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

Len Feremans[1], Patrick Schäfer[2], and Wannes Meert[3]

[1] University of Antwerp, Department of Computer science, Antwerpen, Belgium
[2] Humboldt-Universität zu Berlin, Department of Computer Science, Berlin, Germany
[3] KU Leuven, Department of Computer Science, Leuven, Belgium
Email: 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 state-of-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

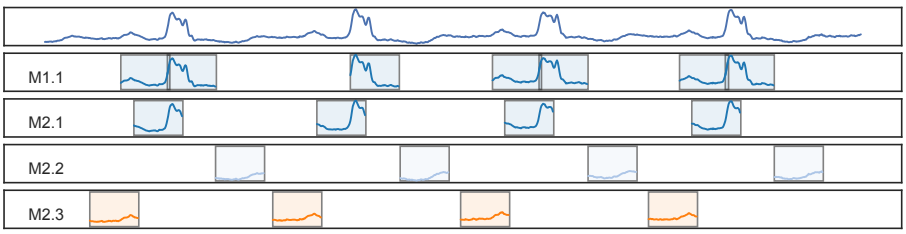

Fig. 1: An example of motifs discovered by Motiflets and MotiPlus. Motiflets identifies a single motiflet of size 7 (M1.1), which lacks self-sufficiency and should be split. In contrast, MotiPlus finds three similar local motiflets of size 4 (M2.1–3), which Motiflets cannot detect due to its one-motif-per-size 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 l in 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 motiflets*, 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 similarity or 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 $\lceil i - l \cdot \alpha \rceil \leqslant j \leqslant \lceil i + l \cdot \alpha \rceil$ 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 $P^l = \{T_{i,l} \mid 1 \leqslant i \leqslant n - l + 1\}$. The *normalized Euclidean distance* between two subsequences $T_{i,l}$ and $T_{j,l}$, with means μ and standard deviations σ, is:

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

Where we add ϵ 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 $k \geqslant 2$. 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., $extent(S_k) = max(\{z\text{-}ED(T_{i,l}, T_{j,l}) \mid (T_{i,l}, T_{j,l}) \in S_k \times S_k\})$.*

The extent measures the cluster diameter in normalized space and is at most twice the radius to a subsequence, where the radius is the large distance between a selected subsequence (the core) and other subsequences.

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-motiflet. The extent function of T is defined as $EF(k) = extent(S_k)$ where $k \geqslant 2$, i.e. $EF^l = extent(S^l_2), extent(S^l_3), \ldots, extent(S^l_{k_{max}})$.*

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, ϵ is a small constant used to prevent division by zero, and β 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* $\mathbf{S}$*, S_k is the local k-motiflet if it has the lowest extent of all non-redundant motifs:*

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

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 $S_k = \{T_{i_1,l}, \ldots, T_{i_k,l}\}$ 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 $\iff \nexists S'_{k+1} : S_k \subset S'_{k+1} \wedge extent(S'_{k+1}) < extent(S_{k+1})$. 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.

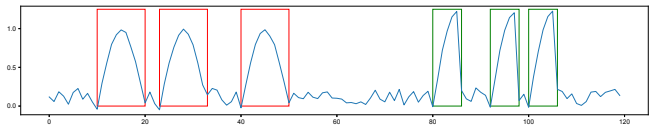

Fig. 2: A synthetic TS with two motifs, each of size 3, in red and green. Both motifs are local motiflets, i.e. the green motif has the lowest extent, and the red non-redundant motif the second lowest extent. Both motifs are k-closed, i.e. the extent increases significantly upon any addition of a non-trivially 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 the Appendix. 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 BestMotiflet. 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 $\mathbf{S}$ with up to m motifs

```
1  procedure MotiPlus(T, l, m, k_max)
2      D ← calc_distance_matrix(T, l)
3      index ← k_motiflets_heap(T, D, l, k_max)
4      S ← {}
5      for i ← 1 to m do
6          S_i ← BestMotiflet(T, index, S, k_max)
7          if S_i = ∅ then
8              break
9          S ← S ∪ {S_i}
10     return S
11 procedure BestMotiflet(T, index, S, k_max)
12     C ← {}
13     for k ← 2 to k_max do
14         for S_k ∈ index[k]  sorted on extent do
15             if non-redundant(S_k, S) ∧ k-closed(S_k) ∧ self-sufficient(S_k) then
16                 C ← C ∪ {S_k}
17                 break
18             else
19                 index[k] ← index[k]\{S_k}
20     if C = {} then
21         return ∅
22     S ← argmax_{S_k ∈ C} elbow(k)
23     return S
```

$\hat{n} \times k_{max}$ varying-sized local motiflets. 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 motiflets and the current set of motifs, $\mathbf{S}$. 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 motiflets 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 motiflet (line 22). As a special case, we return $\varnothing$ 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 motiflet 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-motiflets algorithm For local motiflets 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 motiflets once with size k_{max}. For brevity, we discuss the pseudo-code in the Appendix. 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 $\hat{S} : \hat{S}' \subset \hat{S}$: $extent(\hat{S}') \leqslant extent(\hat{S})$.*

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

We leverage *depth-first search* to create candidate motifs of growing size bottom-up 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 the Appendix.

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

Definition 12. *For a set of motifs, $\mathbf{S} = \{S_1, S_2, \ldots, S_m\}$, we seek high intra-motif 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):*

$$
\begin{aligned}
f(\mathbf{S}) &= norm(smpd(\mathbf{S})) - \gamma \cdot norm(smod(\mathbf{S})) \\
smpd(\mathbf{S}) &= \frac{\sum_{k=1}^{m} \sum_{i=1}^{|S_k|} min\_dist\_in(T_{i,l}, S_k)^2}{\sum_{k=1}^{m} |S_k|} \\
smod(\mathbf{S}) &= \sum_{k=1}^{m} min\_dist\_out(S_k, T)^2 \\
min\_dist\_in(T_{i,l}, S_k) &= min(\{z\text{-}ED(T_{i,l}, T_{j,l}) \mid T_{j,l} \in S_k \wedge T_{i,l} \neq T_{j,l}\}) \\
min\_dist\_out(S_k, T) &= min(\{z\text{-}ED(T_{i,l}, T_{j,l}) \mid T_{i,l} \in S_k \wedge T_{j,l} \in T\})
\end{aligned}
$$

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* 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 motiflets 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 the Appendix.

6 Experiments

In this section, we answer the following research questions[4]: **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-motiflet 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

[4] 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 the Appendix.

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 over each song. 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[5]. Regarding run-time, we find that

[5] 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	**7.6**	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.

MMotif and Motiflets are the 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[6]. 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 [7]. We tune parameters using grid search, thereby using the last 50 TS instances for validation and report the accuracy on

[6] We do not compare with motif discovery methods that performed worse, namely Emma, MrMotif, SetFinder, LatentMotifs, Valmod, and Vonsem

[7] 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.

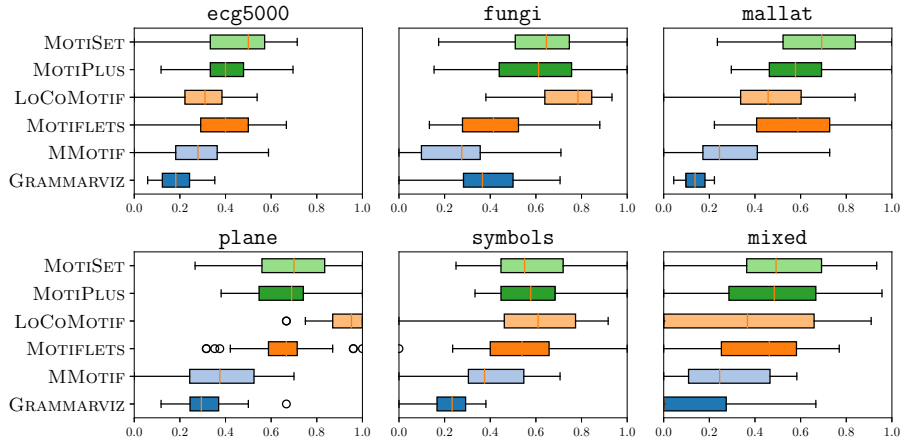

Fig. 3: The distribution of F1-scores for each method on the tsmd-benchmark datasets shows that either LoCoMotif or MotiSet performs best on each dataset. MotiPlus improves upon Motiflets. Grammarviz and MMotif perform significantly worse.

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	0.234 (+-0.097)	0.264 (+-0.196)	0.321 (+-0.088)
MMotif	0.314 (+-0.058)	0.400 (+-0.106)	0.323 (+-0.061)
Motiflets	0.498 (+-0.104)	0.719 (+-0.125)	0.403 (+-0.094)
LoCoMotif	0.565 (+-0.232)	0.604 (+-0.268)	0.564 (+-0.195)
MotiPlus	0.552 (+-0.094)	0.543 (+-0.125)	**0.621** (+-0.087)
MotiSet	**0.586** (+-0.097)	**0.773** (+-0.068)	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 similarity and 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 motiflets 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 motiflets by filtering for k-closed and self-sufficient motifs, while MOTISET uses A* 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-of-the-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 This research received funding from the Flemish Government (AI Research Program), and L.F. is funded on project 12B0V24N by Research Fund Flanders.

References

1. 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)
3. Feremans, L., Cule, B., Goethals, B.: Petsc: pattern-based embedding for time series classification. Data Mining and Knowledge Discovery **36**(3), 1015–1061 (2022)
4. 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)
6. Law, S.M.: Stumpy: A powerful and scalable python library for time series data mining. Journal of Open Source Software **4**(39), 1504 (2019)
7. Lin, J., Khade, R., Li, Y.: Rotation-invariant similarity in time series using bag-of-patterns representation. Journal of Intelligent Information Systems **39**, 287–315 (2012)
8. Lonardi, J., Patel, P.: Finding motifs in time series. In: 2nd Workshop on Temporal Data Mining. pp. 53–68 (2002)
9. 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)
10. 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)
11. Rotman, S., Čule, B., Feremans, L.: Efficiently mining frequent representative motifs in large collections of time series. In: BigDataprocee. pp. 66–75. IEEE (2023)
12. Schäfer, P., Leser, U.: Motiflets: Simple and accurate detection of motifs in time series. PVLDB **16**(4), 725–737 (2022)
13. 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)
14. 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)
15. 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)
17. 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)