Technical Report CMP-C14-01: Finding Motif Sets in Time Series

Anthony Bagnall, Jon Hills and Jason Lines,

Abstract—Time-series motifs are representative subsequences that occur frequently in a time series; a motif set is the set of subsequences deemed to be instances of a given motif. We focus on finding motif sets. Our motivation is to detect motif sets in household electricity-usage profiles, representing repeated patterns of household usage.
We propose three algorithms for finding motif sets. Two are greedy algorithms based on pairwise comparison, and the third uses a heuristic measure of set quality to find the motif set directly. We compare these algorithms on simulated datasets and on electricity-usage data. We show that Scan MK, the simplest way of using the best-matching pair to find motif sets, is less accurate on our synthetic data than Set Finder and Cluster MK, although the latter is very sensitive to parameter settings. We qualitatively analyse the outputs for the electricity-usage data and demonstrate that both Scan MK and Set Finder can discover useful motif sets in such data.

Index Terms—Time series, Motifs, Electricity profiles

1 Introduction

Time-series motifs are subsequences that occur frequently in a time series [1]. They can be used to characterise the typical behaviour of a time series to allow for classification or anomaly detection (e.g. [1]), or as a primitive in, for example, association rule mining (e.g. [2]). Areas of application include medicine [1], image processing [1], and robotics [3]. Our interest lies in finding motifs in household electricity-usage profiles, and using them to disaggregate the data in terms of devices [4].

The key contributions to the study of motifs are presented in [1], [5], [6]. The algorithm described in [1] operates on discretised time series. This approximate approach is inappropriate for our problem, as the electricity-usage data is already aggregated into 15-minute periods, a time frame that makes device disambiguation difficult. Further compression would make detection impossible. The emphasis of the later works is on finding best-matching pairs of subsequences. Our contribution is directed at exact discovery of frequently occurring subsequences, rather than best-matching pairs. We propose three algorithms for this purpose. Two (Sections 3.1 and 3.2) use the MK pair-matching algorithm as a subroutine (although any pair-finding algorithm could be used), and the other finds motif sets based on whole set quality, rather than pair matching (Section 3.3). We assess their performance (Sections 5 and 6) on both real and synthetic data, described in Section 4. All of the code and data used in this paper can be found at [7]; the password for the zipped files is 'motset13'.

• *J. Hills, J. Lines, and A.Bagnall are with the School of Computing Sciences, University of East Anglia, Norwich, Norfolk, United Kingdom. E-mail: {ajb,j.hills,j.lines}@uea.ac.uk*

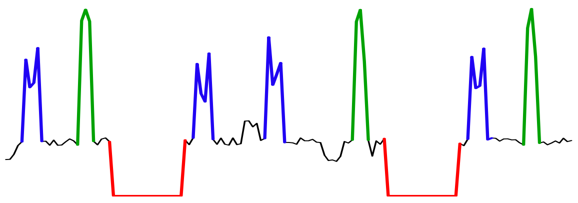
Fig. 1. A simulation of an electricity demand profile, with three motif sets.

2 Background

A *time series* is a sequence $T =< t_1, t_2, ..., t_m >$ of m real-valued numbers. Images and spectrographs [8], and other sequential data, as well as temporally-ordered data, may be regarded as time series. Time-series data-mining research is concerned predominantly with discovering similarities between subsequences of time series, rather than between whole time series, see e.g. [1]. A *subsequence* of length n of a time series T is a time series $S_a =< t_a, t_{a+1}, ..., t_{a+n-1} >$ where $1 \leq a \leq m - (n - 1)$. A *sliding window* produces the set S of all possible subsequences S_i of size n of T. The cardinality of S is $m - (n - 1)$.

We use the term *motif* to refer to a single subsequence (which can be a concrete instance, or the average of the members of its motif set), and the term *motif set* to mean the set of subsequences that are associated with a given motif. The 1-motif set problem is to find the largest subset of S whose members are considered to *match* one another, where two series match if the distance between them is less than some threshold parameter, r, and the match is *non-trivial*. As shown in [9], failing to prevent trivial matching renders motif detection meaningless. There are alternative definitions of a trivial match; we adopt the definition used in [9], and take it that two series cannot match if they overlap.

The K-motif set is defined as the K^{th} most commonly-occurring subsequence that does not overlap with the previous $(K-1)$-motif sets. Finding K motif sets requires that we enforce a separation of at least $2r$ between each motif set and all previous motif sets [1].

2.1 Mueen-Keogh (MK) Best-matching Pair Algorithm.

An exact pair-finding algorithm called Mueen-Keogh (MK) is proposed in [5]. MK finds closest matches between time-series subsequences using a form of early abandon that dramatically speeds up the matching process in the average case. Finding best-matching pairs is of less interest to us than detecting the daily repeating pattern of, for example, a washing machine or oven. This is an important point because the best-matching pair are not necessarily members of a high-cardinality motif set; for example, in Fig. 1, the red subsequences form the best-matching pair, but the blue and green motif sets have higher cardinality. A formal description of MK is given in [5]; we use the algorithm to find best-matching pairs as one stage in the motif-set discovery process; any pair-finding algorithm can be substituted for MK, however.

3 FINDING THE K-MOTIF SETS

In this section, we describe three algorithms for finding the K-motif sets. Two are based on constructing sets using pairs, the other constructs motif sets directly. MK is a fast way of finding matching pairs; however, this does not in itself provide a way of finding motif sets.

3.1 Scan MK.

We have extended the method for finding the range motif outlined in [6] to find approximate K-motif sets. We iterate the process of finding closest pairs and their matches, adding them to a motif set and removing members and their trivial matches from the list of candidates after each iteration. The algorithm is described in Fig. 2. We assume a distance function $d(S_i, S_j)$ is defined (we use Euclidean distance for all experiments).

MK is used to find the best-matching pair of subsequences in S (line 7); if the distance between them is greater than $2r$, the algorithm terminates. Otherwise, the best-matching pair is added to a motif set, the trivial matches of the best-matching pair are removed from S (lines 8-16), and the remaining subsequences are scanned. Any subsequences within $2r$ of both members of the best-matching pair are added to the motif set (lines 17-22).

The `condense` function operates as follows. For each set of contiguously-indexed subsequences, the non-trivial match is taken to be the subsequence with the smallest total distance between it and each of the members of the motif set. The contiguously-indexed subsequences are taken to be trivial matches, and are excluded from the motif set. The motif set is further condensed by removing one subsequence of any pair whose members are more than $2r$ apart. We choose which subsequence to exclude based on the number of clashes. For example, if a subsequence that clashes with three others is greater than $2r$ from a subsequence that clashes with two others, the first subsequence is removed, maximising the cardinality of the set. Ties are decided based on average linkage; the subsequence with the shortest total distance to the other members of the set is retained. Once the motif set is established, its members and their trivial matches are removed from the candidate set, and the process is repeated until no more subsequences are within $2r$ of each other.

3.2 Cluster MK.

The second algorithm we develop is based on hierarchical clustering of best-matching pairs. Hierarchical clustering is a widely used clustering approach, see for example [9], based on finding best-matching pairs of series. We use MK to find the pairs, and an adapted form of bottom-up hierarchical clustering described in Fig. 3.

We find the closest pair of subsequences, then merge this pair to form a new cluster (motif set). The cluster is represented by a new subsequence found by averaging the input subsequences, weighted by the number of subsequences that have already been combined to make each candidate. This ensures the cluster centre accurately reflects the members of the cluster. The process is repeated until the distance between the best-matching pair is greater than r. At this point the subsequence set S will contain the motifs, and the motif sets can be recovered from the clustering data structure.

3.3 Set Finder.

We propose an algorithm to find the K-motif sets directly, based on counting and separating (Fig. 4). Each subsequence is compared to every other subsequence, and the non-trivial matches are counted. The set of counts is sorted. The sorted set is then input to the function `separate`, which checks each subsequence with a non-zero count in order to ensure that it is at least $2r$ apart from subsequences with a greater number of matches. Subsequences that fail the test are removed from the set. An early abandon based on the

Fig. 2. Scan MK

```
Input: $T$: a time series.
    $q$: the number of subsequences for MK.
    $r$: half the maximum width of a motif set.
    $n$: the width of the sliding window.
Output: $M$: a set of motif sets.
 1: $M \leftarrow \emptyset$
 2: $F \leftarrow \texttt{slidingWindow}(T, n)$
    $\{F$ is the full set of subsequences of length $n\}$
 3: $S \leftarrow F$
 4: $k \leftarrow 0$
 5: while $end$ = false do
 6:   $end \leftarrow$ true
 7:   $\{L_1, L_2\} \leftarrow \texttt{MK}(S, q)$ $\{L_1$ and $L_2$ are indexes in $F\}$
 8:   if $d(F_{L_1}, F_{L_2}) \leq 2r$ then
 9:     $end \leftarrow$ false
10:     $k \leftarrow k + 1$
11:     $M_k \leftarrow \{F_{L_1}, F_{L_2}\}$
12:     $D \leftarrow \emptyset$
13:     for $i \leftarrow 1$ to $|S|$ do
14:       if $\texttt{trivialMatch}(F_{L_1}, S_i)$
          $\vee\ \texttt{trivialMatch}(F_{L_2}, S_i)$ then
15:         $D \leftarrow D \cup S_i$
16:     $S \leftarrow S - D$
17:     for $i \leftarrow 1$ to $|S|$ do
18:       if $d(F_{L_1}, S_i) \leq 2r \wedge d(F_{L_2}, S_i) \leq 2r \wedge S_i \notin D$
          then
19:         $M_k \leftarrow M_k \cup S_i$
20:         for $j \leftarrow 1$ to $|S|$ do
21:           if $\texttt{trivialMatch}(S_i, S_j)$ then
22:             $D \leftarrow D \cup S_j$
23:     $S \leftarrow S - D$
24:     $M_k \leftarrow \texttt{condense}(M_k, r)$
25: if $k > 0$ then
26:   $M \leftarrow \{M_1, ..., M_k\}$
27:   $\texttt{sort}(M)$
28: return $M$
```

Fig. 3. Cluster MK

```
Input: $T$: a time series.
    $q$: the number of subsequences for MK.
    $r$: the radius of the motif clusters.
    $n$: the width of the sliding window.
Output: $S$: a set of motifs.
 1: $F \leftarrow \texttt{slidingWindow}(T, n)$
 2: $S \leftarrow F$
 3: while $end$ = false do
 4:   $end \leftarrow$ true
 5:   $\{L_1, L_2\} \leftarrow \texttt{MK}(S, q)$
 6:   if $\texttt{d}(F_{L_1}, F_{L_2}) \leq r$ then
 7:     $end \leftarrow$ false
 8:     $S \leftarrow S - \{F_{L_1}, F_{L_2}\}$
 9:     $c \leftarrow merge(F_{L_1}, F_{L_2})$
10:     $S \leftarrow S \cup c$
11: return $S$
```

value of r is built into the distance function to speed up the algorithm.

Fig. 4. Set Finder

```
Input: $T$: a time series.
    $r$: the maximum distance between matches.
    $n$: the width of the sliding window.
Output: $M$: a set of motif sets.
 1: $S \leftarrow \texttt{slidingWindow}(T, n)$
 2: $C \leftarrow < 0, \ldots, 0 >$
    $\{C$ is counts vector of length $|S|$ initialised to 0$\}$
 3: for $i \leftarrow 1$ to $|S|$ do
 4:   for $j \leftarrow i + 1$ to $|S|$ do
 5:     if $\texttt{d}(S_i, S_j) \leq r \wedge$
        $\texttt{trivialMatch}(S_i, S_j)$= false then
 6:       $C_i \leftarrow C_i + 1$
 7:       $C_j \leftarrow C_j + 1$
 8: $\texttt{sort}(C, S)$
 9: $M \leftarrow \texttt{separate}(C, S)$
10: return $M$
```

The storing and recovery of the motif sets is omitted for clarity, but is easily achieved by retaining references to subsequences in addition to count data (see [7]).

4 Data

Section 4.1 describes the synthetic datasets we generate in order to test for statistically significant differences between the algorithms. Section 4.2 describes the electricity-usage profiles we mine for motif sets.

4.1 Synthetic Data.

We specify a parameterised data space from which datasets are drawn, and randomly generate independent datasets for a given set of parameters. The simulated data is white noise (observations of i.i.d. normally-distributed random variables with $\mu = 0$ and $\sigma = 1$) with shapes added to the noise at random intervals (see [7] for more details).

The minimum and maximum time series length, number of distinct shapes, and instances of each shape, are fixed parameters of the data, as are the length and amplitude of each instance. To generate a dataset, we randomly select one or two different shapes, and a number of instances for each shape. The shapes are added to the white noise at random locations, and do not overlap. An example of this distorted motif data is shown in Figure 5.

4.2 Electricity-usage Data.

The electrical device data originate from a trial of smart meters in 187 homes across the United Kingdom (see [10], [8]). The equipment was used to monitor the electricity consumption of each household in

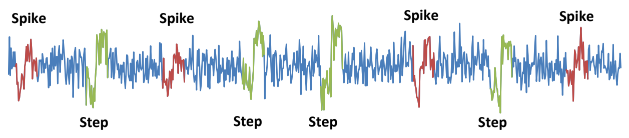

Fig. 5. An example simulated motif problem, with two motif sets, a Spike set (highlighted in red), and a Step set (highlighted in green).

Watt Hours (Wh) at 15-minute intervals. Each series corresponds to the entire consumption of a household over the duration of the trial. The motivation for using this data is that the UK government has mandated that all households must be equipped with smart metering equipment by 2020. As a consequence, there will be very large quantities of data that must be processed in an efficient manner.

One confounding factor is that devices of a similar nature have very similar usage profiles. Devices such as fridges and freezers, or computers and televisions, are very difficult to distinguish. In addition, the device-specific data is user-orientated. There is no central control over the devices that are monitored; the consumers have direct access to the monitoring equipment and all device labels are user-specified. Hence, labelling is potentially unreliable. Because of these confounding factors, it would be beneficial to have a reliable, automated method of detecting and identifying specific device use.

5 Synthetic Data Results

Our primary aim is to assess how accurately the algorithms discover motif sets; we include timing results on the companion website [7] for completeness. Cluster MK is slower than the other two algorithms on the synthetic data; Set Finder is slower on the electricity data, though the difference is marginal.

5.1 Performance Evaluation.

For synthetic data containing only one motif set, we assess performance by calculating the number of *true positives* (TP), *false positives* (FP), and *false negatives* (FN). A TP occurs when the algorithm returns an index within $\frac{n}{2}$ of any shape, where n is the length of the shape (in this case, fixed at 29). Any index returned by the algorithm that is not a TP is an FP, and any shape in the data that is not associated with a TP is an FN. We use two standard measures of accuracy: $precision = TP/(TP + FP)$, and $sensitivity = TP/(TP + FN)$.

For synthetic data containing two motif sets, the sets of indexes generated by the algorithm are paired with the indexes of the motif sets in the data, giving a combination we refer to as a *matching*. A score is calculated for each matching as follows. Each index that is not paired with another index adds the value of n (the length of the shape in the data) to the score. In our experiments, n was fixed at 29. For each pair of indexes, the absolute value of the difference between the two is calculated, with a ceiling fixed at the value of n. Hence, pairing an index of 13 with an index of 21 gives a score of 8. The scores are tallied for each possible combination of indexes within sets, and of matchings between sets. Our measure rewards close matches, and punishes false negatives and false positives equally.

To assess significance for differences between algorithms at a given value of r, we perform a two-sample T-test with an alpha value of 0.05.

We compare the algorithms on two problems: finding a single set of shapes inserted into random noise, and finding two sets of shapes inserted into random noise. The second problem is more complex, and more representative of real-life applications. We use a range of r values for two reasons. First, we are interested in discovering the value of r that is appropriate for various situations. Second, we are interested in how the algorithms compare to one another over a range of values; if we used a single value for r, our results might be misleading.

5.2 r value.

For the synthetic data, all algorithms perform best when r is around $\frac{n}{2}$ for motif length n. We speculate that the high level of noise in the data prevents successful discovery of motifs with smaller values of r. For noisy data, we recommend setting r at this level as a heuristic. If r is higher than $\frac{n}{2}$ on our synthetic data, performance begins to degrade.

The electricity data is much less noisy than the synthetic data. On that data, we achieve good results with $r = \frac{n}{8}$. This equates to an r value of 3.6 for the synthetic data, a value at which the sensitivity is 0. The synthetic data is too noisy to tolerate such a low value of r. As a general rule, we suggest indexing r to n, and decreasing the value of r as the level of noise in the data decreases.

5.3 Problems with a single motif set.

For the experiments using a single shape, we tested the three algorithms over 1000 datasets containing three to five instances of the shape, using different values of r in the range $r = 5 - 25$. Figure 6 shows the results of these experiments. The statistically significant differences are listed below:

- Cluster MK is significantly more sensitive than the other algorithms in the range $r = 9 - 15$, and less sensitive in the range $r = 21 - 25$. It is less precise than the others in the range $r = 21 - 25$.
- Set Finder is significantly more precise than the other two in the range $r = 16 - 25$, and more sensitive than Scan MK in the range $r = 10 - 20$.

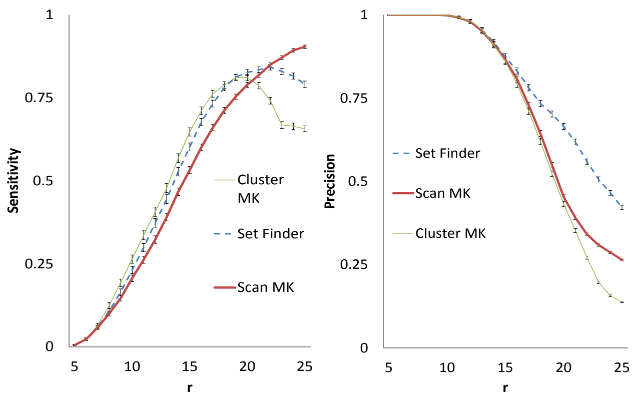

Fig. 6. The mean sensitivity (left) and precision (right), with standard error, over a random sample of 1000 instances of the single shape datasets for Scan MK, Cluster MK, and Set Finder, with values of r in the range 5 to 25.

- Scan MK is less sensitive than the other two algorithms in the range $r = 10 - 25$, and more sensitive in the range $r = 22 - 25$.

We conclude that Cluster MK and Set Finder are more accurate than Scan MK; Cluster MK is better for finding motifs, and Set Finder is better for avoiding false positives. Cluster MK has the disadvantage that it is very sensitive to the value of r, and its performance degrades quickly outside of the optimum range. Scan MK is significantly more sensitive at high values of r, but the concomitant loss of precision suggests that it will give many false positives in this range, which may be unsuitable for certain tasks.

5.4 Problems with two motif sets.

Finding multiple motif sets is more complex, not least because the algorithms must distinguish between the subsequences that belong to different sets (see Section 5.1). For the single shape problem, we permitted multiple sets to be aggregated; for the two-shape problem, each set returned by the algorithm is assigned to at most one of the motif sets in the data. Hence, an output of a single set containing all of the instances of both shapes is rewarded only for finding one motif set, and punished for missing the other set and for false positives. Equally, if the algorithm finds all instances of both motif sets, but splits them into many different sets, it is punished accordingly.

The results of the two-shape experiment are shown in Figure 7. The statistically significant differences are listed below:

- Set Finder is significantly better than Scan MK in the range $r = 11 - 15$ and significantly better than Cluster MK in the range $r = 15 - 18$.
- Cluster MK is significantly better than Scan MK in the range $r = 9 - 12$; this is reversed in the range $r = 16 - 18$.

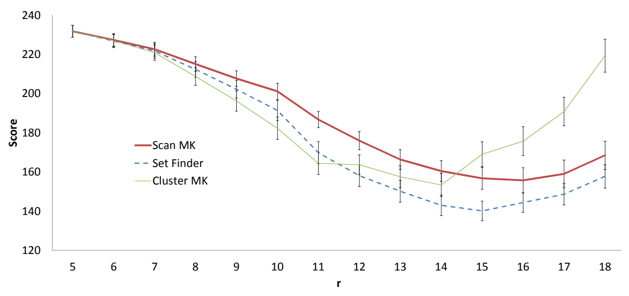

Fig. 7. The mean difference score (as defined in Section 5.1) and standard error over a random sample of 100 instances of the two shape datasets for Scan MK, Cluster MK, and Set Finder with values of r in the range 5 to 18. Lower scores indicate better performance.

- The best values of r for all algorithms are in the range $r = 14 - 16$, when r is approximately $\frac{n}{2}$.

Our results suggest that Set Finder is the most accurate algorithm when r is approximately $\frac{n}{2}$, which we suggest is an appropriate value for noisy data. Once again, Cluster MK is very sensitive to the value of r, which is a weakness of the algorithm, as small differences in r (which is difficult to estimate precisely), can cause the algorithm's performance to deteriorate.

6 Electricity-usage Data

In this section, we present qualitative analysis on the performance of Set Finder and Scan MK on household electricity-usage data. We use a window size of 4, as this represents one hour. Our results show that analysis in terms of motif sets is likely to be fruitful for profiling device usage.

We first analyse the performance of the Set Finder algorithm on a usage profile. The usage profile contains usage instances of three devices: a dishwasher, a washing machine, and an oven. Using the values $r = 0.5$ and $n = 4$, the algorithm returns two sets of indexes. Unsurprisingly, the larger set contains all instances of four consecutive zeros, representing the instances of no device usage. More interestingly, the other set consists of indexes that closely resemble the usage profile of the washing machine. Figure 8 shows the usage profile with the discovered motif set highlighted in red. The 2-motif set found by the algorithm correctly identifies all instances of the washing machine in this usage profile, and no other devices.

The precision is 1 (no false positives), and for the washing machine device, sensitivity is also 1 (no false negatives). The sensitivity measured over all devices is 0.21; while this may appear to be fairly poor, it is better than some of the results obtained on the synthetic data.

The electricity-usage problem has an added level of complexity because the motif sets representing different devices contain subsequences of different lengths;

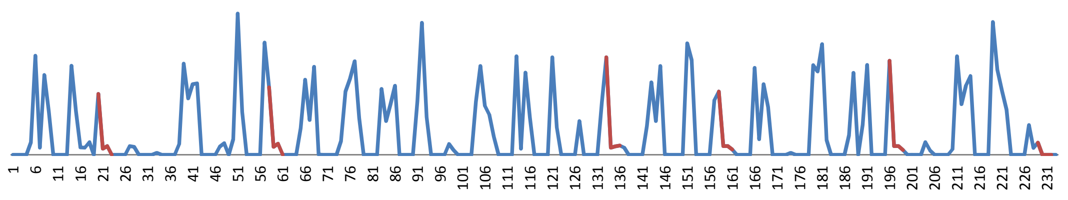

Fig. 8. A household electricity-usage profile; the subsequences returned by the Set Finder algorithm as members of the two-motif set are highlighted in red, and represent the washing machine device.

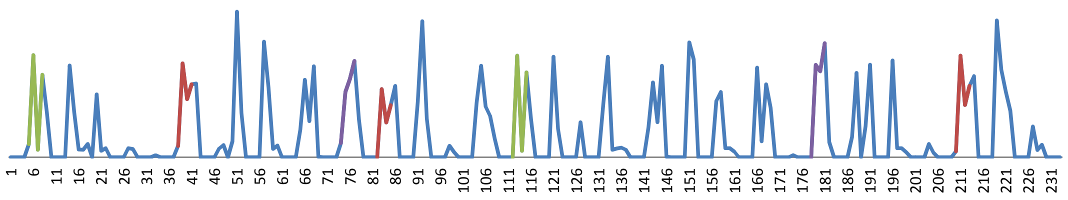

Fig. 9. A household electricity-usage profile; the subsequences returned by the Scan MK algorithm are highlighted as follows: 4-motif set (red) and 5-motif set (green), are instances of the dishwasher device. The 6-motif set (purple) is two instances of the oven device.

this necessitates variation in the values of n and r, and explains why the algorithm found one device perfectly, while missing the others. An appropriate approach for such data would involve producing output for many values of n and r, and post-processing the discovered motif sets to find the set of devices in the data.

We turn now to the performance of the Scan MK algorithm. Again, we use fixed values of $n = 4$ and $r = 0.5$. As identified by the algorithm, the 1-motif set is largely 0 elements. As with the Set Finder algorithm, we disregard this set. We also disregard the 2-motif set and 3-motif set, as they contains very similar data that would be post-processed as belonging with the 1-motif set. The other sets of indexes returned by the algorithm are interpretable as follows. The algorithm has been reasonably successful at finding the dishwasher device, although post-processing would be required to combine the 4-motif set (highlighted in red on Figure 9) and the 5-motif set (highlighted on Figure 9 in green). The precision of the combined set is 1 (no false positives); the sensitivity for the dishwasher is 0.56. The overall sensitivity is 0.24; this value includes the two oven devices identified as the 6-motif set (highlighted in purple in Figure 9). It should be noted that the device-specific sensitivity of Scan MK was lower than that of Set Finder, even though Scan MK benefited from generous post-processing.

7 Conclusion

Finding motif sets in time series is essentially a form of clustering, and it is necessary to define a heuristic search technique to find motif sets, as the problem is NP-complete. We have proposed and compared three such algorithms for motif-set discovery: Scan MK, Cluster MK, and Set Finder. Extensive experimentation shows that Set Finder is significantly more accurate than Scan MK on synthetic data containing one and two shapes, for the values of the range parameter r for which the algorithms perform best. Cluster MK is competitive providing that appropriate values of r are used; however, it is very sensitive to the value of r.

Finding motif sets is applicable to problems in a wide range of domains, including medicine, image processing, and robotics. We have extended our experiments to investigate the problem of profiling device usage from household electricity-consumption data. We found the motif set approach showed promise for identifying specific devices from data; we can reasonably expect to improve this performance dramatically on less aggregated data, and by using varying values of n and r followed by post-processing.

References

[1] J. Lin, E. Keogh, S. Lonardi, and P. Patel, "Finding motifs in time series," in *Proc. of the 2nd Workshop on Temporal Data Mining*, 2002, pp. 53–68.

[2] G. Das, K. Lin, H. Mannila, G. Renganathan, and P. Smyth, "Rule discovery from time series," *Knowledge Discovery and Data Mining*, pp. 16–22, 1998.

[3] T. Oates, M.D. Schmill, and P.R. Cohen, "A method for clustering the experiences of a mobile robot that accords with human judgments," in *Proc. of the* 17^{th} *National Conference on Artificial Intelligence*, 2000, pp. 846–851.

[4] J. Froehlich, E. Larson, G. Sidhant, G. Cohn, M. Reynolds, and P. Shwetak, "Disaggregated end-use energy sensing for the smart grid" in *Pervasive Computing, IEEE*, vol. 10, no. 1, pp. 28–39, 2011.

[5] A. Mueen, E. Keogh, Q. Zhu, S. Cash, and B. Westover, "Exact discovery of time series motifs," in *Proceedings of the SIAM international conference on data mining*, 2009, pp. 473–484.

[6] A. Mueen, E. Keogh, Q. Zhu, S. Cash, M. Westover, and N. Bigdely-Shamlo, "A disk-aware algorithm for time series motif discovery," *Data Mining and Knowledge Discovery*, vol. 22, no. 1, pp. 73–105, 2011.

[7] `https://sites.google.com/site/timeseriesmotifsets/`.

[8] A. Bagnall, L. Davis, J. Hills, and J. Lines, "Transformation based ensembles for time series classification," in *Proceedings of the SIAM International Conference on Data Mining*, pp. 307–319.

[9] E. Keogh and J. Lin, "Clustering of time-series subsequences is meaningless: implications for previous and future research," *Knowledge and Information Systems*, vol. 8, no. 2, pp. 154–177, 2005.

[10] J. Lines, A. Bagnall, P. Caiger-Smith, and S. Anderson, "Classification of household devices by electricity usage profiles," in *Intelligent Data Engineering and Automated Learning*, 2011, pp. 403–412.