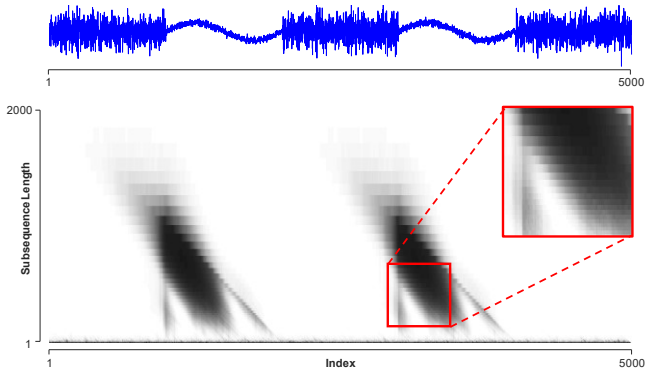

Fig. 5. A real-valued visualization of the Pan Matrix Profile of a time series T with length 5000 with a magnified "detached motif," $P_i < P_{i+1}$ for some index.

Calculating the PMP for a time series T for a range of subsequences r reduces to the calculation of r matrix profiles $P_1, P_2, \ldots, P_r$. This can easily be calculated using the brute force algorithm outlined in TABLE 1.

TABLE 1: A BRUTE FORCE ALGORITHM TO CREATE THE PMP

Input:	T:	Time series
	L:	Subsequence length lower bound
	U:	Subsequence length upper bound
	S:	Subsequence length range step size
Output:	PMP:	Pan matrix profile

```
1  R = L : S : U  // [L, L+S, L+2S, ..., U]
2  PMP = [] // |T| x |R| matrix of zeros
3  for r in R
4    PMP_r = BuildMP(T, r) // (Definition 4)
5  return PMP
```

In this algorithm we begin by explicitly specifying the range R of subsequence lengths we wish to explore (line 1) and then calculate the matrix profile P_r for consecutive subsequence lengths $L + iS$ for $i = 0, \ldots, (U - L)/S$ (lines 3-4). In line 4 we use the . MP, currently SCRIMP or STOMP [25][21]. Using this algorithm, we can generate the complete PMP shown in Fig. 6 with maximal motifs at subsequence length $m_1 = 68$ and $m_2 = 610$. An approximation of the PMP is depicted in Fig. 6 after 16 iterations, that is, after calculating $P_1, P_2, \ldots, P_{16}$, about 2% of the exact PMP has been calculated.

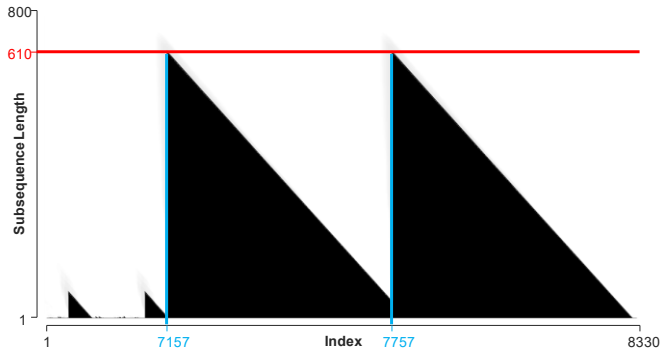

Fig. 6. The PMP of a time series T with length $n = 8330$ where we consider susbequence lengths bounded by lower bound $L = 1$, upper bound $U = 800$, and step size $S = 1$. The slice $m = 610$ corresponds to matrix profile P_{610} which has minimal values at $t_1 = 7157$ and $t_2 = 7757$. Note that this example is based on real data shown in Section IV.B.

Until relatively recently, computing this would have required R invocations of an $O(n^2 r)$ algorithm. As we will show in our experimental section, R could be over 10,000, making this algorithm completely untenable. The STOMP algorithm [25] is able to compute a single MP in just $O(n^2)$ time, giving us an overall $O(n^2 R)$ algorithm to compute the PMP. This may be tenable for small datasets, especially if we avail the GPU or multicore versions of STOMP that now exist [25]; however, it is clearly limiting given the typical sized datasets that modern data analysts need to deal with.

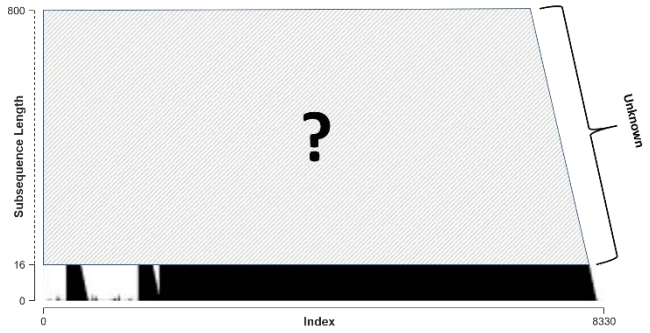

Fig. 7. An approximation of the exact PMP depicted in Fig. 6. After calculating 16 matrix profiles $P_1, P_2, \ldots, P_{16}$ or performing 2% of the required work, a little over 2% of the exact PMP has been calculated.

Though we strongly suspect that $O(n^2 R)$ may be optimal for the task-at-hand, our key insight is while it may take $O(n^2 R)$ time to converge to the exact answer, by carefully ordering the computations, we can typically allow it to converge to 99% of the exact answer, after doing less than 1% of the computations. By computing the PMP with a novel anytime algorithm, we can increase the size of datasets considered by at least two orders of magnitude.

Before continuing, it is important to ward off a possible misunderstanding. There already exist anytime algorithms to compute the MP, in particular STAMP [21] and the more recently introduced SCRIMP [25]. However, these algorithms only compute a *single* MP. To ensure fast convergence of the PMP we must optimize anytime performance at a higher level.

C. SKIMP: An Anytime Algorithm to Compute PMP

We are finally in a position to introduce SKIMP, which we outline in TABLE 2. Unlike brute-force search, SKIMP recursively subdivides the range into equally spaced regions with increasing granularity (line 1-2). This has a similar effect to iterating through a balanced binary search tree on the range $r = [1,2, \ldots, (U - L)/S]$ using breadth-first search.

TABLE 2: THE SKIMP ALGORITHM

Input:	T:	Time series
	L:	Subsequence length lower bound
	U:	Subsequence length upper bound
	S:	Subsequence length range step size
Output:	PMP:	Pan matrix profile

```
1  T = BuildBalancedBST(L,U,S)
2  R = BFS(T)
3  PMP = [] // |T| x |R| matrix of zeros
4  for r in R
5    PMP_r = BuildMP(T, r)          // (Definition 4)
6  return PMP
```

As the initialized PMP (line 3) is approximated for each subsequence r in R (line 4-5), we see a "blocky" approximation of triangles being progressively refined exemplified in Fig. 8. The example shown is slightly cleaned and contrived for clear display in this limited format of presentation, but [27] contains videos of this process created on several real-world datasets. This process is reminiscent of the classic idea of progressive refinement of raster images [17]; however, in that case, the limiting factor was bandwidth, while for us it is CPU time.

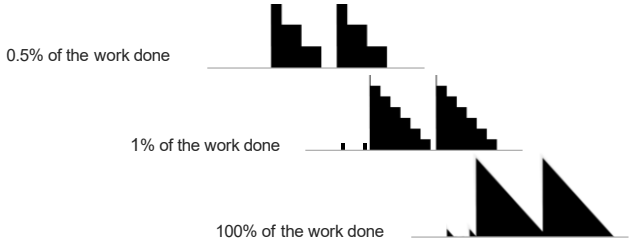

Fig. 8. A visual trace of the PMP shown in Fig. 6 approximated using known PMP values. *top*) After four iterations a single motif has been located. *middle*) A new motif with a significantly smaller subsequence length is located while the initially discovered motif is refined. *bottom*) The complete PMP as shown in Fig. 6.

As we will show in Section IV, SKIMP's approximation of the PMP depicted in Fig. 6, and many other real-world datasets converge significantly faster than brute-force search, typically achieving 99% accuracy in less than 1% of the work.

D. Ranking Motifs of Different Lengths

Given that SKIMP will allow us the possibility to find motifs of any length in the range r, it is natural to ask how we can rank motifs of different lengths. In many cases, we envision that a higher-level algorithm will make requests for motifs of different lengths, based on its own criteria (which could possibly include out-of-band information), thus absolving us of the responsibility to address this question. Nevertheless, it is an interesting question to answer. For a handful of user cases, especially on relatively small datasets, we envision this being an interactive process. Thus, we have built an interactive and visual tool to allow a user to explore and discover multiple length motifs. Nevertheless, we clearly need an algorithm that allows us to meaningfully rank motifs of different lengths.

The question largely reduces to how we trade-off fidelity vs. length. For example, which of the following pairs of strings should we adjudge more similar, {rat|rod} or {rhinoceros|rhinovirus}? A naïve application of string edit distance would rank the former more similar, however most people would find the latter pair more similar. Normalizing by dividing by the length of the strings achieves this [20].

However, a *linear* trade-off for fidelity vs. length is not appropriate for time series. Using the Euclidean distance, we normalize all MPs by dividing by the square root of the reciprocal of the length of the subsequence [13]. Of course, if we work in the correlation space, this is a non-issue.

E. Computing PMP with Unbounded U

The PMP allows for the first truly parameter-free algorithm for finding time series motifs (we could envision *several* algorithms to find motifs from the PMP, in Section III.C we gave one such example). While L is bounded by the shortest logical subsequence length, and S simply affects the desired level of granularity, the reader may argue that the value of U is a parameter and could be as long as $n/2$. However, U is only a parameter in a very weak sense, so long as it is larger than the length of the longest motif in the data, its value is inconsequential. For example, in our termite DNA example in Section IV.B the longest motif has length 610. As we did not know this in advance, we set U to a very conservative 2,400. This clearly worked, but one could argue that about 75% of the computations (from 618 to 2,400) were wasted. Can we prevent such wasted computations?

If we assume that we have a test to detect when the first row (from L upwards) of the PMP is devoid of meaningful motifs, then a simple algorithm suggests itself. We can compute this test on the MP_i with $i = L$, then iteratively double i, computing then testing MP_i until the test fails. We can then use i as the value of U, and simply call the SKIMP algorithm.

Note that since we have already computed $\log_2(U)$ of the $|R| = (U - L)/S$ MPs that SKIMP will compute, we can slightly modify SKIMP to ingest these MP and avoid recomputing them.

This idea is predicated on the assumption that we have a test to detect when the first row of the PMP is devoid of meaningful motifs. One way to achieve this, is to set a threshold for the correlation. We can calculate the maximum correlation for each subsequence and if the correlation falls below the threshold then we stop calculating the PMP for the larger subsequence length as outlined in TABLE 3.

TABLE 3: FINDING A SUITABLE UPPERBOUND FOR U

Input:	T:	Time Series
	t:	Threshold
Output:	S:	Maximum subsequence length

```
1  k = 8                  // 8 is the shortest sensible motif
2  c = inf
3  while(c >= t)          // t is set with domain knowledge
4    c = maximum(PCmatrixProfile(T,k))
5    k = k * 2            // iteratively double
6  return k
```

Using this algorithm Fig. 9 indicates a suitable upper bound at subsequence length 660 when using a maximum correlation threshold of $t = 0.988$ on a mtDNA sequence of *Coptotermes suzhouensis* (a termite) depicted in Fig. 14. Gratifyingly, we see dip just after 612, the objectively correct location [12].

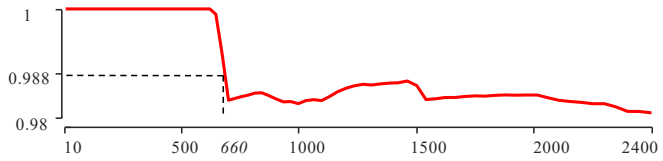

Fig. 9. The maximum correlation for different subsequence length from a mtDNA sequence of *C. suzhouensis* depicted in Fig. 14. Setting the threshold to 0.988 stops calculation of PMP for lengths greater than 660.

This opens the question of how we can set t. While this is a domain dependent value, it seems to be robust *within* a single domain. For example, $t = 0.988$ worked well on the termite DNA, the soybean DNA (Fig. 18), and all other DNA data we considered.

F. Extracting the Top-K motifs

An almost trivial application of the PMP is our algorithm for extracting the length agnostic top-K motifs as outlined in TABLE 4.

TABLE 4: DISCOVERING TOP-K MOTIFS (ANOMALIES)

Input:	PMP:	Pan Matrix Profile
	T:	Time series
	k:	Number of motifs
Output:	TM:	top-K motifs

```
1  TM ← {}
2  while |TM| < k
3    [idx, s] = maximum(PMP)
4    if TM is not covering T[idx:idx + s]
5       TM ← T[idx: idx + s]
6    Apply exclusion zone // Definition 7
7  return TM
```

Given a time series T, the corresponding PMP for some range, and a user-defined value for the number of motifs k, our algorithm returns the subsequences $T_{i,m}$ which correspond to the top-k motifs of the PMP. To extract the top-k motifs, we repeatedly search the PMP for its minimum value (line 3) and then add the corresponding subsequence to our top-k motifs TM only if TM does not span the subsequence (lines 4-5). Afterwards, we apply an exclusion zone using the recovered subsequences to ensure we do not find a trivial match (line 6). Using this algorithm, we discovered the top-2 motifs for the mitochondrial DNA sequence shown in Fig. 14 $T_{7713,615}$ and $T_{7092,70}$ and their respective nearest neighbors $T_{7148,615}$ and $T_{6900,70}$ which is in near perfect agreement with the ground truth noted in [12] and will be illustrated in Fig. 14.

G. Anomaly Detection

The previous algorithms for detecting subsequence length agnostic top-k *motifs* can be easily modified to detect *anomalies* with variable lengths. By "inverting" the PMP, $PMP' = 1 - PMP$ the top-k motifs produced by the algorithm would correspond to the top-k anomalies. We exemplify this algorithm's ability to perform anomaly detection on an automated pedestrian counting system developed in Taipei to better understand pedestrian activity within the municipality. This information examines how people use different city locations at different times of day to better inform decision-making and infrastructure planning. We extract data from the Xindian District Office as shown in Fig. 10.

Fig. 10. Pedestrian count data from Taipei Xindian District Office metro station starting on December 2015 and ending at March 2017.

Fig. 11 shows one fairly typical week of this behavior.

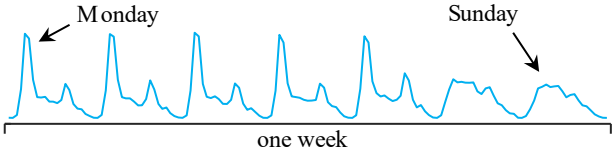

Fig. 11. One fairly typical week data of the pedestrian counting data of Taipei.

After computing the PMP for this data from L = 20 points (~one day) to U = 200 points (~10 days), we can then extract the top-k anomalies using the modified version of the top-k motifs algorithm described in Section F.

Fig. 12 shows the top-4 anomalies that exist in this dataset with the anomaly shown in red.

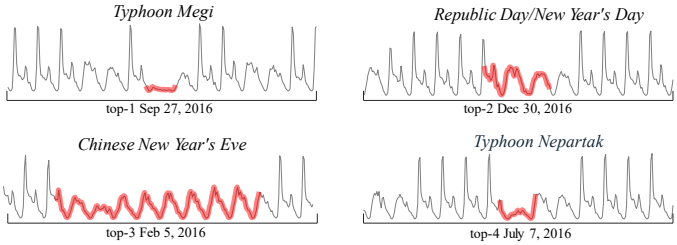

Fig. 12. The top-4 anomalies (red) detected for the data shown in Fig. 10.

Note that these four anomalies represent four different lengths. The first anomalies length is 27 points (~one day). We found the ground truth for this event which is described by [26] as "*On September 26/27, 2016 … Typhoon Megi …* (made) *landfall on Taiwan's southeast coast*" The second and third anomalies belong to Republic day/New year's day and Chinese New Year's Eve with a length of 59 (~2.5 days) and 185 points (~one week). The last anomalies' length is 32 points (~1.5 days) which is reported by [5] as "*The storm is predicted to make landfall on the island nation on July 7.*"

If we set the length of classic anomaly detection algorithm to one day, it fails to find Chinese New Year's Eve or Republic Day/New year's day. In contrast, if we set the length of classic anomaly detection to one week, we cannot detect the other anomalies are that are present in Fig. 12. This strongly motivates the need for computing similarity search for variable length.

IV. Experimental Evaluation

To ensure that our experiments are reproducible, we have built a website [27] which contains all data/code/raw spreadsheets for the results, in addition to many experiments that are omitted here for brevity. Unless otherwise stated, all experiments were run on a Dell XPS 8920, with Intel Core i7-7700 CPU @ 3.6GHz and 64GB RAM.

A. A Benchmark for the All-Length Motif Problem

To concretely ground our ideas throughout this paper, let us consider a motivating problem introduced to us by (blinded). They are interested in finding motifs in time series from a large industrial distillation column. The apparatus is massive with great thermal and mechanical inertia, so that it suffices to sample it once per minute (1/60 Hertz). When doing analytics, it is common for them to consider data from the previous year; thus, we have a time series with n = 525,600 points. Let us call this dataset DisCol. Occasionally, DisCol is searched for motifs which are used in downstream analytics to perform root-cause analysis. Though most patterns last for about a day, the fast cooling process of the apparatus by a summer rain shown can induce patterns lasting for only a few hours; thus, there is great uncertainty in the potential length of motifs motivating the desire to identify motifs between the length of one hour (L = 60

minutes) and one day (U = 1,440 minutes), a range of $|R|$ = 1,380 values.

Prior to the introduction of the Matrix Profile, the only exact algorithm to find all such high-dimensional motifs was brute force search, which would take $O(n^2r^2)$ time. The factor r appears both as the subsequence length (more conventionally denoted m), *and* the number of times we must run the motif search. Concretely, on our desktop, this would require about 48 years. Using the recently introduced STOMP algorithm, which can find motifs of a fixed length in time independent of that length, this can be reduced to $O(n^2r)$ time, or about 23 hours. However, since STOMP is a batch algorithm, it is natural to ask how quickly we can converge to an acceptable approximation of the final PMP.

We have created a proxy for the data in question by editing together some publicly available industrial benchmarks from a similar process. In Fig. 13 we show how fast SKIMP converges on this dataset.

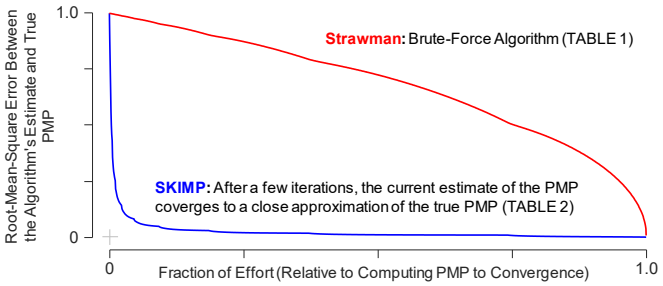

Fig. 13. The root-mean-square error of our approximation of the fully converged PMP when using SKIMP and the brute-force algorithms. When compared to the ten hours required to fully compute the PMP using the brute-force algorithm (TABLE 1), SKIMP required less than 41 min, or 3% of the required effort to achieve an approximation with less than 10% error. A video showing the convergence of the PMP approximation using SKIMP is available at [27].

In an ideal case, there should exist an approximation of the PMP which converges to the exact PMP using a small fraction of the effort required to compute the complete PMP. In less than 3% of the required effort, SKIMP was able to approximate the full PMP with less than a 10% root-mean-square error. Though it may seem as if the root-mean-square error plot depicted in Fig. 19 converges slower than the plots depicted in Fig. 13 and Fig. 20, this plot is significantly shorter (by an order of magnitude) but still only requires a handful of iterations before converging to the complete PMP.

B. DNA-Based Benchmarks for the All-Length Motif Problem

To demonstrate the utility and correctness of time series motif length discovery, we exploit a technique long used by the time series data mining community. By converting discrete DNA sequences to real-valued time series, we can explore the time series space of the DNA sequence and then attempt to confirm our findings with molecular biologists. In Fig. 14 we show the complete mitochondrial DNA sequence of *Coptotermes suzhouensis*, a subterranean termite pest of wooden structures as a real-valued time series and its corresponding PMP as a motif-heatmap in Fig. 15.

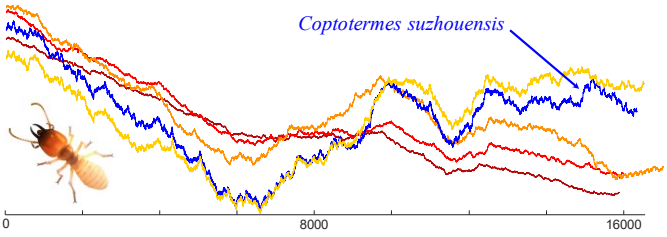

Fig. 14. The mitochondrial DNA sequence of five randomly chosen insects, including the 16,326 bp mitochondrial DNA sequence of *Coptotermes suzhouensis* (blue).

These results are suggestive of a strongly conserved motif of a length of about 610 in the center of the sequence. In a recent paper announcing the complete mitochondrial genome of this insect, the authors noted that the mitogenome had two repeat units, **A** and **B**. Unit **A** is just 66 bp long, however "*The **B** repeats consisted of one complete unit B1 (562 bp) and a partial unit B2 (38 bp)*" [12]. The reader will appreciate that 562 + 38 sums to 610, which is just 0.03% less than our suggested motif length of 612 for this dataset.

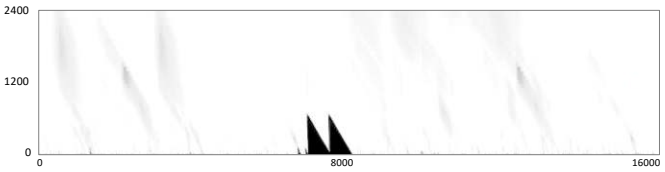

Fig. 15. A motif-heatmap of (the time series representation of) the mitochondrial DNA sequence of *Coptotermes suzhouensis* in Fig. 14 from L= 10 to U = 2,400.

Moreover, if we zoom in as shown in Fig. 16, the **A** motif is also clearly visible.

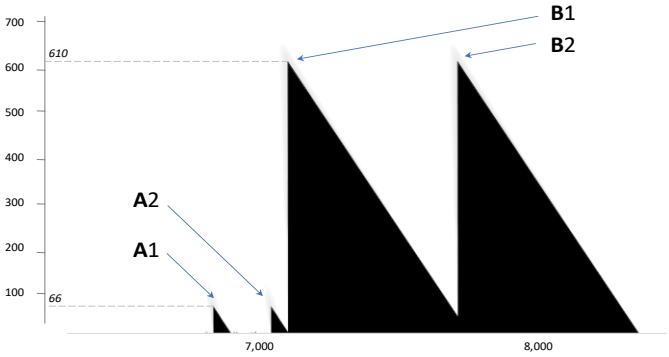

Fig. 16. A zoom-in of the center bottom of Fig. 15 shows that the motif heatmap also discovered the much shorter **A** motif.

The reader may wonder if DNA time series is "too easy" given the level of conservation observed. To address this, we can revisit our termite example. This time, before converting the DNA string to time series, we randomly changed every base with a one in sixty-four probability, simulating a high mutation rate. As the motif heatmap in Fig. 17 reveals, this level of noise makes no appreciable difference in our ability to find the motifs.

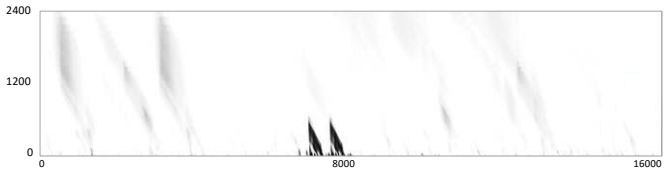

Fig. 17. (contrast with Fig. 15) A motif-heatmap of (the time series representation of) the mtDNA of sequence *Coptotermes suzhouensis* after 1-in-64 bases were randomly changed.

Above we have demonstrated our technique on an insect which was chosen to be visually clear in two-column format, but it is arguably too simple to really challenge our algorithms. In order to stress test SKIMP, we can turn to plant mtDNA. It has long been noted that "*Unlike the relatively simple mitochondrial genomes of animals, the genomes of nonparasitic flowering plant mitochondria are large and complex*." [1]. Thus, we consider the mitochondrial genome of Soybean (*Glycine max*) [3]. Because it is 402,540 bp long and has repeats that differ over three orders of magnitude in length, it is difficult to do it full justice in this paper. In the accompanying web materials [27], we show a video of our methods applied to it, and here we content ourselves with a figure that allows us to see only the longer motifs.

From the literature we know that repeats in plants may be as long as 10,000, thus we consider $L = 1{,}000, U = 10{,}000$, with $S = 1$. To run this dataset to convergence requires about 42 days, however, as the video at [27] shows, in about half a day, the basic shape of the final motif heatmap has already emerged.

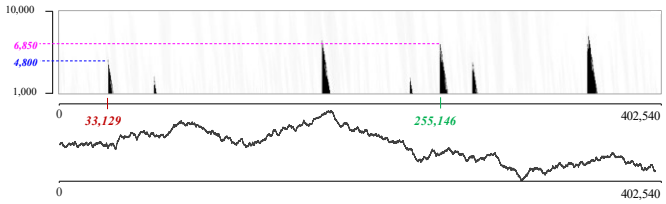

Fig. 18. *bottom*) The mtDBA of Soybean in a time series representation. *top*) The motif-heatmap of Soybean. The location and length of two motifs are highlighted. As [3] discovered, the (location/*length*) of the first occurrence of **R**2 is (33,155/*4,692*), which is very close to our result of (33,129/*4,800*). Similarly, [3] notes the (location/*length*) of **R**1d is (255,146/*6,502*), we discovered a motif the exactly the right location 255,146, with a slightly different length *6,850*.

This experiment offers strong evidence of the utility of our anytime approach. In addition, SKIMP was able to approximate the PMP with less than 10% root-mean-square error when performing less than 4% of the required effort as shown in Fig. 19.

Note that such DNA repeats could also be visualized using dot-plots. However, recall that dot-plots require $O(n^2)$ space, whereas motif-heat maps require only $O(nr)$ space, and their long aspect ratio is amenable to panning interactions when dealing with long sequences. More importantly, dot-plots are only well defined for discrete strings while motif-heat maps facilitate the visualization of *real-valued* data.

C. All-Length Motifs in Seismology

In this section we consider motif discovery in seismic data. It may not be obvious, but two earthquakes from the same location, even if recorded decades apart, will have similar waveforms. The waveform similarity results show that the waveform from the source to station is affected by the same process (i.e., seismic refraction, reverberation, and reflections).

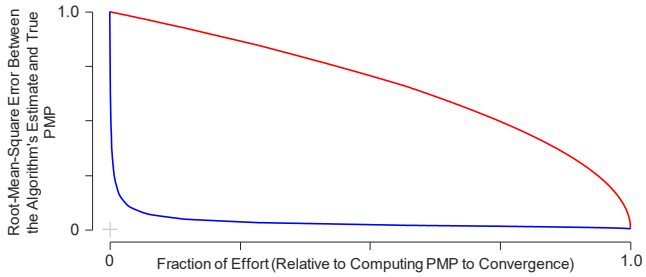

Fig. 19. The change in the root-mean-square error when approximating the PMP after performing a fraction of the work required to compute the PMP to convergence on the mtDBA Soybean time series representation. Using less than 4% of the time required to compute the entire PMP, SKIMP found an apporximation to the PMP which had a root-mean-square error of less than 10%. A video showing the converging of SKIMP is available at [27].

Seismologists can exploit different aspects of seismograms (e.g., seismic wave amplitude at various stations) to calculate the magnitude of earthquakes. One simple method of calculating earthquake magnitudes is to use the *duration* of the earthquake signals, and then apply a formula to map it to *magnitude* [2][11]. One problem of using this method is that although the onset of the earthquake signal is usually clear, the tail of earthquake signal cannot always be determined clearly as the signal saturates in the background seismic noise. Moreover, for distant events, the exact timing of the onset may also be difficult to determine. This process is usually performed by visual inspection of the earthquake waveforms and thus requires human effort [2] and can contain bias/error from using different analysts.

Here we tested the PMP as an alternative way of estimating duration magnitude for local earthquakes. We picked four earthquakes from seismicity in the central San Andreas fault near Parkfield, CA and tested our approach using the data recorded at a PGH station from the northern California seismic network. We considered 260 seconds of the gain controlled seismic data at 20 Hz.

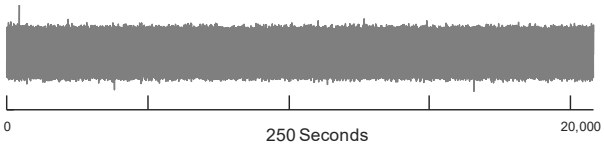

Fig. 20. Concatenated seismic waveform (after applying a gain control) recorded at PGH seismic station and resampled at 20 Hz for four earthquakes located near the Central San Andreas fault, near Parkfield, CA. *left*) the full dataset.

It is important to note that we only processed the data by deleting irrelevant sections, bringing four events within a 260 second time span, to allow us to create intuitive plots.

A classic equation to map duration to amplitude is [11]:

$$M_d = 2.0(\log T) + 0.0035T - 0.87 + stacor \quad (1)$$

Here M_d is the earthquake magnitude calculated based on duration, T is the earthquake waveform duration, and *stacor (station correction term*) is a constant that depends on the station's characteristics (instrument, near the station structural properties) and methods [11]. In this case, we do not have the

stacor parameter, and therefore, we cannot calculate the *absolute* magnitude from the PMP. However, if we take the difference between magnitude of two events, the *stacor* term cancels, and we can test if the PMP approach can estimate the *difference* between two earthquake magnitudes. With this in mind, we computed the PMP as shown in Fig. 21.

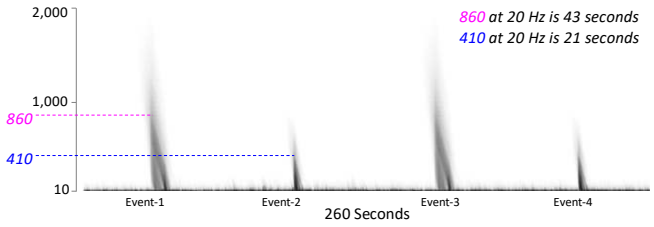

Fig. 21. A motif-heatmap of the PGH (Parkfield, CA) seismic data. The events shown originate from two distinct locations where Event-1 and Event-3 are from one origin and Event-2 and Event-4 are from another.

We ran our top-k motif algorithm with $k = 2$. The algorithm indicated a length of ~43 seconds for Event-1 and Event-3 and ~21 seconds for Event-2 and Event-4.

By plugging these values into (1), we estimated the magnitude difference between earthquakes to be 0.70. The Northern California Earthquake Catalog Search (NCSN) catalog [13] reports the local magnitude of Event-1 (event ID number 21476722) is 2.00 and Event-2 (event ID number 21432310) is 1.25. Thus, the values from the NCSN catalog indicate the difference between magnitudes to be around 0.75, in close agreement with our estimate of 0.70.

Fig. 22 shows SKIMP approximating the PMP spanning a range of subsequence lengths from 10 to 2000 which reduced the approximation's error to less than 10% while performing only 15% of the required time and effort.

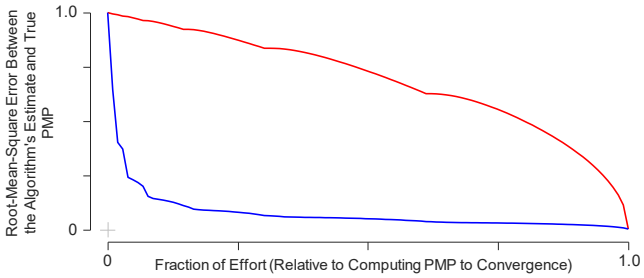

Fig. 22. The root-mean-square error of our PMP approximation using the SKIMP and brute-force algorithms on seismic waveform data. SKIMP achieved a root-mean-square error of less than 10% when using only 15% of the time and effort required to fully compute the PMP. A video showing the convergence of the PMP approximation using SKIMP is available at [27].

Note that our method is fully automated and can be used at large scale. In contrast, traditional methods are subjective and require significant human effort. For example [2] noted, "*The estimate of event duration is visually defined by the analysts from P-onset time until the point when the signal envelope decays down to the pre-event noise level. Nevertheless,* (because of human subjectivity) *definition of event duration is not homogeneous at each observatory.*"

D. *Reducing Space Requirements*

Recall our motivating problem was DisCol. For this dataset we wish to support length agnostic motif search in the range of one hour to one day, at one minute steps, which requires storing 1,380 Matrix Profiles and Matrix Profile Indices for the PMP.

Naively, the memory requirement for this is as follows: for the Matrix Profiles, $527{,}040 \times 1{,}380 \times 4$ bytes $= 5.81$ gigabytes and for the Matrix Profiles Indices, $527{,}040 \times 1{,}380 \times 4$ bytes $= 2.90$ gigabytes, totaling 8.72 gigabytes. This is not untenable for modern machines, but it is uncomfortably large. Can we improve this?

The first observation is that we do not need the Matrix Profiles Indices to create the motif heatmaps. So, as each one is computed in line 4 of TABLE 2, we can flush them to disk, just in case one of them is later requested by some downstream algorithm.

The second observation is that if our main task is to produce a motif-heatmap, then we do not need to keep all 527,040 values of each Matrix Profile, as this would give us a finer resolution that we could possibly display. We have at most 7,680 pixels (the 8K standard) of width to specify. Thus, we can aggregate chunks of values, map them to a single pixel, and then flush the original higher resolution Matrix Profile to disk in case one of them is later requested by some downstream algorithm.

Given this basic approach, there is no real memory bottleneck for computing the PMP.

V. Related Work

The literature on time series motif discovery is large and growing, see [21] and the references therein. However, to the best of our knowledge, there are no other algorithms that can approximately or exactly discover *all* motifs of arbitrary lengths.

The work closest in spirit to ours is VALMOD [13]. The idea of VALMOD is to compute the MP for the shortest length of interest, then use the information gleaned from it to guide a search through longer subsequence lengths, exploiting lower bounds to prune off some calculations. This idea works well for the first few of the longer subsequence lengths, but the lower bounds progressively weaken, making the pruning ineffective. Thus, in the five case studies they presented, the mean value of U/L was just 1.24. In contrast, consider that our termite example in Fig. 15 has a U/L ratio of 240, more than two orders of magnitude larger. Thus, VALMOD is perhaps best seen as finding motifs with some tolerance for a slightly (~25%) too short user-specified query length, rather than a true "motif-of-all-lengths" algorithm. Also note that apart from the shortest length, VALMOD only gives *some* information for the other lengths, unlike PMP, which contains exact distances for *all* subsequences of *all* lengths.

In a sequence of papers, Lin and colleagues introduce a series of tools to allow interactive discovery of variable-length time series patterns [22]. However, this work is not directly comparable to PMP. First, because they use a discretized representation of the data (for efficiency), they are always condemned to finding approximate answers. Second, the system only returns information about a small *subset* of the patterns, whereas PMP contains exact distances for *all* subsequences of

all lengths. Finally, there are many parameters to be set and choices to be made in the grammar inference algorithm. However, like us, the authors see great value in attempting to visualize the results of the motif search.

Unsurprisingly, given the explosion of interest in deep learning, there is at least one paper on "deep" motifs [9]. However, in spite of the title of the work, the algorithms/representations presented are what the data mining community would call (semi-supervised) *clustering*, not motif discovery.

Note that our ability to find motifs without specifying their length ahead of time, removes the final parameter in time series motif discovery. While progress in data mining is often measured only in time or accuracy, we would argue that this is a significant milestone. The first paper to propose time series motif discovery required the user to set five parameters (length, SAX cardinality, SAX dimensionality, mask size, iterations) [4]. Mueen managed to reduce this to just two (length, number of reference points) [14], and the original Matrix Profile reduced it to just one (length) [21]. Every reduction in the number of parameters seems to have been accompanied by a dramatic increase in the number of practitioners exploiting motif discovery. We hope that this final reduction will continue this trend.

VI. Conclusions

We have introduced the first practical technique to find motifs and discords [21] for all lengths. Given the glut of information that this provides, we have also introduced a novel visualization that allows a practitioner to understand the location, length, and fidelity of all motifs in her dataset. We have shown that these new tools allow us to find useful conserved structures and anomalies in domains as diverse as bioinformatics, transportation, and seismology.

In future work plan to investigate the implications of our ideas for other algorithms that exploit the matrix profile, including chain discovery [24] and segmentation.

VII. Acknowledgements

We would like to thank NSF 1631776

VIII. References

[1] Brainstorm User Guide URL, Retrieved May 10 2019: neuroimage.usc.edu/brainstorm/Tutorials/ArtifactsDetect

[2] Castello, B., M. Olivieri, and G. Selvaggi. "Local and duration magnitude determination for the Italian earthquake catalog, 1981–2002." *Bulletin of the Seismological Society of America* 97.1B (2007): 128-139.

[3] Chang, S., Wang, Y., Lu, J., Gai, J., Li, J., Chu, P., Guan, R. and Zhao, T., 2013. The mitochondrial genome of soybean reveals complex genome structures and gene evolution at intercellular and phylogenetic levels. *PLoS One*, *8*(2), p.e56502.

[4] Chiu, B., Keogh, E. and Lonardi, S., 2003, August. Probabilistic discovery of time series motifs. In Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining (pp. 493-498). ACM.

[5] https://earthobservatory.nasa.gov/images/88326/typhoon-nepartak

[6] https://earthobservatory.nasa.gov/images/88820/typhoon-megi

[7] Imani, Shima, et al. "Matrix Profile XIII: Time Series Snippets: A New Primitive for Time Series Data Mining." *2018 IEEE International Conference on Big Knowledge (ICBK)*. IEEE, 2018.

[8] Kemp, B., Zwinderman, A.H., Tuk, B., Kamphuisen, H.A. and Oberye, J.J., 2000. Analysis of a sleep-dependent neuronal feedback loop: the slow-wave microcontinuity of the EEG. *IEEE Transactions on Biomedical Engineering*, *47*(9), pp.1185-1194.

[9] Kozik, A., Rowan, B., Lavelle, D., Berke, L., Schranz, M. E., Michelmore, R. W., & Christensen, A. C. (2019). The alternative reality of plant mitochondrial DNA. bioRxiv, 564278.

[10] Lee, N. K., Azizan, F. L., Wong, Y. S., & Omar, N. (2018). DeepFinder: An integration of feature-based and deep learning approach for DNA motif discovery. Biotechnology & Biotechnological Equipment, 32(3), 759-768.

[11] Lee, W. H. K., Bennet, R. E. and Meaghu, K. L., A method of estimating magnitude of local earthquakes from signal duration. U.S. Geological Survey Open File Report, 1972, 28 pp

[12] Li, J., Zhu, J.L., Lou, S.D., Wang, P., Zhang, Y.S., Wang, L., Yin, R.C. and Zhang, P.P., 2018. The Complete Mitochondrial Genome of Coptotermes 'suzhouensis'(syn. Coptotermes formosanus)(Isoptera: Rhinotermitidae) and Molecular Phylogeny Analysis. *Journal of Insect Science*, *18*(2), p.26.

[13] Linardi, M., Zhu, Y., Palpanas, T.,. Keogh,. E: Matrix Profile X: VALMOD - Scalable Discovery of Variable-Length Motifs in Data Series. SIGMOD Conference 2018: 1053-1066

[14] Mueen, A., Keogh, E., Zhu, Q., Cash, S. and Westover, B., 2009, April. Exact discovery of time series motifs. In *Proceedings of the 2009 SIAM international conference on data mining* (pp. 473-484). Society for Industrial and Applied Mathematics.

[15] Northern California Earthquake Catalog Search, URL. Accessed 5-25-19. ncedc.org/ncedc/catalog-search.html

[16] Rani, M.S.A. and Mansor, W., "Detection of Eye Blinks From EEG Signals for Home Lighting System Activation," Proceeding of the 6th International Symposium on Mechatronics and its Applications (ISMA09). Sharjah, UAE 24-26 March 2009.

[17] SLOAN, K. R., JR., AND TANIMOTO, S. L. 1979. Progressive refinement of raster images. 1EEE Trans. Comput. 28, 11 (Nov.), 871-874.

[18] Valderrama, J.T., de la Torre, A. and Van Dun, B., 2018. An automatic algorithm for blink-artifact suppression based on iterative template matching: Application to single channel recording of cortical auditory evoked potentials. *Journal of neural engineering*, *15*(1), p.016008.

[19] Vidal, Antonio, and Luis Munguía. "A new coda-duration magnitude scale for northern Baja California, Mexico." Geofísica internacional 44, no. 1 (2005): 11-22.

[20] Weigel, A. and Fein, F., 1994, October. Normalizing the weighted edit distance. In Proceedings of the 12th IAPR International Conference on Pattern Recognition, Vol. 3-Conference C: Signal Processing (Cat. No. 94CH3440-5) (Vol. 2, pp. 399-402). IEEE.

[21] Yeh, C.C.M., Zhu, Y., Ulanova, L., Begum, N., Ding, Y., Dau, H.A., Silva, D.F., Mueen, A. and Keogh, E., 2016, December. Matrix profile I: all pairs similarity joins for time series: a unifying view that includes motifs, discords and shapelets. In *2016 IEEE 16th international conference on data mining (ICDM)* (pp. 1317-1322). IEEE.

[22] Yifeng Gao, Jessica Lin, Huzefa Rangwala: IterativE Grammar-Based Framework for Discovering Variable-Length Time Series Motifs. ICDM 2017: 111-116.

[23] Zhu, Y., Gharghabi, S., Silva, D. F., Dau, H. A., Yeh, C. C. M., Senobari, N. S., ... & Mueen, A. The Swiss Army Knife of Time Series Data Mining: Ten Useful Things you can do with the Matrix Profile and Ten Lines of Code.

[24] Zhu, Y., Imamura, M., Nikovski, D., & Keogh, E. J. (2018, July). Time Series Chains: A Novel Tool for Time Series Data Mining. In IJCAI (pp. 5414-5418).Yan Zhu, Makoto Imamura, Daniel Nikovski, Eamonn J. Keogh: Time Series Chains: A Novel Tool for Time Series Data Mining. IJCAI 2018: 5414-5418.

[25] Zhu, Y., Zimmerman, Z., Senobari, N.S., Yeh, C.C.M., Funning, G., Mueen, A., Brisk, P. and Keogh, E., 2016, December. Matrix profile ii: Exploiting a novel algorithm and gpus to break the one hundred million barrier for time series motifs and joins. In 2016 IEEE 16th international conference on data mining (ICDM) (pp. 739-748). IEEE.

[26] Zilberstein, S., & Russell, S. (1995). Approximate reasoning using anytime algorithms. In Imprecise and approximate computation (pp. 43-62). Springer, Boston, MA.

[27] Project Website: http://sites.google.com/view/pan-matrix-profile