Mining Similarity-Aware Distinguishing Sequential Patterns from Biomedical Sequences

Tinghai Pang†, Lei Duan∗, Jesse Li-Ling‡ and Guozhu Dong‡
∗School of Computer Science, Sichuan University, Chengdu, China
Email: pantinghai@163.com, leiduan@scu.edu.cn
†Institute of Genetic Medicine, School of Life Science, Sichuan University, Chengdu, China
Email: jiling@scu.edu.cn
‡Department of Computer Science & Engineering, Wright State University, USA
Email: guozhu.dong@wright.edu

Abstract—Mining distinguishing sequential patterns, which indicate unique properties of a target family of biomedical sequences, is useful for the explanation and characterization of phenomena concerning, as well as for the identification of biomarkers for, the target family. However, previous studies on mining distinguishing sequential patterns did not consider the important and widely-occurring case where biochemical similarity exists among the elements in a given type of biomedical sequences. To fill that gap, this paper considers mining distinguishing sequential patterns for data where sequence elements can be similar to each other; the associated patterns will be called similarity-aware distinguishing sequential patterns (simDSP). After presenting the challenges on mining simDSP, we present simDSP-Miner, a mining method with effective pruning techniques, for mining simDSPs with domain-specific similarity knowledge. Our empirical study using real-world protein sequences demonstrates that simDSP-Miner is effective and efficient, and it can discover more novel distinguishing sequential patterns than previous algorithms for mining distinguishing sequential patterns.

Index Terms—domain-specific similarity; distinguishing sequential pattern; protein sequence;

I. INTRODUCTION

DNA, protein, and medical sequence data is ubiquitous in the biomedical domain, and discovering sequential patterns from biomedical sequence data is useful for analyzing gene/protein families and diseases. Among various kinds of sequential patterns, distinguishing sequential patterns are useful for characterizing a given group of sequences and distinguishing that group from other groups of sequences. For example, given two sets of patients having different outcomes for a given gene therapy treatment, analyzing the distinguishing sequential patterns between the DNA sequences of the two sets can help doctors better understand when and why the treatment is effective or not on a given patient.

Several methods, such as [1], [2], [3], have been proposed to discover distinguishing sequential patterns. However, these methods ignore the domain-specific similarity among the sequence elements. Similarity among elements in sequences are present in many applications. For example, several groups of amino acids share similar biochemical properties – see Table I, which implies that similar amino acids often behave in similar manners. In medical records for patients, different medical symptoms can be very similar to each other; e.g. for the Congenital Heart Disease, “pulmonary atresia” is synonymous with “pulmonary artery atresia”, and “tetralogy of Fallot” is synonymous with “pulmonary artery atresia, overriding aorta, ventricular septal defect, and hypertrophy of right ventricle”.

Ignoring these similarity relationships can imply that the significance and support of individually discovered patterns are weak, and it can also lead to explosive growth in the number of patterns.

In this work, we introduce a new kind of sequential pattern called the similarity-aware distinguishing sequential pattern (or simDSP). A simDSP considers the domain-specific similarity among sequence elements. The key feature that makes simDSPs different from previous sequential patterns is that each element of a simDSP is composed by a set of items which are similar with each other according to the domain knowledge. Moreover, from mined simDSPs, we can also identify the similarity relationships with regards to the comparison between different groups.

There are technical challenges to discover simDSPs from biomedical sequences. These technical challenges include:

- Like other kinds of distinguishing sequential patterns [1], [2], [3], simDSP doesn’t have the Apriori property since we allow gap constraints (which make patterns more general and flexible).
- There are a large number of trivial simDSPs which can be inferred by others. Effective pruning methods to eliminate trivial simDSPs are critical for the computational efficiency.
- The consideration of domain-specific similarity may lead to large number of candidate patterns. Because of that, some strategy to balance the efficiency and memory cost is necessary.

This paper makes the following main contributions: (1) Introducing a novel biological data mining problem of simDSPs mining. (2) Designing an efficient algorithm for discovering simDSPs. (3) Conducting extensive experiments on real-world protein sequences, to evaluate our simDSPs mining algorithm, and to demonstrate that the proposed algorithm can find
such that $\forall$ distinct biomedical expressions $e, e' \in \Sigma$, we write $e \simeq e'$ if $e$ is similar to $e'$. The total set of such similarity relationships in a given application is denoted as $\kappa$; $\kappa$ is given by users and is considered as domain knowledge.

A similarity group $g$ is a set of biomedical expressions such that $\forall e, e' \in g$ it is the case that $e \simeq e'$. We will represent a similarity group as an itemset, as in $\{e_1, e_2, e_3\}$. It is implicitly understood that elements in a similarity group are interchangeable. If $\forall e \in \Sigma \setminus g$, there is no $e' \in g$ satisfying $e \simeq e'$, we say that $g$ is a maximal similarity group over $\Sigma$. For clarity, we write $g^*$ if $g$ is maximal. A similarity group set $\mathcal{G}$ over $\Sigma$ is a set of maximal similarity groups such that (i) $\forall g_1, g_2 \in \mathcal{G}$, $g_1 \cap g_2 = \emptyset$; (ii) $\bigcup g^* = \Sigma$.

We require that the similarity relationship in a similarity group be reflexive, symmetric, and transitive. That is, for $e, e', e'' \in \Sigma$, we have

- reflexivity: $e \simeq e$;
- symmetry: if $e \simeq e'$, then $e' \simeq e$;
- transitivity: if $e \simeq e'$ and $e' \simeq e''$, then $e \simeq e''$.

### Example 1

Table I lists the 20 amino acids divided into $9$ similarity groups based on their properties. For example, Histidine, Lysine, and Arginine make one similarity group, since their physico-chemical properties are "polar positive".

A sequence $S$ over $\Sigma$ is an ordered list of biomedical observations with the form of $S = \langle e_1, e_2, \ldots, e_n \rangle$, where $e_i \in \Sigma (1 \leq i \leq n)$ is an element. The length of $S$, denoted by $||S||$, is the number of elements in $S$. We denote by $S[i]$ the $i$-th element in $S (1 \leq i \leq ||S||)$. The gap between $S[i]$ and $S[j]$ ($1 \leq i < j \leq ||S||$) is the number of elements between $S[i]$ and $S[j]$. The gap constraint $\gamma$ is defined as an interval which consists of two nonnegative integers $[\gamma.\min, \gamma.\max]$.

A similarity-group-based sequence $P$ over similarity group set $\mathcal{G}$ is an ordered list of similarity subgroups of the form $P = \langle g_1, g_2, \ldots, g_m \rangle$, where for each $g_i (1 \leq i \leq m)$, there exists $g^* \in \mathcal{G}$ satisfying $g_i \subseteq g^*$. We denote by $||P||$ the number of similarity subgroups in $P$, and $P[i]$ the $i$-th similarity subgroup in $P (1 \leq i \leq ||P||)$. For a similarity-group-based sequence $P'$, if there exist integers $1 \leq k_1 < k_2 < \cdots < k_{||P||} \leq ||P||$, such that $P[k_i] \subseteq P'[i]$ for all $1 \leq i \leq ||P'||$, then we say that $P'$ covers $P$. For example, $\langle\{D, E\}, \{N, Q\}\rangle > \langle\{D, E\}, \{N\}, \{A, V\}\rangle$.

We emphasize that the similarity subgroups in a similarity-group-based sequence are allowed to be non-maximal. We made this choice on purpose, since this will make it flexible to discover the kind of similarity subgroups supported by data under consideration under the constraint given by domain knowledge – the discovered similarity subgroups for a given maximal similarity group can be different in different similarity-group-based sequences and in different positions of one sequence.

For a sequence $S$ and a similarity-group-based sequence $P$ satisfying $||S|| > ||P||$, if there exist integers $1 \leq k_1 < k_2 < \cdots < k_{||P||} \leq ||S||$, such that

- $S[k_i] \subseteq P[i]$ for all $1 \leq i \leq ||P||$, and
- $\gamma.\min \leq k_{i+1} - k_i - 1 \leq \gamma.\max$ for all $1 \leq i < ||P||$,

then we say $P$ occurs in $S$ with gap constraint $\gamma$, denoted by $P \sqsubseteq_\gamma S$.

### Example 2

Consider the protein sequences in Table II using Table I as the similarity groups. Let $P = \langle\{D, E\}, \{N\}, \{A, V\}, \{N\}\rangle$, $\gamma = \{0, 1\}$. Then $P$ occurs in $S_1$, since $S_1[1] \subseteq P[1]$, $S_1[3] \subseteq P[2]$, $S_1[5] \subseteq P[3]$, $S_1[7] \subseteq P[4]$ and $3 - 3 - 1 \in [0, 1]$, $5 - 3 - 1 \in [0, 1]$, $7 - 5 - 1 \in [0, 1]$.

The support of a similarity-group-based sequence $P$ with gap constraint $\gamma$ in sequence set $D$, denoted by $\text{Sup}(D, P, \gamma)$, is defined by Equation 1.

$$\text{Sup}(D, P, \gamma) = \frac{|\{S \in D \mid P \sqsubseteq_\gamma S\}|}{|D|}$$  \(1\)

Since $\gamma$ (predetermined by users) is a fixed constant, we often write $\text{Sup}(D, P, \gamma)$ as $\text{Sup}(D, P)$ for brevity.
Definition 1 (Similarity-aware Distinguishing Sequential Patterns with Gap Constraint). Given two sets of sequences $D_+$ and $D_-$, two support thresholds $\alpha$ and $\beta$, and gap constraint $\gamma$, a similarity-group-based sequence $P$ is a Similarity-aware Distinguishing Sequential Pattern with Gap Constraint (simDSP), if the following conditions are true:

1) (support contrast) $Supt(D_+, P) \geq \alpha$ and $Supt(D_-, P) \leq \beta$;

2) (non-redundancy) There does not exist a sequence $P'$ such that $P'$ covers $P$ and $P'$ satisfies Condition 1.

Given $\alpha$, $\beta$, and $\gamma$, the similarity-aware distinguishing sequential pattern mining problem is to find all the simDSPs from the target set $D_+$ against the non-target set $D_-$. 

Remark: For a fixed dataset, patterns with larger similarity groups tend to have larger supports and patterns with smaller similarity groups tend to have smaller supports. For simDSP mining, there are two competing datasets, namely $D_+$ and $D_-$. In order for a pattern $P$ to become a simDSP, the “support contrast” condition becomes easier to satisfy if the similarity groups are large when considering $P$'s support in $D_+$ and if similarity groups are small when considering $P$'s support in $D_-$. To illustrate, consider the data set in Table II with $\gamma = [0, 1]$. For pattern $P = <F, A, V>$, we have $Supt(D_+, P) = 2$, and $Supt(D_-, P) = 2$. For pattern $P' = <F, V>$ (whose second element is a smaller similarity group than the second element in $P$), we have $Supt(D_+, P') = 1$ and $Supt(D_-, P') = 0$. $P'$ is a distinguishing sequential pattern but $P$ is not. The pull towards larger similarity groups by $D_+$ and the pull towards smaller ones by $D_-$ imply that we can find many similarity subgroups supported by the data that are different from the similarity groups given by the user (which are most likely observed from all data of the application instead of from the two competing families $D_+$ and $D_-$).

Example 3. Consider Table II using Table I as the similarity groups again. Suppose gap constraint $\gamma = [0, 1]$, two support thresholds $\alpha = 1.0$, $\beta = 0.4$. For $P = <\{D, E\}, \{N\}, \{A, V\}, \{N\}>$, $P$ occurs in $S_1$, $S_2$, $S_3$ and $S_4$. So $Supt(D_+, P) = 3/3 = 1 \geq \alpha$, $Supt(D_-, P) = 1/3 \leq \beta$. Moreover, it can be verified that $P$ satisfies the non-redundancy condition. Thus, $P$ is a simDSP.

Table III lists the frequently used notations in this paper.

III. RELATED WORK

With the rapid development of technologies for gene discovery and DNA sequencing, large volume of biomedical sequence data have been generated [4]. To automatically analyze biomedical sequences, many effective methods have been designed [5]. For example, Langmead et al. [6] presented a fast and memory-efficient alignment algorithm for short DNA sequences to the human genome; Krek et al. [7] designed a method to identify common targets of microRNAs; Ferreira et al. [8] studied mining frequent sequence patterns satisfying some pre-defined constraints from protein sequences; Wu et al. [9] studied mining frequent patterns without user-specified gap constraints from DNA sequences. Huang et al. [10] mined the clinical activity sequence from medical behaviors. Bailey et al. [11] discovered and analyzed sequence motifs from biomedical sequences. And there are some works used gap constraint to discover patterns from DNA sequences [12], [13], [14].

Similarity is a well studied subject in information retrieval and data mining. A comprehensive review of the abundant literature on similarity is clearly beyond the capacity of this paper. Several studies on the topic, such as [15], [16], [17], [18], provide some treatments on introducing similarity measure into pattern mining. In this work, instead of using predefined similarity measure, we consider the domain-specific similarity among the sequence elements, which is flexibly defined by domain experts. As a result, the patterns discovered could be more explainable and reasonable.

Distinguishing sequential patterns, indicating unique properties of a target set of biomedical sequences are useful for the explanation and characterization of phenomena concerning, for the identification of biomarkers for, the target set. They are also useful as features. She et al. [19] used discriminating subsequences to predict outer membrane proteins. Shah et al. [20] constructed a classifier by contrast patterns to predict peptide folding. Ghosh et al. demonstrated the significant clinical value of distinguishing sequential pattern by using distinguishing sequential patterns to predict hypotension risk [21] and septic shock for ICU patients [22].

Several methods for discovering distinguishing sequential patterns have been proposed. Dong et al. [23] discovered emerging patterns from contrasts datasets. Ji et al. [1] designed a method called ConsGapMiner for discovering minimal dis-
tonguing patterns with gap constraint. Wang et al. [2] defined density-aware distinguishing sequential pattern by introducing the concept of density into the mining problem. Yang et al. [3] considered mining distinguishing sequential patterns from the sequences in which each element is an itemset. To avoid setting unsuitable support thresholds, Yang et al. [24] designed an algorithm to find top-k distinguishing sequential patterns. Gao et al. [25] applied an evolutionary computing method to mine top-k distinguishing sequential patterns without predefined gap constraints. Li et al. [26] solved the problem of imbalanced data when mining distinguishing patterns. Besides, there also some works about the application of distinguishing patterns such as classification and prediction [27], [28], [29]. Clearly, our study is significantly different from the previous studies on distinguishing sequential pattern mining, since the previous mining methods do not consider the similarity among sequence elements.

IV. DESIGN OF simDSP-Miner

In this section, we present our simDSP-Miner algorithm for discovering simDSPs from two given families of biomedical sequences. In general, simDSP-Miner consists of three main steps: (i) the generation of candidate elements of simDSPs, (ii) the generation of candidate simDSPs. (iii) the evaluation of candidate simDSPs. In addition, we present a strategy to trade-off the computation time and memory cost in this section.

A. Candidate Elements Generation

Please recall that a simDSP is an ordered list of similarity subgroups satisfying the “support contrast” and “non-redundancy” conditions. For candidate simDSP generation, the first step is enumerating all elements that can be used to compose a candidate simDSP. We note that each element in a simDSP, i.e. a similarity subgroup, is not required to be a maximal similarity group. For a given similarity group set \( \mathcal{G} \), an intuitive way to generate candidate elements is generating all subsets of every maximal similarity group in \( \mathcal{G} \). Clearly this method is time-consuming, and many useless candidate elements not supported by data may be generated. Instead, simDSP-Miner prunes unpromising candidates using Pruning Rule 1.

**Lemma 1.** Given a set of sequences \( D \), for a similarity-group-based sequence \( P = \langle g_1, g_2, \ldots, g_m \rangle \), we have

\[
\text{Sup}(D, P) \leq \min_{1 \leq i \leq m} \text{Sup}(D, <g_i>).
\]

**Proof:** (Outline) We observe that \( <g_i> \) (\( 1 \leq i \leq m \)) covers \( P \). So, for each sequence \( S \in D \), \( <g_i> \) occurs in \( S \) if \( P \) occurs in \( S \). So \( \text{Sup}(D, P) \leq \text{Sup}(D, <g_i>). \)

**Lemma 2.** Given a set of sequences \( D \) and two similarity subgroups \( g \) and \( g' \) satisfying \( g' \subseteq g \), we have

\[
\text{Sup}(D, <g'>) \leq \text{Sup}(D, <g>).
\]

**Proof:** (Outline) As \( g' \subseteq g \), if \( <g> \) occurs in a sequence \( S \in D \), then \( <g'> \) occurs in \( S \). So \( \text{Sup}(D, <g'>) \leq \text{Sup}(D, <g>). \)

---

**Pruning Rule 1.** For a similarity subgroup \( g \), if \( \text{Sup}(D, <g>) < \alpha \), then all subsets of \( g \) can be pruned.

For a maximal similarity group \( g^* \), simDSP-Miner enumerates each subset of \( g^* \) by traversing the subset lattice of \( g^* \) (e.g. Figure 1) in a breadth-first manner.

Furthermore, simDSP-Miner employs Lemma 3 to estimate an upper bound of \( \text{Sup}(D, <g>) \) for a quick pruning.

**Lemma 3.** Given a set of sequences \( D \), for a similarity group \( g = \{e_1, e_2, \ldots, e_n\} \), we have

\[
\text{Sup}(D, <g>) \leq \sum_{1 \leq i \leq n} \text{Sup}(D, <\{e_i\}>)
\]

**Proof:** (Outline) The claim holds because \( LHS = |\{S \in D \mid <g> \text{ occurs in } S\}| = | \bigcup_{e_i \in g} \{S \in D \mid <\{e_i\}> \text{ occurs in } S\}| \leq \sum_{e_i \in g} |\{S \in D \mid <\{e_i\}> \text{ occurs in } S\}| = RHS. \)

Algorithm 1 gives the pseudo-code of generating candidate elements of simDSPs by simDSP-Miner.

---

**B. Candidate Pattern Generation**

Our simDSP-Miner uses a candidate-generation-and-test approach instead of the popular depth-first approach (which was used in some previous studies on distinguishing subsequence pattern mining [1], [3]). The main reason for us to make this choice is because a candidate-generation-and-test approach allows us to more effectively prune the search space and reduce the computation cost.

In accordance with the non-redundancy condition in the definition of simDSP (Definition 1), we have the following pruning rule.

**Pruning Rule 2.** For a candidate pattern \( P \), if there exists a discovered simDSP covering \( P \), then \( P \) can be pruned.

We denote by \( \mathcal{C}^\ell \) the set of candidate simDSPs with length-\( \ell \) (\( \ell \geq 1 \)). Initially, \( \mathcal{C}^1 \) is the set of all candidate elements got by Algorithm 1. Through the supports computing and redundancy checking, \( \mathcal{C}^\ell \) can be further partitioned into three disjoint subsets:

- \( \mathbb{P}^\ell \): the set of simDSPs with length-\( \ell \);
Algorithm 1 candidateElement($D_+, \mathcal{G}, \alpha$)

Input: $D_+$: target set of sequences, $\mathcal{G}$: similarity group set, $\alpha$: minimal support threshold for $D_+$.

Output: $CE$: candidate elements of simDSPs.

1: for each maximal similarity pattern $g^* \in \mathcal{G}$ do
2: \hspace{1em} $C \gets \{g^*\}$;
3: \hspace{1em} while $C \neq \emptyset$ do
4: \hspace{2em} $NG \gets \emptyset$;
5: \hspace{2em} for each itemset $g \in C$ do
6: \hspace{3em} if $Sup(D_+, \langle g \rangle) \geq \alpha$ then
7: \hspace{4em} $CE \gets CE \cup \{g\}$;
8: \hspace{3em} end if
9: \hspace{2em} end for
10: \hspace{1em} remove all subsets of $g$ from $NG$; // Pruning Rule 1
11: \hspace{1em} end if
12: \hspace{1em} end for
13: \hspace{1em} $C \gets NG$;
14: \hspace{1em} end while
15: \hspace{1em} return $CE$;

---

- $\mathbb{U}^\ell$: a set of patterns $P$ where simDSP-Miner can determine that $P$ is not a simDSP.
- $\mathbb{R}^\ell$: the set of patterns $P$ such that $Sup(D_+, P) \geq \alpha$ and $Sup(D_-, P) > \beta$.

simDSP-Miner uses $\mathbb{R}^\ell$ to generate $\mathbb{C}^{\ell+1}$. Since any pattern containing a pattern in $\mathbb{R}^\ell \cup \mathbb{U}^\ell$ cannot be a simDSP, this will not lead to the loss of any simDSP. simDSP-Miner will generate candidate patterns with length-(\ell+1) by “appending” a similarity subgroup in $\mathbb{C}^1$ to each pattern in $\mathbb{R}^\ell$.

**Definition 2.** Let $P$ be a pattern in $\mathbb{R}^\ell$, and $g$ be a similarity subgroup in $\mathbb{C}^1$. A length-(\ell+1) pattern in $\mathbb{C}^{\ell+1}$ is composed by appending $g$ to the end of $P$, denoted by $P \oplus g$:

$$P \oplus g = \langle P_1, P_2, \ldots, P_{\ell}, g \rangle$$

**Suppose** $P = \langle\{D, E\}, \{N\}\rangle \in \mathbb{R}^2$, $g = \{A, V\} \in \mathbb{R}^1$. Then, $P \oplus g = \langle\{D, E\}, \{N\}, \{A, V\}\rangle \in \mathbb{C}^3$.

**Lemma 4.** Given a set of sequences $D$, a pattern $P$ and a similarity subgroup $g$, we have

$$Sup(D, P) \geq Sup(D, P \oplus g)$$

**Proof:** It is easy to see that, for each sequence $S$ in $D$, if $P \oplus g$ occurs in $S$ then so does $P$.

**Pruning Rule 3.** For candidate pattern $P$, if $Sup(D_+, P) < \alpha$, then all patterns covered by $P$ can be pruned.

By using Equation 2 for every pair of a pattern in $\mathbb{R}^\ell$ and a similarity group in $\mathbb{C}^1$, all patterns in $\mathbb{C}^{\ell+1}$ can be generated. However, the efficiency will be low if $|\mathbb{R}^1|$ is large. For the sake of efficiency, we propose a method to filter out “unpromising” candidate elements from $\mathbb{R}^1$ for pattern $P \in \mathbb{R}^{[P]}$. To specify the method, we firstly present two definitions.

**Definition 3** (Max-Suffix). Given a pattern $P = \langle P_1, P_2, \ldots, P_n \rangle$ with $n \geq 2$, the max-suffix of $P$, denoted by $suf(P)$, is $\langle P_2, \ldots, P_{n-1}, P_n \rangle$. For the case $|P| = 1$, $suf(P) = \lambda$ (the empty sequence).

**Example 4.** For a pattern $P = \langle\{D, E\}, \{N\}, \{A, V\}\rangle$, $suf(P) = \langle\{N\}, \{A, V\}\rangle$.

**Property 1.** Given a set of sequences $D$, for a pattern $P$ and a similarity group $g$, we have

$$\{S \in D \mid P \oplus g \sqsubseteq S\} \subseteq \{S \in D \mid suf(P) \oplus g \sqsubseteq S\}$$

**Definition 4** (Follow-Set). Given two sets of sequences $D_+$ and $D_-$, two support thresholds $\alpha$ and $\beta$, the follow-set of pattern $P$, denoted by $f(P)$, is $\{g \in \mathbb{R}^1 \mid Sup(D_+, P \oplus g) \geq \alpha \land Sup(D_-, P \oplus g) > \beta\}$.
Property 2. \( fs(P) \subseteq fs(suf(P)) \).

Consider sequences in Table II. Figure 2 illustrates an example of generating candidate simDSPs using follow-sets. The values of \( \alpha, \beta, \) and \( \gamma \) are the same as those in Example 3. Take pattern \( P = \langle \{N\} \rangle \) for an instance, simDSP-Miner generates new candidate patterns by appending similarity groups in \( \mathbb{R}^1 \) to the end of \( P \). Then simDSP-Miner checks the support of each new candidate pattern. If \( Sup(D_+, \langle \{N\}, \{A, V\} \rangle) \geq \alpha, Sup(D_-, \langle \{N\}, \{A, V\} \rangle) > \beta, \) and \( Sup(D_+, \langle \{N\}, \{F\} \rangle) \geq \alpha, Sup(D_-, \langle \{N\}, \{F\} \rangle) > \beta, \) we add \( \{A, V\} \) and \( \{F\} \) to \( fs(\langle \{N\} \rangle) \). For a pattern \( P = \langle \{D, E\}, \{N\} \rangle \), we have \( suf(P) = \langle \{N\} \rangle \), and \( fs(suf(P)) = \{\{A, V\}, \{F\}\} \). Therefore, the new candidate patterns are \( \langle \{D, E\}, \{N\}, \{A, V\} \rangle \) and \( \langle \{D, E\}, \{N\}, \{A, V\} \rangle \) can be generated by appending \( fs(\langle \{N\}, \{A, V\} \rangle) \) to pattern \( \langle \{D, E\}, \{N\}, \{A, V\} \rangle \).

C. Framework of simDSP-Miner

For the sake of computational efficiency, we adopt the following strategies.

In the implementation of simDSP-Miner, we choose the vertical representation of a set of sequences and a bitset implementation of the positions of each sequence element because of their well-known advantages, discussed in previous works, such as [30].

In simDSP-Miner similarity subgroups in \( \mathbb{R}^1 \) are sorted by the size descending order, so that patterns consisting of larger similarity subgroups can be generated early. Then, we can make use of Pruning Rule 2 to remove redundant candidate patterns more efficiently.

As stated above, the advantage of candidate-generation-and-test approach is high efficiency. However, the memory cost is huge, since many temporary results have to be stored. Thus, the candidate-generation-and-test approach is unsuitable for very large data sets. To overcome this challenge, we propose a hybrid search strategy (a combination of breadth-first search and depth-first search). Specifically, simDSP-Miner firstly performs the breadth-first strategy to generate candidate patterns using pre-defined amount of memory, then performs the depth-first strategy to continue the remaining search. Please note that by a slightly modification (using the suffix of \( P \) instead of \( P \) in Definition 4) the idea of follow-set can still be used for generating new candidate patterns in the case of performing depth-first strategy.

Algorithm 2 summarizes the pseudo-code of the framework of simDSP-Miner. We can see that the framework of simDSP-Miner is flexible to be implemented in either breadth-first search strategy or depth-first search strategy.

V. Empirical Evaluation

In this section, we report a systematic empirical study on real-world protein sequences to verify the effectiveness and efficiency of our simDSP-Miner. We also list some interesting patterns discovered by simDSP-Miner, as well as state the potential application value of simDSP-Miner from a biomedical perspective. The protein sequences used in our experiments are fetched from Pfam1.

All experiments are conducted on a computer with an Intel Core i7-4790 3.60 GHz CPU, and 16 GB main memory, running Windows 7 operating system. All algorithms are implemented in Java and compiled by JDK 8.

A. Effectiveness

We apply simDSP-Miner to two protein families of clan RRM and two protein families of clan CL0010 (the characteristics of the sequence set are listed in Table IV). The domain

\[ \text{http://pfam.xfam.org} \]
knowledge (K) is the properties of amino acids (Table I), by which amino acids can be grouped by their properties.

We use experiments to confirm our intuition that by considering similarity groups more distinguishing sequential patterns with larger support contrast can be discovered. To this end, we apply ConSGapMiner proposed by [1] to mine distinguishing sequential patterns (DSPs) with the same running parameters. Please note that the mining results of ConSGapMiner are the same as simDSP-Miner when the size of each similarity group is 1.

1) Importance of considering “similarity”: Table V presents some of the simDSPs discovered by simDSP-Miner with Nup35_RRM_2 as D+ and RNA_bind as D−. There are some interesting patterns, such as \(<\{D, E\}, \{D, E\}, \{D, E\}\)>, which consists of three amino acid subgroups with polar negative property. This pattern occurs in Nup35_RRM_2 frequently but infrequently in RNA_bind. The patterns \(<\{D, E\}, \{P, G\}, \{T, N, Q\}, \{V, L, I, M\}\)>, \(<\{A, V, L, I\}, \{H, R\}, \{F, Y, W\}, \{S, T, Q\}\) and \(<\{S, T\}, \{A, V, L, I, M\}, \{H, K, R\}, \{D, E\}\>) indicate that there are some fixed combinations of amino acids with different properties that occur in the protein family Nup35_RRM_2 but not in RNA_bind; these combinations can be important in making Nup35_RRM_2 and RNA_bind different from each other. They may also help us understand the structure and the conserved sequence of these protein families. For example, the pattern \(<\{A, V, L, I\}, \{H, R\}, \{F, Y, W\}, \{S, T, Q\}\) shows that the amino acids with the order of the following four properties, namely “non-polar alphatic, polar positive, non-polar aromatic and polar neutral”, is unique in Nup35_RRM_2.

As shown in Table VI, the number of simDSPs is much larger than that of DSPs when \(\alpha = 0.8\), \(\beta = 0.4\). We increase the contrast threshold by increasing \(\alpha\) and decreasing \(\beta\). Then we find that there are fewer patterns of DSPs in Table VII. Please note that there is no pattern discovered by ConSGapMiner while several simDSPs are discovered when \(\alpha = 0.95\). From Table V, we can observe that all DSPs are covered by simDSPs. For example, \(<\{E\}, \{E\}\) is covered by \(<\{H, R\}, \{E\}\>, \(<\{E\}, \{A\}\) is covered by \(<\{A, M\}, \{A\}\)>, and \(<\{A\}, \{A\}\) is covered by \(<\{A, M\}, \{A\}\>.

2) Biomedical application value: A number of simDSPs have been identified as potential classifiers for the two groups of sequence data. Such sequences may represent important amino acid combinations which can determine the critical structure of the protein molecules or constitute important functional motifs or domains of the protein molecules, although the validity of such characteristic sequences are to be verified by biological experiments. From a molecular evolution perspective, simDSPs are particularly useful for identifying members of gene or protein families over a broad range of species. While mutations may occur randomly, only those mutations compatible with life (replacement of nucleotides or amino acids whose features are similar to the wild-type one, thereby will confer minimal impact on the individual) may be selected and transmitted.

B. Efficiency

To the best of our knowledge, there are no previous works tackling exactly the same problem like our study. Therefore, we only evaluate the efficiency of simDSP-Miner using different search strategies. Specifically, we implement simDSP-Miner using breadth-first approach (named simDSP-Miner-b), simDSP-Miner using the depth-first approach (named simDSP-Miner-d), and simDSP-Miner using a combination of breadth-first search and depth-first search (named simDSP-Miner-b&d), respectively.
Figure 3 shows the efficiency of our algorithms with varying \( \gamma \). The experiment uses sequences in Table IV, and support threshold \( \alpha = 0.98 \), \( \beta = 0.40 \). For simDSP-Miner-b\&d, we set the memory threshold as 7 GB, i.e., simDSP-Miner-b\&d performs with breadth-first search strategy with 7 GB memory before performing depth-first search strategy. As shown in Figure 3(a), with the assistance of follow-set, simDSP-Miner-b and simDSP-Miner-b\&d have a better performance than simDSP-Miner-d. The runtime of all algorithms is increasing with the increase of \( \gamma \). Even though there are usually more candidate patterns as shown in Figure 3(c), thanks to the use of follow-set to generate new candidate patterns, it is clear to see that simDSP-Miner-b checks fewer candidate patterns than simDSP-Miner-d. As shown in Figure 3(b), simDSP-Miner-b\&d uses smaller size of memory than simDSP-Miner-b but is more efficient than the version using depth-first search approach. We can see that simDSP-Miner-b\&d makes a nice trade-off between the computational efficiency and memory.
cost.

Figure 4 and Figure 5, respectively, present the efficiency of our algorithms with regards to support thresholds $\alpha$ and $\beta$. For simDSP-Miner-b&d, we set the memory threshold as 5 GB, i.e., simDSP-Miner-b&d performs with breadth-first search strategy with 5 GB memory before performing depth-first search strategy. We can see that the running time of each algorithm decreases with the increase of $\alpha$. The reason lies that more candidate patterns are pruned by Pruning Rule 3 with larger $\alpha$. From Figure 4(b) and Figure 5(b), we can see that the memory cost of simDSP-Miner-d and simDSP-Miner-b&d are approximately the same, but simDSP-Miner-b&d runs faster than simDSP-Miner-d, since some redundant patterns cannot be discovered in time to help prune wasted computation in depth-first search.

VI. DISCUSSION

The similarity group in this paper is given by domain knowledge. For example, the similarity group in protein sequences analysis is obtained by the properties of amino acids. For other sequences, there are many ways to measure the similarity of elements. For example, when the element of textual sequence is a word, and the similarity among words can be measured by some general similarity measures such as edit distance and semantic distance. In shopping sequences, the similarity of goods can be obtained by their function or their categories.

Similarity-aware distinguishing sequential pattern mining, which considers the similarity of elements, is a general method. As presented above, the consideration of similarity between elements can help us identify interesting new patterns. This idea can also be applied in other distinguishing sequential patterns such as density-aware distinguishing sequential pattern [2] and itemset-based distinguishing sequential pattern [3].

VII. CONCLUSION

In this paper, we have proposed a new problem, namely mining similarity-aware distinguishing sequential patterns with gap constraint (simDSPs). Each element of simDSPs in our study is a similarity subgroup, which can help us discover novel interesting patterns compared to previous methods. Then we presented an efficient algorithm with effective pruning techniques to discover simDSPs. Our experiments verify the effectiveness and efficiency of simDSP-Miner.

For future work, there are many research problems that are worthy of studying. First, we can use top-$k$ instead of $\alpha$ and $\beta$ to define the most distinguishing patterns, so that it is more convenient for users to use simDSP-Miner for pattern discovery. It is also interesting to use simDSPs to construct classifiers for biomedical applications. Furthermore, we plan to apply simDSP-Miner to other kinds of biomedical sequences including the phenotypic data after converting such data into sequences.

ACKNOWLEDGMENT

This work was supported in part by NSFC 61572332, NSFC 81473446, the Fundamental Research Funds for the Central Universities 2016SCU04A22, and the China Postdoctoral Science Foundation 2016T90850.

REFERENCES


