# MotiPlus and MotiSet: Discovering the Best Set of Motiflets in Time Series

Len Feremans<sup>1</sup>, Patrick Schäfer<sup>2</sup>, and Wannes Meert<sup>3</sup>

University of Antwerp, Department of Computer science, Antwerpen, Belgium Humboldt-Universität zu Berlin, Department of Computer Science, Berlin, Germany KU Leuven, Department of Computer Science, Leuven, Belgium len.feremans@uantwerpen.be, patrick.schaefer@hu-berlin.de, wannes.meert@kuleuven.be

Abstract. Motif discovery algorithms find repeating patterns in time series with high similarity. Many methods exist for this important task, which has numerous applications. This work is guided by the following two ambitious questions: What is the "perfect" motif? What is the "perfect" set of motifs? To answer the first question, we consider all motifs of a certain size and rank them based on a robust measure of similarity. To determine the optimal size of a motif, we assess the quality of a motif relative to a lattice of sub- and supermotifs and define two novel quality constraints. To answer the second question, we balance multiple contrastive quality criteria for a set of motifs. The set of motifs should be diverse, non-redundant, and include highly similar motifs of varying sizes. Due to the exponential search space, the exact search for the best motif and set of motifs is a major concern. We leverage the lattice structure of time series and prune most candidate motifs and sets of motifs. For discovering a set of motifs, we propose two variations. The first is based on a greedy search and filters using the aforementioned quality constraints. The second algorithm is based on A\* search, by directly measuring the quality of thousands of candidate sets of motifs. We evaluate our method qualitatively on music datasets and quantitatively on a time series motif discovery benchmark. The proposed algorithms achieve stateof-the-art results, improving precision by 27.9% and recall by 10.1%over LOCOMOTIF, and significantly outperforming strong baselines like GRAMMARVIZ, MMOTIF, and MOTIFLETS.

# 1 Introduction

Time series (TS) consist of sequences of continuous values that are typically recorded over time. In the last two decades, research has focused on the automatic discovery of motifs, i.e. identifying a set of repeating subsequences in TS [8]. Motif discovery algorithms efficiently enumerate frequently occurring patterns and provide these insights to experts, enhancing interpretability in AI systems. Furthermore, motifs are used for downstream tasks such as TS clustering, classification, segmentation, and anomaly detection, and have applications in smart devices, financial monitoring, healthcare, studying natural phenomena

in astronomy, and more [1,3]. Unlike clustering, motif discovery does not aim to comprehensively describe the entire TS; instead, it seeks to identify multiple relevant patterns, distinguishing them from the remaining TS. Many motif discovery algorithms have been proposed. A natural trade-off arises between reporting a few large (i.e., frequently repeating) motifs with lower similarity and many smaller motifs with higher similarity. Evaluating a single method is challenging because changing hyperparameters produces multiple alternative sets of motifs. A common practice is to perform a qualitative evaluation, however authors can cherry-pick parameters and present favorable examples of motifs on only a few TS. Recently, a benchmark and evaluation measure for comparing motif discovery methods was introduced, the *tsmd-benchmark* [15]. It provides a more comprehensive evaluation than existing metrics, and more challenging benchmark than earlier ones. The authors compare 11 different methods based on F1-score, precision, and recall across 14 datasets. They find that LoCOMOTIF performs best, followed by MOTIFLETS, MMOTIF, and GRAMMARVIZ [6,13,14].

In this context, we introduce two novel algorithms for identifying a high-quality set of highly similar motifs: MOTIPLUS and MOTISET. Both methods address the limitation that most motif discovery methods rely on heuristics and identify only a single set of motifs. Depending on the parameter settings and heuristics used, the quality and robustness of the resulting set can vary significantly. MOTIPLUS and MOTISET first identify local motiflets, which are motifs of a certain size with the highest overall similarity. We define two key quality constraints for motifs: k-closedness and self-sufficiency. Both constraints help determining the correct size of the motifs. Specifically, if adding a subsequence leads to only a minor increase in similarity, the motif is not k-closed. Likewise, if a motif can be decomposed into two distinguishable smaller motifs, it is not self-sufficient. MOTIPLUS uses greedy search and in Fig. 1 we present an example output of MOTIPLUS and MOTIFLETS [12]. MOTISET searches for the optimal set of motifs thereby evaluating thousands of candidate sets using A\* search [4]. Using A\* search, we prioritize sets of motifs with the highest value, allowing us to stop MOTISET at any time and report the best solution found so far. We compute the

M1.1	$\sim$	M	~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~	~~~~~
M2.1	$\sim$	$\sim$	$\sim$	$\checkmark$
M2.2				
M2.3				

Fig. 1: An example of motifs discovered by MOTIFLETS and MOTIPLUS. MOTI-FLETS identifies a single motified of size 7 (M1.1), which lacks self-sufficiency and should be split. In contrast, MOTIPLUS finds three similar local motifiets of size 4 (M2.1–3), which Motifiets cannot detect due to its one-motif-per-size limitation.

value by balancing high intra-motif similarity with low inter-motif similarity. In the search for the "perfect" motif and the "perfect" motif size, a major challenge is the *exponential* size of the search space. Given n subsequences of length lin a time series, there are  $n^k$  possible motifs of size k. For sets of motifs, the search space is even larger, with  $n^{k \cdot m}$  possible sets of motifs of size m. We design efficient algorithms that leverage the *lattice* structure of the time series to prune most candidate motifs and sets of motifs, reducing the number of possibilities by an order of magnitude. In summary, we make the following key contributions:

- We identify the best motifs, referred to as *local motifiets*, which are defined as motifs with the highest local similarity, measured by *extent*. Additionally, we introduce two new quality constraints, *k-closedness* and *self-sufficiency*, to select the best motif of varying size.
- We present a tighter *approximate* algorithm and a more efficient *exact* algorithm to discover high-quality motifs.
- We introduce a novel greedy algorithm, MOTIPLUS, which efficiently identifies the top-m non-redundant motifs across various motif sizes.
- We develop a novel A\*-based algorithm MOTISET that optimizes the selection of a diverse, non-redundant motif set with high intra-motif and low inter-motif similarity.
- We conduct a case study on music datasets and find that MOTIPLUS identifies more motifs with higher similarity, and outperforms existing motif discovery methods on detecting repeating lyrical segments.
- We demonstrate that MOTIPLUS and MOTISET achieve state-of-the-art accuracy on the *tsmd-benchmark*, attaining the highest F1 score, precision, and recall [15].

The paper is organized as follows. Section 2 reviews related work; Section 3 introduces preliminaries. Sections 4 and 5 describe MOTIPLUS and MOTISET, respectively. Section 6 presents the experimental evaluation, and Section 7 concludes.

### 2 Related work

Challenges addressed by recent motif discovery methods include: (i) eliminating hyperparameters, such as the length and distance thresholds, (ii) reporting varying-length motifs, (iii) reporting larger motifs instead of motif pairs, (iv) adopting alternatives to Euclidean distance, such as Dynamic Time Warping, and (v) clustering motifs to increase accuracy. Recent motif discovery methods include MMOTIF, GRAMMARVIZ, MOTIFLETS, LOCOMOTIF, SNIPPETS, HIME, SWAMP, SPIKELET, VALMOD, and FRM-MINER [1,6,11,12,13,14]. Many motif discovery methods have complementary quality objectives and definitions of motif representations, which makes it difficult to compare all methods.

GRAMMARVIZ, EMMA, and FRM-MINER first convert TS to discrete sequences and use discrete pattern mining to enumerate candidate motifs [11,13]. Discretisation can impede the exact discovery of high-quality continuous motifs and require carefully selection of parameters such as alphabet size and word length. Many recent methods are based on the Matrix Profile [17]. MMOTIF finds the most similar subsequence pairs under z-normalized Euclidean distance (z-ED) [6,8]. Next, larger motifs are created by selecting all subsequences within a specified radius. MOTIFLETS searches for the top-1 motif, or motiflet, with the lowest extent while varying the size of the motif. A disadvantage of MOTIFLETS is that it discovers fewer motifs and does not search recursively for non-redundant motifs. More recently, LOCOMOTIF proposed discovering varying-length motifs using Dynamic Time Warping. A disadvantage of LOCOMOTIF and MMOTIF is that they are sensitive to the distance threshold parameter, which leads to variability in the quality of the resulting set of motifs.

MOTIPLUS is related to MMOTIF because we perform a recursive search, thereby excluding regions in the TS covered by previously discovered motifs. MOTIPLUS is also related to the TS clustering that uses Minimum Description Length to evaluate motifs during search [10]. However, the self-sufficiency measure is not based on information theory. The concept of closed motifs was defined earlier, but relates motifs of different subsequence lengths, not of different sizes [9]. MOTIPLUS and MOTISET both search for motifs with low extent, similar to MOTIFLETS. Our algorithm for finding the exact motiflet is related to frequent pattern mining in transaction databases, such as APRIORI and ECLAT [18]. In contrast to the aforementioned greedy motif discovery methods, MOTISET evaluates many possible sets of motifs. The proposed value function is related to clustering algorithms such as K-MEANS and DBSCAN. However, we consider a variable number of clusters and do not aim to cluster all points in the TS [2,5].

### **3** Background and definitions

In this section, we introduce some background terminology for time series analysis and motif discovery.

### 3.1 Time series and motifs

A continuous time series T is a sequence of n real-valued measurements  $(x_1, x_2, \ldots, x_n)$ . A subsequence  $T_{i,l} = (x_i, x_{i+1}, \ldots, x_{i+l-1})$  consists of l measurements starting at index i. A subsequence  $T_{i,l}$  overlaps, or trivially matches, another subsequence  $T_{j,l}$  if  $[i-l\cdot\alpha] \leq j \leq [i+l\cdot\alpha]$  where  $\alpha \in [0,1]$ . By default, we set  $\alpha = 0.5$ , ignoring subsequences that share more than l/2 values. We enumerate all subsequences using a sliding window of size l over T, denoted as  $P^l = \{T_{i,l} \mid 1 \leq i \leq n-l+1\}$ . The normalized Euclidean distance between two subsequences  $T_{i,l}$  and  $T_{j,l}$ , with means  $\mu$  and standard deviations  $\sigma$ , is:

$$z - ED(T_{i,l}, T_{j,l}) = \sqrt{\sum_{t=1}^{l} \left(\frac{x_{i+t-1} - \mu_i}{\sigma_i + \epsilon} - \frac{x_{j+t-1} - \mu_j}{\sigma_j + \epsilon}\right)^2}$$

Where we add  $\epsilon$  to the denominator to avoid division by zero. Given a TS T and length l, a motif  $S_k$  is a subset of k non-overlapping subsequences, i.e.,  $S_k = \{T_{i_1,l}, T_{i_2,l}, \ldots, T_{i_k,l}\} \subseteq \mathcal{P}(P^l)$ , where  $k \ge 2$ . We note that there are  $\hat{n} = n - l + 1$  subsequences and  $\hat{n}^k$  motifs of size k, assuming trivial matching subsequences are ignored.

### 3.2 Motiflets

We search for the most similar motif of a given size and provide the following relevant definitions from MOTIFLETS [12]. The main idea behind MOTIFLETS is straightforward: it searches for motiflets of varying sizes up to  $k_{max}$  and selects the maximal motiflets, typically fewer than three.

**Definition 1.** Extent: The extent of a motif  $S_k$  with motif length l is defined as the maximum pairwise distance between any two subsequences in  $S_k$ , i.e.,  $extent(S_k) = max(\{z \text{-}ED(T_{i,l}, T_{j,l}) \mid (T_{i,l}, T_{j,l}) \in S_k \times S_k\}).$ 

Extent measures the cluster diameter in normalized space, and is at most twice the radius to a subsequence.

**Definition 2.** k-motiflet: A k-motiflet is the motif of size k with the lowest extent:  $S_k$  is the k-motiflet  $\iff \nexists S'_k \subseteq \mathcal{P}(P^l)$  : extent $(S'_k) < extent(S_k)$ .

The extent function is used to compare motiflets of different sizes and to tune the motif length l.

**Definition 3.** Extent function: Given a TS T and length l, let  $S_k$  be the k-motifiet. The extent function of T is defined as  $EF(k) = extent(S_k)$  where  $k \ge 2$ , i.e.  $EF^l = extent(S_2^l)$ ,  $extent(S_3^l)$ , ...,  $extent(S_{k_{max}}^l)$ .

**Definition 4.** A k-motiflet is maximal if there is an elbow point at k in the EF:

$$elbow(k) = \frac{EF(k+1) - EF(k) + \epsilon}{EF(k) - EF(k-1) + \epsilon} > \beta$$

Here,  $\epsilon$  is a small constant used to prevent division by zero, and  $\beta$  is a hyperparameter that governs the sensitivity of detecting elbow points, which we set to 1 by default.

### 4 MotiPlus: Discovering the top-m motifs in time series

In this section, we define the desirable properties of motifs and sets of motifs. Next, we introduce MOTIPLUS, a greedy algorithm designed to discover a set of m motifs. Finally, we present both an approximate and an exact algorithm for identifying motifs with the highest local similarity.

#### 4.1 Desirable properties motifs and set of motifs

We define the optimal motifs as local motiflets, which have the smallest extent among all non-redundant motifs. Next, we introduce a quality constraint on a motif and examine whether adding or removing a subsequence affects its similarity, i.e., whether the motif is *k*-closed. Additionally, we check if a motif can be decomposed into smaller submotifs, each with higher similarity, i.e., if the motif is *self-sufficient*. These concepts are illustrated in Fig. 2.

**Definition 5.** Local k-motiflet: Given a candidate motif  $S_k$  and a set of motifs **S**,  $S_k$  is the local k-motiflet if it has the lowest extent of all non-redundant motifs:

$$S_k \in \mathcal{C}_k \text{ is the local } k \text{-motiflet} \iff \nexists S'_k \in \mathcal{C}_k : extent(S'_k) < extent(S_k) \text{ where}$$
$$\mathcal{C}_k = \{S_i | S_i \in \mathcal{P}(P^l) \land |S_i| = k \land \text{ not-redundant}(S_i, \mathbf{S})\}$$

Two motifs  $S_i$  and  $S_j$  are redundant if a subsequence  $T_{k,l} \in S_i$  trivially matches with any subsequence in  $S_j$ .

We note that a known *blind spot* in MOTIFLETS is that it discovers only a single highly similar motif repeating k times. A second goal is to determine the optimal size of a motif.

**Definition 6.** Submotif and supermotif: For a motif, we define a submotif  $S' \subset S_k$  of  $S_k = \{T_{i_1,l_1}, \ldots, T_{i_k,l_l}\}$  as any proper subset of subsequences. Likewise, we define a supermotif as any proper superset.

**Definition 7.** Maximal submotif and supermotif: The maximal supermotif  $S_{k+1}$  of a motif  $S_k$  is the supermotif with the lowest extent. Formally,  $S_k \subset S_{k+1}$  is maximal  $\iff \nexists S'_{k+1} : S_k \subset S'_{k+1} \land extent(S'_{k+1}) < extent(S_{k+1})$ . A similar definition holds for submotifs.

By suppressing motifs that are not k-closed, we avoid reporting motifs that are either too small or too large, as illustrated in Fig. 2.



Fig. 2: A synthetic TS with two motifs, each of size 3, in red and green. Both motifs are local motiflets, i.e. the green motif has the lowest extent, and the red non-redundant motif the second lowest extent. Both motifs are k-closed, i.e. the extent increases significantly upon any addition of a non-trivially matching subsequence. Finally, we note that the union of both motifs of size 6 is k-closed, but not self-sufficient.

**Definition 8.** k-closed: A motif  $S_k$  is k-closed if there is a maximal submotif  $S_{k-1}$  and maximal supermotif  $S_{k+1}$  such that

$$elbow(S_k) = \frac{extent(S_{k+1}) - extent(S_k) + \epsilon}{extent(S_k) - extent(S_{k-1}) + \epsilon} > \beta$$

Another issue, illustrated in Figs. 1 and 2, occurs when motif discovery methods mistakenly combine discernible smaller motifs into a single supermotif. Therefore, we define self-sufficient motifs, inspired by a quality measure used in discrete pattern mining [16].

**Definition 9.** Self-sufficient motif: A motif S of size 4 or higher is self-sufficient if there is no partitioning of S into two disjoint subsets  $S = S_A \cup S_B$  of size 2 or higher such that  $extent(S_A) + extent(S_B) < extent(S)$ .

To verify self-sufficient motifs, we use an approximate solution based on the *Minimum Spanning Tree* (MST). The MST is a tree where each of the k subsequences in a motif represents a node, and the k nodes are connected by adding edges with the smallest distances. Using the MST enables efficient enumeration of partitions consisting of two submotifs with high similarity. A limitation of using the MST for verifying self-sufficiency is that it may result in *false positives*, where a motif that is not self-sufficient is erroneously included in the discovered set of motifs. The algorithm for verifying self-sufficiency is described in Appendix A.3. Finally, the task of enumerating the top-m non-redundant motifs within a TS is defined as follows:

**Definition 10.** Top-*m* enumeration: Given a TS T, a top-*m* enumeration aims to identify the top-*m* set of non-redundant motifs  $\mathbf{S} = \{S_1, S_2, \ldots, S_m\}$ , having high similarity.

#### 4.2 Enumerating top-m motifs in a time series

In this section, we define MOTIPLUS, a greedy algorithm for discovering the m best motifs. The algorithm iteratively searches for the best motif that: (i) has the locally lowest extent, (ii) is not redundant, (iii) satisfies the *k*-closed and self-sufficiency constraints, and (iv) has the maximal elbow value for varying size. We note that the algorithm is efficient by using an index of local motiflets, which is pruned at each iteration. It has hyperparameters l, the length of subsequence occurrences; the maximum motif size  $k_{max}$ ; and m, the maximum number of motifs to return. The worst-case time complexity is  $\mathcal{O}(k^2 \cdot n^2)$ .

Algorithm. In Algorithm 1, we present the procedures MOTIPLUS and BESTMO-TIFLET. We begin by computing the distance matrix  $D \in \mathbb{R}^{\hat{n} \times \hat{n}}$ , where  $\hat{n} = n-l+1$ (line 2). In the distance matrix, each cell  $D_{i,j}$  contains the z-ED between subsequences starting at indices i and j of length l. Next, we compute local motiflets using the algorithm described in Section 4.3. That is, we call K\_MOTIFLETS\_HEAP once with the maximum size of a motif  $(k_{max})$  and return an index with at most

### Algorithm 1: MOTIPLUS: Discovering the top-*m* motifs

```
Input: A time series T, motif length l, number of motifs m, the maximum
                motif size k_{max}
    Result: Set S with up to m motifs
    procedure MOTIPLUS(T, l, m, k_{max})
 1
          D \leftarrow \text{CALC_DISTANCE_MATRIX}(T, l)
 \mathbf{2}
          index \leftarrow K_MOTIFLETS_HEAP(T, D, l, k_{max})
 3
 \mathbf{4}
          \mathbf{S} \leftarrow \{\}
          for i \leftarrow 1 to m do
 5
                S_i \leftarrow \text{BestMotifLet}(T, index, S, k_{max})
 6
 7
               if S_i = \emptyset then
                     break
  8
                \mathbf{S} \leftarrow \mathbf{S} \cup \{S_i\}
 9
10
          return S
11
    procedure BESTMOTIFLET(T, index, \mathbf{S}, k_{max})
12
          \mathcal{C} \leftarrow \{\}
\mathbf{13}
          for k \leftarrow 2 to k_{max} do
14
                for S_k \in index[k] sorted on extent do
                     if non-redundant(S_k, \mathbf{S}) \land k\text{-}closed(S_k) \land self\text{-}sufficient(S_k) then
15
                           \mathcal{C} \leftarrow \mathcal{C} \cup \{S_k\}
16
                           break
17
                     else
18
                           index[k] \leftarrow index[k] \setminus \{S_k\}
19
\mathbf{20}
          if \mathcal{C} = \{\} then
21
               return \emptyset
          S \leftarrow argmax \ elbow(k)
22
                    \tilde{S}_k \in C
23
          return S
```

 $\hat{n} \times k_{max}$  varying-sized local motifiets. The main loop in MOTIPLUS calls the subroutine BESTMOTIFLET at most m times (lines 5-9). Early termination is possible if no additional motif is discovered (line 7). BESTMOTIFLET has two additional parameters: the index of candidate motifiets and the current set of motifs, **S**. We iterate over each value of k and retrieve candidate motifs ordered by extent from the index (lines 13-14). Next, we iterate over at most  $\hat{n}$  motifs and verify whether each motif is not *redundant* and satisfies the k-closed and self-sufficiency constraints (line 15). The first motif of size k that satisfies the constraints is added to the list of local motifiets of varying size (line 16). Motifs that do not satisfy the constraints are *pruned* from the index. Finally, we use the elbow-based heuristic to return the maximal local motifiet (line 22). As a special case, we return  $\emptyset$  if no motif satisfying the constraints is discovered.

**Time complexity.** The complexity of MOTIPLUS is dominated by calling K\_MOTIFLETS\_HEAP once, which has a complexity of  $\mathcal{O}(k^2 \cdot n^2)$ , and computing the distance matrix. The time complexity for computing the distance matrix naively is  $\mathcal{O}(l \cdot n^2)$ . Using MASS [17], this can be reduced to  $\mathcal{O}(n \cdot \log n)$ . In the

worst case, we verify the constraints of all candidates in the index, regardless of the value of m, where there are at most  $\mathcal{O}(n \cdot k)$  candidates in the index. The overall worst-case complexity is thus given by  $\mathcal{O}(k^2 \cdot n^2 + n \cdot k \cdot n) = \mathcal{O}(k^2 \cdot n^2)$ .

### 4.3 Discover the motif with the overall highest similarity

In this section, we discuss both an approximate and exact algorithm for local motified discovery. The approximate algorithm directly optimizes extent, resulting in higher fidelity to the exact solution, and returns many motifs of the same size with the lowest extent. The exact algorithm applies admissible pruning of supermotifs based on the *monotonicity of extent* inspired by related work in discrete pattern mining [18]. We note this is the first feasible solution that considers all  $n^k$  motifs.

A tighter approximate k-motifiets algorithm For local motifiets discovery we return a list of n motifs of size k ranked by extent. We use greedy search with extent as a heuristic. We note that greedy search evaluates  $k \times \hat{n}$  subsequence candidates, whereas related work focuses on the non-trivially matching nearest subsequences [6,12]. An advantage is that by evaluating extent we take all pairwise distances into account, leading to lower extent. An interesting property of greedy search, is that any submotif of size  $2, \ldots, k_{max} - 1$  also has the lowest extent. Hence, we only have to search for the local motifiets once with size  $k_{max}$ . For brevity, we discuss the pseudo-code in Appendix A.1. The proposed greedy search optimization using extent requires  $\mathcal{O}(k^2 \cdot n)$  time and the total complexity is  $\mathcal{O}(k^2 \cdot n^2)$ . In contrast, the nearest-neighbor search has a complexity of  $\mathcal{O}(k \cdot n^2)$ .

An efficient exact k-motiflets algorithm Next, we present a novel exact algorithm for the discovery of k-motiflets. The main goal is to design an algorithm that is efficient in time and space by applying pruning using the lattice of motifs. We represent a motif  $S_k = \{T_{i_1,l}, T_{i_2,l}, \ldots, T_{i_k,l}\}$  as  $\hat{S}_k = \{i_1, i_2, \ldots, i_k\}$  where each subsequence  $T_{i,l}$  is represented by its starting index  $i \in [1, \hat{n}]$ . Using this representation we enumerate all subsets of size  $1, 2, \ldots, k$ , and finally return the subset of size k having the lowest extent. The following theorem is used to prune motifs during exact search.

**Theorem 1.** Given a motif  $\hat{S}'$  and supermotif  $\hat{S}$ , we find that extent is monotonically increasing, that is, for all  $\hat{S} : \hat{S}' \subset \hat{S}$ :  $extent(\hat{S}') \leq extent(\hat{S})$ .

**Proof:** The proof is trivial, i.e.  $extent(\hat{S}) = max(z - ED(T_{a,l}, T_{b,l}) | a, b \in \hat{S}\}) = max(\{z - ED(T_{a,l}, T_{b,l}) | a, b \in \hat{S}'\} \cup \{z - ED(T_{a,l}, T_{b,l}) | a \in \hat{S} \setminus \hat{S}' \land b \in \hat{S}\}) \ge extent(\hat{S}').$ 

We leverage *depth-first search* to create candidate motifs of growing size bottomup and prunes all supermotifs based on Theorem 1. We initialise the lower bound of extent using the approximate solution. If any motif subset (possible of small size) already has an extent higher than the approximate solution, we do not have to evaluate any supermotif. Additionally, we make use of a second theorem for initial pruning of motif pairs having a large distance. For brevity, the pseudo-code is discussed in Section A.2.

### 5 MotiSet: Discover the best set of motifs

One disadvantage of most motif discovery methods is that greedy search is suboptimal. In contrast, MOTISET searches for an optimal solution using A<sup>\*</sup> and measures the value of thousands of candidate sets of motifs. To evaluate a set of motifs, we propose a new value function that balances contrasting aspects of motifs. MOTISET has the same hyper-parameters as MOTIPLUS, however the maximal number of motifs to return (m) is optional. In theory, the worst-case complexity of MOTISET is  $O((\hat{n} \cdot k_{max})^m)$ . However, we prune redundant motifs thereby achieving an order-of-magnitude reduction in the number of possible candidate sets. Moreover, the A<sup>\*</sup> search is an anytime algorithm and we suggest to stopping after a fixed number of candidate sets of motifs have been evaluated.

**Definition 11.** Optimal set of motifs: Given a set of candidate motifs  $S \subseteq \mathcal{P}(P^l)$ , and a value function f, we define that the set of motifs  $\mathbf{S}$  is optimal  $\iff \forall \mathbf{S}' \subseteq S : f(\mathbf{S}) \leq f(\mathbf{S}')$ .

**Definition 12.** For a set of motifs,  $\mathbf{S} = \{S_1, S_2, \ldots, S_m\}$ , we seek high intramotif and low inter-motif similarity. We define the value as the difference between the sum of the minimal pairwise distances (smpd), and the sum of the minimal outer distances (smod):

$$f(\mathbf{S}) = norm(smpd(\mathbf{S})) - \gamma \cdot norm(smod(\mathbf{S}))$$

$$smpd(\mathbf{S}) = \frac{\sum_{k=1}^{m} \sum_{i=1}^{|S_k|} min\_dist\_in(T_{i,l}, S_k)^2}{\sum_{k=1}^{m} |S_k|}$$

$$smod(\mathbf{S}) = \sum_{k=1}^{m} min\_dist\_out(S_k, T)^2$$

$$min\_dist\_in(T_{i,l}, S_k) = min(\{z\text{-}ED(T_{i,l}, T_{j,l}) \mid T_{j,l} \in S_k \land T_{i,l} \neq T_{j,l}\})$$

$$min\_dist\_out(S_k, T) = min(\{z\text{-}ED(T_{i,l}, T_{j,l}) \mid T_{i,l} \in S_k \land T_{i,l} \in T\})$$

For smaller motifs the intra-motif similarity (smpd) will be high, while for larger motifs that inter-motif similarity (smod) will be low. The rationale behind the value function is that we want larger motifs, at least if they are semantically similar. Smpd compute the squared minimal distance between each subsequence  $T_{i,l}$  and every other subsequence  $T_{j,l}$  within each motif. The rationale for aggregating minimal distances inter-motif is that the extent (and radius) ignores the variability in distances within a motif. We normalize the smpd values by the total number of subsequences in the set of motifs. For smod we compute the minimal distance between any subsequence  $T_{i,l}$  in a motif and any non-trivially matching subsequence  $T_{j,l}$  outside the motif. Finally, we normalize the smpd and smod values across all candidate sets of motifs between 0 and 1. Based on our experiments, we found that setting  $\gamma = 0.2$  yields good performance.

**Algorithm.** Using A<sup>\*</sup> search we prioritize the expansion for sets of motifs with the current lowest value. To limit the search space, we start with local motiflets,

which are restricted to  $\hat{n} \times k_{max}$  motifs (see Section 4.3). In the first phase, vertical pruning is applied, because many local motifiets of the same size are redundant, i.e., they share trivially matching subsequences. During search, we apply horizontal pruning, because many motifs of different sizes are submotifs, supermotifs or partially overlapping. Additionally, we enforce an ordering between motifs to avoid enumeration of all possible orders of motifs within each candidate set. By combining both types of pruning, we achieve an order-of-magnitude reduction in the number of possible candidate sets of motifs. For brevity, we define the pseudo-code for the A\* algorithm in Appendix A.4.

### 6 Experiments

In this section, we answer the following research questions<sup>1</sup>: Q1: What is the quality of the approximate algorithm and the runtime of the exact algorithm for discovering motiflets? Q2: How does MOTIPLUS compare to existing motif discovery methods qualitatively on music datasets? Q3: How do MOTIPLUS and MOTISET compare to existing motif discovery methods on the tsmd-benchmark?

# 6.1 The runtime and the quality for discovering motiflets using the exact and approximate algorithm (Q1)

In the first experiment, we measure the quality of the approximate solution and the runtime of the exact algorithm against the brute-force algorithm. We use a benchmark dataset of 12 pairs of univariate TS of the same class from [7].

Quality of the approximate algorithm. We compare the approximate algorithm proposed in Section 4.3 with the original algorithm from MOTIFLETS which finds the nearest subsequences to the core. The goal of both methods is to achieve high fidelity to the exact solution at a fraction of the cost. We measure the quality of the approximate solutions as the *ratio* of the extents of the approximate to the exact solution (i.e. values close to 1 indicate the extent is close to the exact solution). The motif length  $l \in [25, 200]$  is selected based on the minimum area under the curve of the extent function, and we set k to 7. The proposed approximate algorithm discovers a k-motiflet in all 12 time series, achieving an extent ratio of 0.9 or higher. The approximate algorithm from [12] achieves an extent ratio of 0.9 or higher in only 8 out of 12 time series.

**Runtime of the exact algorithm.** We compare the runtimes for discovering the exact k-motifiet using the algorithm proposed in Section 4.3 against the brute-force algorithm from MOTIFLETS. We vary k between 2 and 15 and stop execution if computation exceeds 30 minutes. As expected, the brute-force algorithm fails for relatively small values of k (as low as 6) due to the exponential number of candidates. In contrast, the exact algorithm based on depth-first search completes

<sup>&</sup>lt;sup>1</sup> MOTIPLUS and MOTISET are implemented in Python; motiflet search is in Java. Data and code are available at https://bitbucket.org/len\_feremans/kmotiflets

execution within 30 minutes for all TS for k up to 15. We conclude that the proposed exact algorithm is an *order-of-magnitude* more efficient and feasible to run in practice for reasonable values of k. However, given the additional computational cost, and the high quality of the approximate method, we prefer the approximate method in further experiments. For brevity, we report detailed plots in Appendix A.5.

# 6.2 How does MotiPlus qualitatively compare to existing motif discovery methods on music data? (Q2)

In this experiment, we compare MOTIPLUS with MMOTIF, LOCOMOTIF, and MOTIFLETS. We collected six songs with synchronized lyrics from an online audio streaming platform. We evaluate the discovered set of motifs by measuring the *Jaccard similarity* between each ground truth (GT) motif and its best-matching counterpart.

**Experimental setup.** We preprocess raw audio samples using the second Mel-Frequency Cepstral Coefficient (MFCC) channel sampled at 100Hz [17]. Each song is about four minutes long. After preprocessing, a time series (TS) consists of approximately n = 50,000 values, which is large. Each song contains between 3 and 12 repeating lyric segments synchronized to the TS, which we use as GT motifs. For MMOTIF, we use the implementation from STUMPY [6]. For each method, we selected the best parameters using grid search. For MMOTIF, the radius is set to 2. For LoCOMOTIF, we set the minimum and maximum lengths to the same value, *rho* to 0.5, and *warping* to False. For MOTIPLUS and MOTIFLETS, we set the maximum motif size  $k_{max}$  to 15. For all methods we set the number of motifs, m, to 50. We select the motif length  $l \in [100, 400]$  based on the minimum area under the curve of the extent function.

**Results.** In Table 1, we report the mean Jaccard similarity with ground-truth lyrics and statistics such as the mean extent, and coverage, the percentage of time series values covered by motif subsequences. We find that MOTIPLUS has a higher Jaccard similarity than MMOTIF on four out of six datasets. MMOTIF achieves higher accuracy on two datasets. However, MMOTIF primarily searches for top motif pairs and only secondarily for larger motifs, resulting in lower extent. MOTIFLETS has the lowest average extent, which we expect since it finds the top-1 motiflets. However, it fails to detect many GT motifs. For LOCOMOTIF, the accuracy is quite low, and the average extent is highest, which is surprising. We inspected whether discovered motifs align with meaningful lyrical or instrumental phrases. MOTIPLUS generally produces motifs that better follow musical structure. Its coverage is also more evenly distributed across songs. In contrast, LOCOMOTIF and MMOTIF sometimes return repetitive background patterns—such as sustained chords or rhythmic noise—that do not correspond to meaningful musical or lyrical segments<sup>2</sup>. Regarding run-time, we find that MMOTIF and MOTIFLETS are the

 $<sup>^2</sup>$  For brevity, we report all discovered motifs, including instrumental motifs, on our website

Dataset	LoCoMotif	MMotif	Motiflets	MotiPlus
Ice ice baby	0.118	0.336	0.092	0.271
Beverly hills	0.185	0.538	0.331	0.570
La Isla Bonita	0.105	0.451	0.069	0.326
Billie Jean	0.123	0.000	0.011	0.442
The Chain	0.085	0.175	0.012	0.204
The Pretender	0.178	0.275	0.152	0.358
Mean Jaccard Sim.	0.132	0.296	0.113	0.362
Mean extent	17.4	13.0	<b>7.6</b>	10.0
Mean coverage	45.4%	83.2%	15.3%	76.4%

Table 1: Comparing the mean *Jaccard similarity* between known lyrics and discovered counterpart motifs for each method on the music datasets. We also report the extent and coverage averaged across all datasets.

fastest, requiring approximately 30 seconds per dataset. The bottleneck is the computation of the distance matrix, which is required by all methods. MOTIPLUS requires approximately 60 seconds per dataset. Finally, we note that LoCoMOTIF is somewhat slower and has a high memory consumption, taking several minutes and exceeding the 16GB limit.

# 6.3 How do MotiPlus and MotiSet compare to existing motif discovery methods on the tsmd-benchmark? (Q3)

We adopt the evaluation metrics, labeled TS datasets, and experimental setup from the *tsdm-benchmark* [15]. We compare MOTIPLUS and MOTISET with the four best-performing methods in the *tsdm-benchmark*, namely LOCOMOTIF, MMOTIF, GRAMMARVIZ, and MOTIFLETS<sup>3</sup>. We evaluate on TS datasets with fixed-length GT motifs namely Ecg5000, Fungi, Mallat, Plane and Symbols. Each dataset consists of 200 TS instances with a variable number of GT motifs that differ in length and size. We evaluate the F1 score, precision and recall using the PROM matrix. We penalize off-target motifs, i.e., methods that discover more motifs than are present in the GT set.

**Experimental setup.** We make the following adjustments to the experimental setup in the *tsdm-benchmark*. We include a Mixed dataset, which constitutes a heterogeneous collection by combining TS instances from the aforementioned datasets (excluding Mallat due to its longer motif length). We combine test and validation TS instances and add noise to TS segments not covered by any GT motif resulting in TS datasets having a longer length and a larger set of GT motifs, making the task more challenging <sup>4</sup>. We tune parameters using grid search, thereby using the last 50 TS instances for validation and report the accuracy on

<sup>&</sup>lt;sup>3</sup> We do not compare with motif discovery methods that performed worse, namely EMMA, MRMOTIF, SETFINDER, LATENTMOTIFS, VALMOD, and VONSEM

<sup>&</sup>lt;sup>4</sup> The generator from the tsmd-benchmark creates motifs based on different TS classes within each TS dataset. However, it uses distinct subsets of the available classes to generate test and validation instances.



Fig. 3: The distribution of F1-scores for each method on the tsmd-benchmark datasets shows that either LOCOMOTIF or MOTISET performs best on each dataset. MOTIPLUS improves upon MOTIFLETS. GRAMMARVIZ and MMOTIF perform significantly worse.

the first 50 TS instances. For LoCOMOTIF, we set *warping* to True and search for the optimal value of *rho*. For GRAMMARVIZ and MMOTIF we search for the best parameters as suggested in [15]. For MOTIFLETS, MOTIPLUS and MOTISET we optimize the maximal motif size  $k_{max}$  within the range  $[k_{gt} - 3, k_{gt} + 3]$ , where  $k_{gt}$  is the maximum size of GT motifs. For MOTISET we set the maximum number of iterations to 10 000. Finally, we set the motif length l and number of motifs m to match the GT length and maximum count.

**Results.** Our study replicates the findings of the original study. We report two major differences in reported accuracy. First, we observe a much lower accuracy for GRAMMARVIZ. GRAMMARVIZ typically discovers a much larger set of motifs than other methods since it does not have a parameter m, resulting in an unfair

Dataset	F1	Precision	Recall
GRAMMARVIZ MMOTIF MOTIFLETS LOCOMOTIF	$\begin{array}{c} 0.234 \ (+-0.097) \\ 0.314 \ (+-0.058) \\ 0.498 \ (+-0.104) \\ 0.565 \ (+-0.232) \end{array}$	$\begin{array}{c} 0.264 \ (+-0.196) \\ 0.400 \ (+-0.106) \\ 0.719 \ (+-0.125) \\ 0.604 \ (+-0.268) \end{array}$	$\begin{array}{c} 0.321 \ (+-0.088) \\ 0.323 \ (+-0.061) \\ 0.403 \ (+-0.094) \\ 0.564 \ (+-0.195) \end{array}$
MotiPlus MotiSet	0.552 (+-0.094) <b>0.586</b> (+-0.097)	0.543 (+-0.125) <b>0.773</b> (+-0.068)	<b>0.621</b> (+-0.087) 0.531 (+-0.108)

Table 2: We report the average F1-score, precision, recall, and standard deviation across six datasets from the tsmd-benchmark. We find that MOTISET achieves the highest F1-score and precision, while MOTIPLUS attains the highest recall.

advantage, which we correct by penalizing off-target motifs. Secondly, we find that tuning  $k_{max}$  increases the accuracy of MOTIFLETS substantially. The average F1 score, precision, and recall across all datasets are shown in Table 2. We find that MOTISET achieves the highest overall F1 score and precision. MOTIPLUS achieves the highest overall recall, but has lower precision than LOCOMOTIF and MOTIFLETS. In Figure 3, we show the distribution of F1 scores for each dataset. We find that LOCOMOTIF performs best on Plane, Fungi, and Symbols, while MOTISET performs best on Ecg5000, Mallat, and Mixed, suggesting that both methods are complementary in terms of F1 score. We argue that the increased performance of LOCOMOTIF on certain datasets, such as Plane, is likely due to the use of Dynamic Time Warping, which is known to be more accurate than Euclidean distance on certain TS datasets [1]. We find that MOTIPLUS improves on LOCOMOTIF in 3 out of 6 datasets and in 5 out of 6 datasets compared to MOTIFLETS. We observe that the increase in accuracy compared to MOTIFLETS is correlated with the number of GT motifs in each dataset, i.e., Fungi has the most motifs. MOTIPLUS requires setting the maximum motif size, however this parameter mainly affects runtime. In contrast, LOCOMOTIF and MMOTIF rely on sensitive distance thresholds, which vary across datasets and hinder reproducibility, leading to high variability on the Mixed dataset. Concerning runtime, we find that GRAMMARVIZ, MMOTIF, MOTIFLETS, and MOTIPLUS are generally faster, completing each dataset under two minutes. The runtime of LOCOMOTIF is slightly longer, taking approximately 10 minutes. Finally, we note that MOTISET requires about 30 minutes. The runtime varies significantly for each time series (TS) instance and depends on the number of iterations, which ranges from 100 to the upper limit of 10,000.

### 7 Conclusion

In this paper, we presented MOTIPLUS and MOTISET, two novel motif discovery methods that identify the top-*m* motifs in time series data across varying motif sizes, ensuring high internal similarity. Both methods are supported by efficient algorithms for scalable motif discovery. We approximate local motifiets using extent as a heuristic and propose an exact method for k-motiflet discovery that is significantly faster than existing approaches. MOTIPLUS iteratively searches for diverse, high-quality motifiets by filtering for k-closed and self-sufficient motifs, while MOTISET uses A<sup>\*</sup> search to optimize motif set selection. We qualitatively evaluated MOTIPLUS on six music datasets with synchronized lyrics as ground truth and found it uncovers lyrical segments missed by MOTIFLETS, MMOTIF, and LOCOMOTIF. On the tsmd-benchmark, both methods outperform state-ofthe-art baselines, with MOTISET and MOTIPLUS improving precision by 27.9% and recall by 10.1% over LOCOMOTIF. While MOTIPLUS runs comparably to baselines, MOTISET is up to three times slower depending on the number of iterations. In future work, we aim to explore variable-length motiflets, improve robustness, and study ensemble methods and their use in downstream time series tasks.

Acknowledgements L.F. is funded on project 12B0V24N by Research Fund Flanders.

### References

- Alaee, S., Mercer, R., Kamgar, K., Keogh, E.: Time series motifs discovery under dtw allows more robust discovery of conserved structure. Data Mining and Knowledge Discovery 35, 863–910 (2021)
- 2. El-Sonbaty, Y., Ismail, M.A., Farouk, M.: An efficient density based clustering algorithm for large databases. In: 16th IEEE international conference on tools with artificial intelligence. pp. 673–677. IEEE (2004)
- Feremans, L., Cule, B., Goethals, B.: Petsc: pattern-based embedding for time series classification. Data Mining and Knowledge Discovery 36(3), 1015–1061 (2022)
- Hart, P.E., Nilsson, N.J., Raphael, B.: A formal basis for the heuristic determination of minimum cost paths. IEEE transactions on Systems Science and Cybernetics 4(2), 100–107 (1968)
- 5. Jain, A.K., Dubes, R.C.: Algorithms for clustering data. Prentice-Hall, Inc. (1988)
- Law, S.M.: Stumpy: A powerful and scalable python library for time series data mining. Journal of Open Source Software 4(39), 1504 (2019)
- Lin, J., Khade, R., Li, Y.: Rotation-invariant similarity in time series using bag-ofpatterns representation. Journal of Intelligent Information Systems **39**, 287–315 (2012)
- Lonardi, J., Patel, P.: Finding motifs in time series. In: 2nd Workshop on Temporal Data Mining. pp. 53–68 (2002)
- Nguyen, H.L., Ng, W.K., Woon, Y.K.: Closed motifs for streaming time series classification. Knowledge and information systems 41(1), 101–125 (2014)
- Rakthanmanon, T., Keogh, E.J., Lonardi, S., Evans, S.: Time series epenthesis: Clustering time series streams requires ignoring some data. In: 2011 IEEE 11th international conference on data mining. pp. 547–556. IEEE (2011)
- Rotman, S., Čule, B., Feremans, L.: Efficiently mining frequent representative motifs in large collections of time series. In: BigDataprocee. pp. 66–75. IEEE (2023)
- Schäfer, P., Leser, U.: Motiflets: Simple and accurate detection of motifs in time series. PVLDB 16(4), 725–737 (2022)
- Senin, P., Lin, J., Wang, X., Oates, T., Gandhi, S., Boedihardjo, A.P., Chen, C., Frankenstein, S.: Grammarviz 3.0: Interactive discovery of variable-length time series patterns. TKDD 12(1), 1–28 (2018)
- Van Wesenbeeck, D., Yurtman, A., Meert, W., Blockeel, H.: Locomotif: Discovering time-warped motifs in time series. Data Mining and Knowledge Discovery pp. 1–30 (2024)
- Van Wesenbeeck, D., Yurtman, A., Meert, W., Blockeel, H.: Quantitative evaluation of motif sets in time series. arXiv preprint arXiv:2412.09346 (2024)
- 16. Webb, G.I.: Self-sufficient itemsets: An approach to screening potentially interesting associations between items. TKDD 4(1), 1–20 (2010)
- Yeh, C.C.M., Zhu, Y., Ulanova, L., Begum, N., Ding, Y., Dau, H.A., Zimmerman, Z., Silva, D.F., Mueen, A., Keogh, E.: Time series joins, motifs, discords and shapelets: a unifying view that exploits the matrix profile. Data Mining and Knowledge Discovery **32**, 83–123 (2018)
- 18. Zaki, M.J., Meira, W.: Data mining and analysis: fundamental concepts and algorithms. Cambridge University Press (2014)

# A Appendix

### A.1 A tighter approximate k-motiflets algorithm

We discuss the pseudo-code of the proposed approximate local motified discovery algorithm. The algorithm uses a greedy search and evaluates candidate motifiets using extent, while MOTIFLETS uses a radius-based metric with leads to lower-quality results as illustrated in Fig. 4.

Algorithm. The procedure K\_MOTIFLETS\_HEAP is shown in Algorithm 2. We represent a motif  $S_k = \{T_{i_1,l}, T_{i_2,l}, \ldots, T_{i_k,l}\}$  as  $S_k = \{i_1, i_2, \ldots, i_k\}$  where each subsequence  $T_{i,l}$  is represented by its starting index. The output  $\mathcal{R}$  consists of at most  $\hat{n} = n - l + 1$  motifs ranked by extent to enable *local* k-motiflet discovery (see Section 4.2). We start by iterating  $\hat{n}$  times over each possible subsequence i (line 3). In each iteration, we aim to find the k - 1 subsequences that result in the lowest maximal pairwise distance (lines 4–10). During the k - 1 greedy search iterations, we select the next candidate subsequence that minimises extent and discard trivial-matching subsequences (lines 7-8). We filter out duplicate k-motiflets, i.e. different cores leading to an identical motif set (line 10). Finally, we return the list of k-motiflets sorted by extent.

**Time complexity.** The main loop is executed  $\mathcal{O}(n)$  times. Greedy search optimising the *extent* requires  $\mathcal{O}(k^2 \cdot n)$ -time. The total complexity is  $\mathcal{O}(k^2 \cdot n^2)$ .

Algorithm 2: Discover Local k-Motiflets		
<b>Input</b> : A time series $T$ , distance matrix $D$ , motif length $l$ , motif size $k$		
<b>Result:</b> List of at most $\hat{n}$ motifs sorted by extent.		
1 procedure K_MOTIFLETS_HEAP $(T, D, l, k)$		
$2     \mathcal{R} \leftarrow \{\}$		
3 for $i \leftarrow 1$ to $n - l + 1$ do		
$4 \qquad   \qquad S \leftarrow \{i\}$		
5 $\mathcal{C} \leftarrow \text{REMOVE\_TRIVIAL\_MATCHES}(\{1, 2, \dots, n - l + 1\}, i)$		
6 for $iter \leftarrow 2$ to k do		
7 $j_k \leftarrow argmin \ extent(S \cup \{j\})$		
$j \in \mathcal{C}$		
8 $\mathcal{C} \leftarrow \text{REMOVE\_TRIVIAL\_MATCHES}(\mathcal{C}, j_k)$		
9 $S \leftarrow S \cup \{j_k\}$		
10 $\mathcal{R} \leftarrow \mathcal{R} \cup \{S\}$		
11 $\mathcal{R} \leftarrow sort \mathcal{R} \text{ on extent}$		
12 return $\mathcal{R}$		

# A.2 An efficient exact k-motiflets algorithm

We present the pseudo-code of an exact algorithm for discovering k-motiflets. First, we introduce a second theorem. **Theorem 2.** For each candidate motif of size  $k, S_k = \{T_{i,l}, \ldots, T_{k,l}\}$  with extent d the following holds: if  $|\{T_{j,l} \mid T_{j,l} \in T : z\text{-}ED(T_{i,l}, T_{j,l}) < d\}| < k \iff extent(S_k) > d.$ 

**Proof**: If fewer than k non-trivially matching subsequences have a distance to the core  $T_{i,l}$  smaller than d, then at least one subsequence in the candidate motif  $S_k$  has a distance larger than d. Consequently, the radius to the core and the maximal pairwise distance, or extent, must have a distance greater than d.

The exact algorithm employs *depth-first search* to construct candidate motif sets incrementally from the bottom up, pruning supermotifs based on Theorem 1. Additionally, we apply Theorem 2 for initial pruning. For simplicity, we assume the algorithm discovers the top-1 k-motiflet.

Algorithm. The main and auxiliary procedure are shown in Algorithm 3. First, we initialise the best motiflet and its extent d using the approximate k-motiflet solution (lines 2-3). Next, we initiate a recursive depth-first search over all motifs smaller or equal to size k (line 5). During recursion, we track the current candidate motif S, its extent, and the set C of candidate subsequences that can be added to S. In MOTIFLETS\_DFS, we first check if the size of S is k and if its extent is smaller than the current best extent, thereby possibly improving on the approximate solution and decreasing the upper bound for pruning future candidates (lines 9-10). Next, we prune candidate motifs of size  $\leq k$  if the current extent is greater or equal than d using Theorem 1 (line 12). For singleton motifs we update C by filtering candidate subsequences with a distance less than d and apply Theorem 2 for pruning (lines 14-17) Next, we get the next candidate subsequence i from C that does not trivially match any subsequence in S (line 18). We then compute



Fig. 4: Approximate k-motified discovery using the same 4 data points and either (left) radius to the core (right) extent of the set. Left: For a core subsequence (in red) the two nearest subsequences are retrieved [12]. Right: For a core subsequence we perform greedy search such that the maximal pairwise distance is minimal. Bottom: The resulting 3-motified on the right has a lower extent.

```
Algorithm 3: Discover Exact k-Motiflets
    Input: A time series T, distance matrix D, motif length l, motif size k
    Result: k-motiflet, extent
 1 procedure K_MOTIFLETS_EXACT(T, D, l, k)
         L \leftarrow \text{K_MOTIFLETS_HEAP}(T, D, l, k)
 2
 3
         (motiflet, d) \leftarrow L[0]
 \mathbf{4}
         \mathcal{C} \leftarrow \{1, 2, \dots, n - l + 1\}
         MOTIFLETS_DFS(\{\}, inf, C)
 \mathbf{5}
         return (motiflet, d)
 6
 \mathbf{7}
    procedure MOTIFLETS_DFS(S, extent, C)
 8
         if |S| = k then
 9
              if extent < d then
                                                          // Update lower bound for extent
10
                   (motiflets, d) \leftarrow (S, extent)
         else
11
\mathbf{12}
              if extent \ge d or |\mathcal{C}| = 0 then
                                                                       // Prune using theorem 1
13
                   return
14
              if |S| = 1 then
                   \mathcal{C} \leftarrow \{T_{i,l} \mid T_{i,l} \in T : z\text{-}ED(T_{i,l}, S[0]) < d\}
15
                   if |\mathcal{C}| < k then
                                                                        // Prune using theorem 2
\mathbf{16}
                       continue
17
18
              i \leftarrow \text{NEXT}_{\text{INDEX}}(\mathcal{C})
              inc\_extent \leftarrow max(\{D_{i,j} \mid \forall j \in S\} \cup \{extent\})
19
              \texttt{MOTIFLETS\_DFS}(S \cup \{i\}, \textit{inc\_extent}, \mathcal{C} \backslash \{i\})
20
              MOTIFLETS_DFS(S, extent, C \setminus \{i\})
21
```

the extent of  $S \cup \{i\}$  incrementally (line 19). Finally, we recursively search for all supermotifs with and without the next non-trivially matching subsequence in C (lines 20-21).

**Time complexity.** The worst-case time complexity is  $\mathcal{O}(n^k)$ , assuming trivial matches are ignored. The algorithm is memory-efficient by keeping at most k motifs in memory. In Section 6, we demonstrate that the algorithm is an order of magnitude faster.

#### A.3 An approximate algorithm for verifying self-sufficient motifs

To verify self-sufficient motifs, we employ an approximate solution based on the Minimum Spanning Tree (MST), thereby avoiding the costly generation of all possible subsets of a motif.

Algorithm. We begin by constructing an MST using Prim's algorithm, which adds edges between nodes one by one while avoiding cycles. In this MST, each node represents a subsequence, and each edge corresponds to the z-normalized Euclidean distance between subsequences. Next, we generate the k-1 possible partitions of the MST by removing each edge in turn, ensuring that each partition contains at least two nodes (or subsequences). Finally, we verify whether the sum

of the extents of each partition is smaller than the extent of the candidate motif  $S_k$ .

**Time complexity.** The time complexity for generating the MST using Prim's algorithm is  $\mathcal{O}(k \log k)$ . The approximate solution for verifying self-sufficiency has a time complexity of  $\mathcal{O}(k^3)$ . In contrast, an exact solution requires  $\mathcal{O}(2^k)$  time.

### A.4 MotiSet algorithm for discovering the best set of motifs

We present the pseudo-code of the proposed MOTISET algorithm, which uses A<sup>\*</sup> search to enumerate all sets of motifs and select the set with the lowest value.

Algorithm. Algorithm 4 presents the MOTISET algorithm. We begin by computing the distance matrix and identifying local motiflets using the algorithm described in Section 4.3. Specifically, we call K\_MOTIFLETS\_HEAP once with the maximum motif size  $k_{max}$  and obtain an index containing local motiflets of varying sizes. We iterate over motif sizes from 2 to  $k_{max}$  and remove vertically redundant motifs, i.e., motifs of the same size that are redundant (lines 7–9). The final set of candidate motifs is used by ASTARMOTIF. ASTARMOTIF intializes an empty queue with each candidate motif as a singleton set. We prune candidate motifs before adding them to this singleton set by removing all horizontally redundant candidates, i.e., motifs of different sizes that are redundant. Additionally, we enforce an ordering among motifs, denoted by  $\prec$ , to prevent generating motif sequences instead of sets (line 15). We determine a global ordering by sorting all motifs based on the index of their first subsequence. Next, we start the A<sup>\*</sup> search (lines 18–27). In the main loop, we take the candidate set of motifs  $\mathbf{S}$  with the lowest value  $f(\mathbf{S})$  (see Def. 12) and the set of possible motifs for expansion,  $\mathcal{C}_{\mathbf{S}}$ . Next, we check boundary conditions and terminate the current branch if there are no more candidate motifs to add. We also stop if the current set contains mmotifs if the optional parameter m is defined (line 20-22). If no stopping condition apply, we continue the search. We expand the current set of motifs by adding the first motif from the remaining candidate motifs and adding the new set to the priority queue. As before, we also prune possible future candidate motifs on horizontal redundancy and using the global ordering (line 25). Finally, we return the set of motifs with the overall lowest value (28).

**Time complexity.** The computational complexity of constructing the distance matrix and identifying motiflets is the same as in MOTIPLUS. The complexity of A\* search depends on the cardinality of C and the effectiveness of vertical and horizontal pruning. In the worst case, there are  $O(k \cdot n)$  candidate motifs in C. Checking vertically and horizontal non-redundant motifs takes linear time for each motif. Evaluating the value function takes  $O(m \cdot (k^2 + n))$ -time but can be trivially cached for each motif. The theoretical worst-case time complexity of A\* is  $O((k \cdot n)^m)$ . We note that in practice, we suggest to stop after n iterations, e.g.  $n = 10\,000$ .

Algorithm 4: MOTISET: Discovering the best set of *m* motifs

```
Input: A time series T, motif length l, number of motifs m (optional), the
                       maximum motif size k_{max}
      Result: Set \mathbf{S} with up to m motifs
  1 procedure MOTISET(T, l, m, k_{max})
               D \leftarrow \text{CALC_DISTANCE_MATRIX}(T, l)
  2
               index \leftarrow K\_MOTIFLETS\_HEAP(T, D, l, k_{max})
  3
  \mathbf{4}
              \mathcal{C} \leftarrow \{\}
               for k \leftarrow 2 to k_{max} do
  \mathbf{5}
  6
                     \mathcal{C}_k \leftarrow \{\}
                      for S_k \in index[k] sorted on extent do
  7
                             if not-vert-redundant(S_k, C_k) then
  8
                                     \mathcal{C}_k \leftarrow \mathcal{C}_k \cup \{S_k\}
   9
10
                      \mathcal{C} \leftarrow \mathcal{C} \cup \mathcal{C}_k
              return ASTARMOTIFS(T, \mathcal{C}, m)
11
      procedure ASTARMOTIFS(T, C, m)
12
              Q \gets create \ empty \ priority \ queue
13
               for S \in \mathcal{C} do
14
                      \mathcal{C}_{\mathbf{S}} \leftarrow \{ S' \in \mathcal{C} \mid S \prec S' \land \textit{not-hor-redundant}(S', S) \}
15
                      \operatorname{Push}(Q,\langle\{S\},f(S),\mathcal{C}_{\mathbf{S}}\rangle)
16
              \mathcal{R} \leftarrow \{\}
17
              while Q \neq \emptyset do
18
                       \langle \mathbf{S}, f(\mathbf{S}), \mathcal{C}_{\mathbf{S}} \rangle \leftarrow \operatorname{POP}(Q)
19
                      if C_{\mathbf{S}} = \emptyset or |\mathbf{S}| = m then
20
\mathbf{21}
                              \mathcal{R} \leftarrow \mathcal{R} \cup \{\mathbf{S}\}
22
                             continue
                      \hat{S} \leftarrow \text{FIRST}(\mathcal{C}_{\mathbf{S}})
23
                      \hat{\mathbf{S}} \leftarrow \mathbf{S} \cup \{\hat{S}\}
\mathbf{24}
                      \hat{\mathcal{C}}_{\mathbf{S}} \leftarrow \{ S' \in \mathcal{C}_{\mathbf{S}} \mid \hat{S} \prec S' \land \textit{not-hor-redundant}(S', \hat{S}) \}
\mathbf{25}
                      PUSH(Q, \langle \mathbf{S}, f(\mathbf{S}), \mathcal{C}_{\mathbf{S}} \setminus \{\hat{S}\} \rangle)
26
                      \operatorname{PUSH}(Q, \langle \hat{\mathbf{S}}, f(\hat{\mathbf{S}}), \hat{\mathcal{C}}_{\mathbf{S}} \rangle)
\mathbf{27}
              return argmax f(\mathbf{S})
\mathbf{S} \in \mathcal{R}
\mathbf{28}
```

# A.5 The runtime and the quality for discovering motiflets using the exact and approximate algorithm

In this section, we report detailed results comparing the approximate and exact algorithms for discovering motiflets. The results concerning quality are reported in Fig. 5. We note that on these smaller datasets (n = 1000), the difference in runtime is negligible, with both methods completing in under one second. We conclude that the proposed approximate algorithm closely approximates the exact solution for discovering k-motiflets, which becomes intractable for larger values of k and n.

In a second experiment, we evaluate the runtime of the exact algorithm for discovering motiflets. We ran this experiment on a laptop with a 8-Core Intel CPU and 16 GB of RAM. The runtime in seconds for varying k of the first three datasets is shown in Fig. 6. The original exact algorithm enumerates all possible  $n^k$  subsets if pruning fails using Theorem 2 [12]. As a result, the original algorithm takes several days to complete and runs out of memory for k = 15. We find that the proposed exact k-motiflet algorithm completes within a reasonable time for k up to 15. We note that in real-world time series, the maximal cardinality of



Fig. 5: Comparing the quality of the approximate k-motified discovery algorithms on 12 benchmark TS [7]. We report the ratio of the extent of the top-1 motif found by the approximate algorithm to that of the exact algorithm. We find that the proposed algorithm discovers motifiets that closely match those found by the exact algorithm in all TS instances.



Fig. 6: Comparison of the runtime for the discovery of k-motifiets for increasing values of k using the proposed exact method and the brute-force baseline on the first three datasets from [7]. The baseline method takes more than 30 minutes for k = 6 on TS 2 and 4, whereas the proposed exact method completes for all values of k.

motifs is often limited. For example, in the *tsdm-benchmark*, a maximum value of k = 10 is sufficient.

#### A.6 Selecting hyper-parameters on the TSMD-benchmark

In Table 3, we summarize the parameter settings for each motif discovery method, optimized via grid search. We use  $l_{gt}$ ,  $k_{gt}$ , and  $m_{gt}$  to denote the fixed motif length, maximum motif size, and maximum motif count from the ground truth, respectively. Default parameters were chosen based on preliminary experiments to maximize accuracy. For each dataset, we use the same free parameter settings across all time series instances, identified through grid search on validation data following [15].

LOCOMOTIF and MMOTIF depend on sensitive distance thresholds, while GRAMMARVIZ requires selecting an optimal alphabet and word size for discretization—both of which require careful tuning per dataset. In contrast, MOTIPLUS and MOTISET need only three parameters: motif length, (optional) number of motifs, and maximum motif size. The window length is selected automatically by maximizing the area under the curve, and the maximum size mainly impacts runtime. A default value of  $k_{max} = 10$  performs well across datasets. Finally, we note that exhaustive tuning is generally infeasible in unsupervised motif discovery.

Dataset	Free parameters	Default parameters	Fixed parameters
GRAMMARVIZ	$a \in \{2, 4, 8, 16\},$		$l = l_{gt}$
	$w \in 2, \ldots, \frac{l}{2}$		
MMotif	$r \in \{1, 2, 3, 4, 5\}$		$m = m_{gt},  l = l_{gt}$
LocoMotif	$\rho \in \{0.1, 0.2, \dots, 0.9\}$	warping = True	$m = m_{gt},  l = l_{gt}$
Motiflets	$k_{max} \in \left[k_{gt} - 3, k_{gt} + 3\right]$		$m = m_{gt},  l = l_{gt}$
MotiPlus	$k_{max} \in \left[k_{gt} - 3, k_{gt} + 3\right]$	$self\_sufficient = True,$	$m = m_{gt},  l = l_{gt}$
		$k\_closed = True$	
MotiSet	$k_{max} \in \bigl[k_{gt}-3,k_{gt}+3\bigr],$	$n = 10000, \gamma = 0.2$	$m = m_{gt},  l = l_{gt}$

Table 3: Parameter ranges for grid search in experiments.