# Finding Repeated Flexible Relational Words in Sequences

Nahla. EL ZANT[1]
and
Henry. SOLDANO[1,2]

[1]LIPN-UMR7030, Institut Galilée, Unversité Paris NORD
Avenue J.-B Clement 93430 Villetaneuse France
[2]Atelier de Bio-Informatique, Université Paris VI, 12 rue Cuvier 75005, Paris

**ABSTRACT.**

Finding regularities in sequences is an important problem in various areas. Regularities are often words (in a strict or somewhat flexible meaning) "repeated" in the sequence, i.e. satisfying some constraints about their occurrence. In this paper we deal with relational values that express what relates two positions in a word or a sequence. Then a strict relational word is defined as the set of relational values corresponding to all pairs of positions within a subsequence. A relational word is flexible if the constraint on a relational value is to belong to a set of relational values. We present here an algorithm, called *KMRCRelat,* which is derived from a previous algorithm for identification of repeated flexible words. KMRCRelat find either the k-length or the longest repeated flexible relational words in a sequence or a set of sequences.

**Keywords :** words, motifs, patterns, sequences, relational patterns, repetitions, data mining.

## 1. INTRODUCTION

Finding regularities in sequences is an important problem in many areas. In most cases the entities forming the sequence are described as symbols. An example of regularity is then a word (in a strict or somewhat flexible meaning) "repeated" in a sequence of symbols. For instance the strict word "ab" occurs at position 1 and 3 in the sequence "ababc". In [8,10] flexibility is obtained by replacing the symbols in a word by predefined groups of symbols. For instance the flexible word "{a,b}{b,c}" occurs at position 1, 3 and also at position 4 in the sequence "ababc" as s[4] belongs to {a,b} and s[5] belongs to {b,c}. Now a word is stated as *repeated* whenever its occurrences satisfy some constraint. For instance "ab" is said to be *2-repeated* in "ababc" since it occurs twice. The corresponding constraint when dealing with a set of sequences is that the word has to occur at least once in at least q sequences (over n). Sequences can represent, after some representation change, any kind of sequential objects. Word-like regularities in sequences have been widely investigated [2,7]. In the present work we deal with cases in which items in the sequence are not only described as symbols but also through what relates them to the other items in the sequence. For instance let items be events (that each have some duration) and let S be a sequence such that the first event (with label a) finishes before the second event (with label b). This may be expressed as s(1)="a," s(2)="b", r(1,2) = "finishesBefore", thus describing the relation between items at position 1 and 2 as a *relational value (see §2.2)*. Complex sequences can then be easily investigated such as the 3D-structure of a protein (i.e. a sequence of amino acids). In the present paper we define relational patterns, namely *flexible relational words,* that extend words and flexible words. We also present an efficient algorithm to search for repetitions of such relational patterns.

## 2. DEFINITIONS & NOTATIONS

In what follows we first give useful definitions and notations when considering repeated flexible words in a sequence, as investigated in [8,10]. Then we extend for our purpose these definitions and notations to relational sequences and relational flexible words.

### 2.1. Sequences and flexible words.

***Definition 1*** *A n-length sequence S is composed of n items. The item at position i has a value $s_i$ belonging to an alphabet $\Sigma$.*

***Definition 2*** *Given an alphabet $\Sigma$, a cover of $\Sigma$ is a set of subsets $C_1, C_2, ..., C_p$ of $\Sigma$ such that*

- $\bigcup_{i=1}^{p} C_i = \Sigma$,
- *None of the subsets $C_i$ is included in another.*

We will denote as *groups* the elements of a cover. For example, let $\Sigma$ ={a, b, c} be the alphabet, then $G$ = {$C_1$={a,b},$C_2$={b,c}} is a cover of $\Sigma$.

***Definition 3*** *A k-length flexible word $M = M_0 ... M_{k-1}$ is an element of the cover product $G^k$. We note $|M|$ the length of M.*

***Definition 4*** *Given a n-length sequence $S=s_1 .... s_n \in \Sigma^n$, and a flexible word $M= M_0 .... M_{k-1} \in G^k$, we say that M has an occurrence (or occurs) at position p in S if, for all $i \in \{0, ..., k-1\}$, $s_{p+i} \in M_i$.*

**Definition 5** *A flexible word M is q-repeated in a sequence S if and only if at least q occurrences of this word are found in S.*

***Definition 6*** *A k-length flexible word w is maximal in a sequence S if and only if there is no k-length flexible*

*word M' such that the set of occurrences of M' includes the set of occurrences of M.*

**Example 1** let $S = abbab$, and $G = \{C_1 = \{a,b\}, C_2 = \{b,c\}\}$, then $M = C1\text{-}C1\text{-}C2$ has an occurrence at position 1 in $S$ since $s_1 = a \in C_1$ and $s_2 = b \in C_1$ and $s_3 = b \in C_2$. Furthermore $M$ is a 2- repeated flexible word in $S$ since it also occurs at position 3 in the sequence $S$. Finally $M$ is not maximal since the word $M' = C1\text{-}C1\text{-}C1$ occurs at positions 1, 2 and 3 in $S$ and so the set of occurrences of $M$ in $S$ is included in the set of occurrences of $M'$ in $S$.

Strict $k$-length words, i.e. elements of $\Sigma^k$, are particular cases of flexible words with a cover made of the singletons of $\Sigma$. Note that searching for flexible words that occurs in at least $k$ of the $m$ sequences of a set of sequences can be performed by first concatenating the sequences and then searching for flexible words that are $q$-repeated in the resulting sequence (provided that some attention is paid to avoid false occurrences). For instance flexible words are useful to find structural repetitions within a set of protein sequences [9]. In what follows we only discuss searching for $q$-repeated flexible words (and later flexible relational words) in a sequence. When computing and counting flexible words occurring in a sequence, useful definitions are the following:

***Definition 7*** *The degeneracy $g$ represents the maximum number of groups to which belongs an element of the alphabet $\Sigma$.*

For example let $\Sigma = \{a, c, g, t\}$ and *let* $C1=\{a, c\}$, $C2=\{c, g\}$ and $C3=\{c, t\}$, then $g=3$ (because $c$ belongs to 3 groups).

***Definition 8*** *The k-prefix $Pref_k(M)$ of a flexible word $M$ is the flexible word that corresponds to the k first positions of $M$ ($k \leq n-1$). In the same way the k-suffix $Suff_k(M)$ of $M$ corresponds to the k last positions of M.*

For example the 2-prefix of $M=C1\text{-}C2\text{-}C1$ is $C1\text{-}C2$ and the 2-suffix of $M$ is $C2\text{-}C1$.

### 2.2. Sequences and relational flexible words.

We will now also consider what relates the items at positions $i$ and $j$ in a sequence:

In what follows each pair (i,j) of items (with i<j) of a sequence has a value $r_{i,j}$ (that we call a relational value) belonging to a relational alphabet $\Sigma_R$. We represent here the relational values of a sequence as a set of delay vectors $\{R^d\}$ where for each vector $R^d$, $R^d[i]$ contains the relational value $r_{i,i+d}$. $R^0[i]$ simply is the non relational value $s_i$.

***Example*** Let $\Sigma_R$= {rb, rs, ra}.The relational information about the 3-length sequence S is represented for instance as:

$R^2$ = *rs*
$R^1$ = *rb - ra*
$R^0 = S$ = *a - a - b*

This means that the relation $r_{1,2}$ between position 1 and 2 has the value *rb*, that $r_{1,3}$= *rs*, and that $r_{2,3}$= *ra*.

For instance let items of the sequence S be events that each have some duration and let us suppose that $r_{i,j}$= *rb* means that event at position $i$ finishes *before* event at position $j$, that $r_{i,j}$= *rs* means that the event at position $i$ finishes at the *same* time as the event at position $j$, and that $r_{i,j}$= *ra* means that the event at position $i$ finishes *after* the event at position $j$. Non-relational values *a, b* simply are labels of the events. Then the previous sequence $S$ is such that: the first event has label *a*, finishes before the second event, and finishes at the same time as the third one. The second event has label *a* and finishes after the third one. The third event has label *b*. $S$ corresponds then to the following configuration of the events:

- *a----------.-end*
- *a----------.-end*
- *b----------.-end*

Note that this information can be inferred from the beginning and ending times of the events. So here the relational values can be computed whenever necessary and do not need to be stored. In what follows we suppose that when considering a sequence $S$ we either have access or we can compute all the corresponding delay vectors $R^d$.

We will now define what is a flexible relational word. We will denote as a *relational cover*, a cover $G_R = \{Cr_1, ..., Cr_t\}$ of the relational alphabet $\Sigma_R$.

For instance let $\Sigma_R = \{rb, rs, ra\}$ be the relational alphabet, then $G_R = \{Cr1=\{rb, rs\}, Cr2= \{rs, ra\}\}$ is a relational cover.

As we have previously defined the relational representation of a sequence as a structured set of non relational values $s_i$ and relational values $r_{i,j}$, we define a relational flexible word $M$ as a structured set of elements of the cover $G$ and of the relational cover $G_R$.

***Definition 9*** *A k-length flexible relational word M is defined as:*

$M = \{ M_i \in G / 0 \leq i \leq k-1\} \cup \{ M_{i,j} \in G_R / 0 \leq i \leq k-1$ *and* $i < j\}$ *and is structured as follows:*

$$
M = \begin{array}{ccccc}
 & M_{0,k-1} & & & \\
 & M_{0,k-2} & M_{1,k-1} & & \\
 & .... & & & \\
 & M_{0,1} & M_{1,2} \ldots & M_{k-2,k-1} & \\
M_0 & M_1 \ldots & M_{k-2} & M_{k-1} &
\end{array}
$$

***Definition 10*** *The flexible relational word $M$ has an occurrence at position $p$ in a (relational) sequence $S$ if the non relational part of $M$ has an occurrence at position $p$ and each relational value $r_{p+i,p+j}$ of $S$ belongs to the corresponding relational group $M_{i,j}$.*

***Example*** Let S= abbab be a sequence, $G = \{C1=\{a, b\}, C2=\{b, c\}\}$ and $G_R = \{Cr1=\{\{ra, rb\}, Cr2=\{rb, rc\}\}$, let $M$ be the following flexible 3-length relational word:

$$
M = \begin{array}{ccc}
 & Cr_2 & \\
 & Cr1 \quad Cr_2 & \\
C_1 & C_1 \quad C_2 &
\end{array}
$$

And let $r_{1,2}$= ra, $r_{2,3}$= rb, $r_{3,4}$= rb, $r_{4,5}$= rc, $r_{1,3}$= rc, $r_{2,4}$= ra, $r_{3,5}$= rb. Then M has occurrences at position 1 (see table 1) and position 3 in *S*.

| | | |
|---|---|---|
| $r_{1,3} \in M_{0,2} = Cr2$ | | |
| $r_{1,2} \in M_{0,1} = Cr1$ | $r_{2,3} \in M_{1,2} = Cr2$ | |
| $s_1 \in M_0 = C1$ | $s_2 \in M_1 = C1$ | $s_3 \in M_2 = C2$ |

**Table 1. Occurrence of the flexible relational word M at position 1 in S.**

Definitions of length, q-repetition, k-prefix and k-suffix of a flexible relational word are unchanged with respect to their definitions for flexible word.

***Example*** let *M* be the following 3-length flexible relational word:

*Cr1*
*Cr2 Cr2*
*C1 C2 C1*,

then:

$Pref_k(M) = \begin{matrix} Cr2 \\ C1 \quad C2 \end{matrix}$, and $Suff_k(M) = \begin{matrix} Cr2 \\ C2 \quad C1 \end{matrix}$

The relational degeneracy $g_R$ represents the maximum number of groups to which belongs an element of $\Sigma_R$.
***Example*** Let $R = \{a, c, g, t\}$ and let $Cr1 = \{a, c\}$, $Cr2 = \{c, g\}$ and $Cr3 = \{t\}$, then $g_R = 2$ (because *c* is an element of two relational groups).

What we present here is an extension of KMRC, denoted as *KMRCRelat*, that finds the q-repeated k-length flexible relational words in a sequence.. We first recall KMRC.

## 3. KMR AND KMRC

KMR has been proposed by Karp, Miller and Rosenberg [6] as a O(N.logk) algorithm for finding exact repeated k-length words in a N-length sequence. The KMRC algorithm [10] is an extension of KMR that search for repeated k-length flexible words. In [10] groups are associated to a similarity relation and intended as maximal cliques of symbols. Two k-length substrings *S1* and *S2* are then considered as similar if the two symbols located at the same position in the two substrings are similar, i.e. if *S1* and *S2* represent two occurrences of a same flexible word. Later, in [8], flexible words are generalized by allowing any cover. Furthermore more flexibility was introduced by allowing errors in occurrences. The original KMRC algorithm is based on the following lemma:

***Lemma 1:*** *Let M be a k-length flexible word, let S be a n-length sequence, and let k' be such that $k/2 \leq k' \leq k-1$, then:*

*i is occurrence of M in S if and only if:*

- *i is an occurrence of $Pref_{k'}(M)$ in S,*
- *i+k'is an occurrence of $Suff_{k'}(M)$in S*

KMRC is iterative. In an iteration Lemma 1 is used with no overlapping, i.e. with k'=k/2, to compute the set of q-repeated 2k-length flexible words using the current set of q-repeated k-length flexible words..

Worst case complexity of computing q-repeated k-length flexible words is $O(N.g^k.log(k))$. However KMRC returns flexible words that are maximal (see Definition 6) and worst-case complexity grows to $O(N.g^{2k}.k.\ log(k))$ because of the inclusion tests to perform at each iteration. Note however that practical CPU time and memory is much lower when computing maximal words and that KMRC does not handle and store words but sets of occurrences that we further refer to as extents.

## 4. KMRCRelat

In this section we present the algorithm KMRCRelat that extends KMRC to relational flexible words. KMRCRelat finds all q-repeated flexible relational words in a sequence S. Indeed KMRC corresponds to the case of *KMRCRelat* where there is only one relational group in the relational cover $G_R$: all relational values are then considered as similar and so each flexible word has the same occurrences in S as the unique corresponding relational word. Note that, as KMRC, KMRCRelat search for maximal words and only deals with their extents.

### 4.1. *KMRCRelat*: inputs and outputs

***KMRCRelat inputs :*** -

- An alphabet of symbols $\Sigma$ together with an alphabet $\Sigma_R$ of relational values.
- A cover $G$ of $\Sigma$ and a cover $G_R$ of $\Sigma_R$ to describe flexible relational words.
- The size $k$ of the words and the quorum $q$ of their occurrences. Optionally $k$ is undefined and longest $q$-repeated words are searched for.

The representation of the input sequence, i.e. the set of delay vectors $R^0=S, R^1, \ldots, R^n$ , is dynamically computed or accessed whenever necessary during the execution of *KMRCRelat*.

***KMRCRelat outputs :*** The set of all k-length q-repeated relational flexible words, represented as their sets of occurrences in *S*. Note that only maximal words (i.e. maximal extents) are returned.

### 4.2. Theoretical bases of *KMRCRelat*.

*KMRCRelat* relies on a variant of Lemma 1 for relational words. Let us consider a $(k+1)$-length flexible relational words $M$. Let $p$ be an occurrence of $M$. Therefore $p$ is also an occurrence of its k-prefix $M_P$ and $(p+1)$ is an occurrence of its $k$-suffix $M_S$. Conversely if $p$ is an occurrence of $M_P$ and $(p+1)$ is an occurrence of $M_S$, then to ensure that $p$ is an occurrence of $M$ we only have to check whether the value relating $p$ to $p+k$ in $S$ belongs to the relational group $M_{0,k}$ (see figure 1):

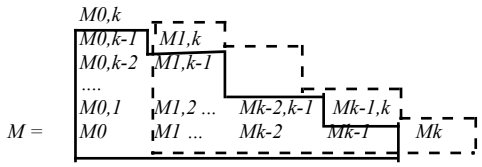



***Figure 1 Reconstructing M from its k-prefix (plain line) and its k-suffix (dashed line).***

We obtain then the following lemma :

***Lemma 2****: p is an occurrence of the (k+1)-length flexible relational word M if and only if:*

- *p is an occurrence of $Pref_k(M)$,*
- *p+1 is an occurrence of $Suff_k(M)$ and*
- $r_{p,p+k} \in M_{0,k}$

### 4.3. *KMRCRelat* steps

*KMRCRelat* is an iterative algorithm. The initialization consists in computing the *q*-repeated 1-length flexible words. The $n^{th}$ iteration computes the *(n+1)*-length *q*-repeated relational flexible words using the *n*-length *q*-repeated relational flexible words as suggested by Lemma 2. As in KMRC, the set of *q*-repeated *n*-length flexible words is represented as both a vector *V* where *V[i]* contains the flexible words (represented as numbers) occurring at position *i* in *S*, and a list *P* of stacks which element *P[#j]* contains the extent, i.e. the set of occurrences, of the $\#j^{th}$ flexible word. In what follows the size of the list *P* is the number of flexible words to represent. Furthermore KMRCRelat uses a vector *W* where *W[i]* contains at the $n^{th}$ iteration all the relational groups containing the relational value $r_{i,i+n}$ found in *S*.

We will examplify KMRCRelat on the following example: :

***Example***.

let $\Sigma$ = {a, b, c}, *G={ C1={a, b}, C2={b, c}},*

$\Sigma_R$*= {r1, r2, r3}}, $G_R$= {Cr1={r1, r2}, Cr2={r2, r3}}*

*S*=

*a a b a a b a c*

and

$R^1$= *r1, r2, r1, r2, r1, r1, r3*

As in this example we only compute 2-length relational flexible words, delay vectors $R^d$ for $d>1$ are useless for our purpose.

At the beginning of the algorithm, *q*-repeated 1-length flexible words are computed by initializing the vector *V*.

In our example there are two 1-length extents numbered as *#1 (C1)* and *#2 (C2)*. *V* is therefore as follows:

| | | *b∈C2=#2* | | | *#2* | | |
|---|---|---|---|---|---|---|---|
| *a∈C1=#1* | *#1* | *b∈C1=#1* | *#1* | *#1* | *#1* | *#1* | *#2* |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |

and the corresponding list of stacks *P* is :

| | |
|---|---|
| *#1* | 7 6 5 4 3 2 1 |
| *#2* | 8 6 3 |

We now describe the different steps of the $n^{th}$ iteration building the *(n+1)*-length *q*-repeated relational flexible words from the *n*-length *q*-repeated relational flexible words. Building the 2-length words from the 1-length words of our previous example will be used to illustrate the iteration.

Steps 1 and 2 are similar to the corresponding steps in KMRC except that in KMRC *n*-prefixes and *n*-suffixes do not overlap, thus resulting in *(2.n)*-length flexible word.

***Step 1:*** Compute the list *P* corresponding to the vector *V*. Both *P* and *V* represent the set of *n*-lengths *q*-repeated flexible relational word. Each place in the list *P* contains a set of occurrences, i.e. an extent. When *n*=1 each place in the list *P* corresponds to a group of *G*.

***Step 2*:** Compute the occurrences of the (k+1)-length *q-repeated words* satisfying the condition 1) and 2) of lemma 2. These occurrences will be pushed into the elements of a list Q of stacks. The size of Q is the same as the size of the list P and its elements are initially empty stacks. More precisely, for each occurrence i in P[#j] we consider the extents in V[i+1]. For each extent #e in V[i+1] we add i to Q[#e]. The list Q of our example for n=1 is presented in figure 2. Note that the separator ";" indicates that an extent P[#j] in P has been exhausted.

| | |
|---|---|
| *_#1* | 1 2 3 4 5 6 **;**3, 6 |
| *_#2* | 2 5 7; |

***Figure 2:The list of stacks Q.***

The first extent in Q is {1, 2, …, 6} and represents the occurrences of M= C1-C1, the second extent {3, 6} represents the occurrences of M= C2 C1 and the third one {2 ,5 , 7}corresponds to M= C1 C2.

Extents are further filtered in order to eliminate all extents that do not respect the quorum q. For example, if q= 3, the second extent {3,6} is deleted.

At the end of this step the list Q contains all the extents obtained by overlapping two n-length flexible words that occur at contiguous positions as specified in Lemma 2.

***Step 3*:** Compute the vector *W*. For this purpose KMRCRelat have to access or compute the delay vector $R^n$ ($R^n$*[i]* contains the relational values $r_{i,i+n}$ between the positions *i* and *i+n* in *S*). In *W* each relational group is represented as an integer. Regarding our example, at the first iteration *W[i]* contains the information regarding the relation between positions *i* and *i+1*. The vector *W* of our example is presented in figure 3.

| | r2∈#2 | | #2 | | | | |
|---|---|---|---|---|---|---|---|
| r1∈#1 | r2∈#1 | #1 | #1 | #1 | #1 | #2 | |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | |

***Figure 3: The vector W***

***Step 4*:** Add the third constraint of Lemma 2, i.e. the relational one, to extents computed in step 3. This step is similar to step 2. The resulting extents will be represented in a list of stacks *Qrelat* whose size is the number of relational groups in $G_R$, and whose elements are initially empty. The algorithm uses *W* and *Q* in order to construct *Qrelat*. More precisely, for each occurrence *i* of the $\#k^{th}$ extent in *Q*, and for each relational group *#c* in *W[i]* we add *i* to *QRelat[#c]*. Here again the separator ";" indicates that the $\#k^{th}$ extent in *Q* has been exhausted. The list *Qrelat* of our example is presented in figure 4 (here *#c = Cr1* or *Cr2*).

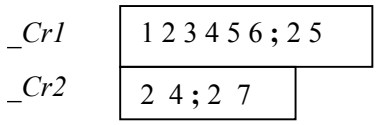


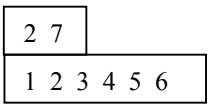

***Figure 4: The list of stacks Qrelat.***

The extent {2, 7}, for instance, corresponds then to the following relational flexible word:

$$M = \begin{matrix} Cr2 \\ C1 \quad C2, \end{matrix}$$

and has been obtained from the extent {2,5,7} representing the flexible word *C1-C2* by adding the constraint $r_{p,p+1} \in Cr2$.

***Step 5*:** Eliminate non-maximal extents and extents smaller than *q*, then renumber the remaining extents, and return the vector *V* representing the *(n+1)*-length *q*-repeated flexible relational words. Now *V[i]* represents the (n+1)-length *q*-repeated flexible relational words that occurs at position *i*. *V* can now be used as an input for a new iteration. In our example the next step 1 results in the following list *P*.




The algorithm stops when it has computed all the *k*-length (or all the longest) *q*-repeated flexible relational words.

### 4.4. Complexity issues.

Concerning worst case time complexity, KRMCRelat, (in the same way as KMR, KMRC and related algorithms) relies on the following observation: a naive generate and test algorithm that evaluates the $O(|G|^k . |G_R|^{k^2})$ *k*-length flexible relational words in a N-length sequence *S* have to compute $O(N.|G|^k . |G_R|^{k^2})$ occurrence tests. As $|G|$ and $|G_R|$ may be large, the computational cost is high even for small words. However, the total number of occurrences of *k*-length flexible relational words in *S* is bounded by $O(N.|g|^k . |g_R|^{k^2})$. As a consequence, when considering non-flexible (i.e. strict) words, we have $g = g_R = 1$ and so there are only *O(N)* such occurrences in i. In the "strict" case, KMRCRelat is then $O(N.k)$. However when degeneracy numbers *g* and $g_R$ increase, the corresponding flexibility results in a much higher number of occurrences to handle within extents. Spatial complexity is then also high: because of the breadth first nature of KMRCRelat, all *k*-length extents are stored at the same time in memory. This allows the selection of "maximal" *k*-length words that in practical cases is very efficient and meaningful, but can also result in serious storage difficulties. We do not further discuss here the overall complexity of KMRCRelat. Simply note that it heavily depends on the cost of the inclusion tests (step 5) but that elimination of non-maximal words is here unavoidable. From our experiments we simply observe that as long as we restrict to small relational words ($k < 25$) or limit flexibility (small values of g and $g_R$) the huge amount of extents that are handled result in a rather small number of "maximal" k-length flexible relational words. More experiments have to be performed to discuss the practical behavior of KMRCRelat.

## 5. *KMRCRELAT* APPLICATIONS

*KMRCRelat* can be applied to find repeated flexible relational words in various kind of sequences. A toy example is given in §2.2 when considering sequences of events where relational values concern the relative positions of events. A similar situation concern secondary structures in RNA sequences (the RNA sequence is then considered as a set of helices, i.e. a set of subsequences that can be embedded or can overlap). In [1], an *ad-hoc* algorithm is used to compute strict relational words representing secondary structures that are shared by a set of RNA sequences. A promising area is the search for repeated motifs in mono or multidimensional signals or series. In this section we discuss how flexible relational words can represents geometric motifs in sequences of points described in a 3D space. The underlying application is the search for repetitions within 3D-structures of proteins. A relational value in this case represents a distance between two points. A sequence of n points is so described using the set of the n(n-1)/2 internal distances between its elements. Such descriptions have been previously used in [4] to find structural motifs in the 3D-structure of a protein that are repeated in a set of target 3D-structures. In figure 5 we find the description of the subsequences *p1-p2-p3-p4* and *p6-p7-p8-p9* in a 9-length sequence *S*. The distances are represented as labels of the edges relating the nodes associated to the points. In figure 6 the same information is represented as two relational subsequences (in this example, points have no labels and so non-relational value $\{s_i\}$ are omitted).

Here we consider that prior discretization of distances has been performed. The alphabet of relational values is then $\Delta_R = \{1,3, .., \}$. A relational group here is an ordered subset $Crj = \{j, ... , j + d\}$ where *d* represents a tolerance: two internal distances $ri,i+b$ and $ri',i'+b$ are then considered as similar if they belongs to a same group i.e. if $| ri,i+b) - r(i',i'+b) | \leq d$. Let us consider in our

example that d=*3*, then we find occurrences of the 4-length relational flexible repeated word *M* represented in figure 7 at positions 1 and 6 in *S*. These occurrences correspond to the subsequences represented in figure 6.

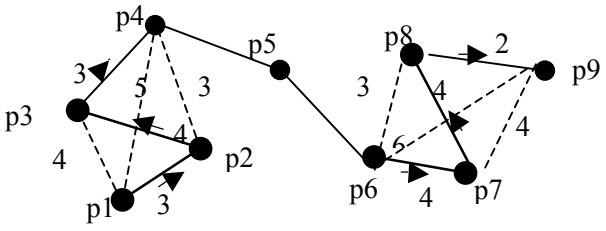



***Figure 5: A numeric structural sequence S defined as a sequence of 3-D points.***

| |
|---|
| *r1,4 = 4*<br>*r1,3 = 3  r2,4 = 3*<br>*r1,2 = 3  r2,3 = 4  r3,4 =3* |
| *r6,9 = 6*<br>*r6,8 = 3  r7,9 = 5*<br>*r6,7 = 5  r7,8 = 4  r8,9 =2* |

***Figure 6:Relational representation of two subsequences in S.***

| |
|---|
| Cr3<br>Cr1  Cr2<br>Cr2  Cr1  $Cr_1$ |

***Figure7: M: relational word.***

A work is in progress to evaluate results obtained in searching for repeated 3D structural motifs on proteins backbones when using KMRCRelat and internal distances based descriptions. The results are compared to those obtained with GOK [9,5] that uses KMRC together with a non-relational representation of the protein backbone as a sequence of pair of angles. The theoretical superiority of the relational approach relies here on avoiding a possible addition of angle errors when defining 3D-substructures as similar.

## 6. CONCLUSION AND PERSPECTIVES

In this paper, we have introduced *flexible relational words* together with an efficient breadth first algorithm that search for *repeated* flexible relational words in a relational sequence. The algorithm extends a previous algorithm designed for flexible non-relational words, and is easily extended to search for repetitions in a set of relational sequences. The notion of relational object or pattern is not new: all first order representations in machine learning and data mining deals with relational objects and patterns. Less attention has been paid to the particular case of relational representation of sequences. Various patterns of interest have however been investigated: for instance *structured models* [2] or *chronicles* [3] both represent a relational pattern built from several non-relational words (or symbols) together with constraints on their occurrences on the sequence. The point of view presented in the current work is to consider particular relational patterns that extends the notion of "word", i.e. a pattern in which 1) all successors of a starting position are constrained 2) the constraint is complete, with some flexibility, with respect to the relational description. As in the case of non-relational words, further extensions could add more flexibility as allowing for errors and gaps when defining occurrences of a relational word. The method is currently under investigation for various kinds of sequential data. Particular attention is paid to the type of patterns and relational descriptions that result useful when dealing with sequential data.

## References


[1] Bouthinon D., Soldano H. (1999):A new method to predict the consensus secondary structure of a set of unaligned RNA sequences. **BioInformatics**, Vol.15, n°10, pp. 72-80, 1999, Oxford University Press

[2] Crochemore M. and Sagot M.-F.(2002), "Motifs in sequences: localization and extraction", in A. Konopka *et al.* (eds.), **Handbook of Computational Chemistry**, Marcel Dekker Inc., 2002, in press**.**

[3] Dousson C.and Vu Duong T.:(1999) Discovering Chronicles with Numerical Time Constraints from Alarm Logs for Monitoring Dynamic Systems. **proceedings of IJCAI** : 620-626 .

[4] Escalier V.,Pothier J., Soldano H.,and Viari A. (1998) "Pairwise and multiple identification of three-dimensional common substructures in proteins." **J. Computational Biology**, Vol. 5, n° 1, 41-56

[5] Jean, P., Pothier, J., Dansette, P., Mansuy, D. and Viari, A. (1997)." Automated multiple analysis of protein structures: application to homology modeling of cytochromes P450", **Proteins: Structure**, **Function, and Genetics**, 28, 1-16

[6] Karp, R.M. ; Miller, R.E. ; Rosenberg, A.L. (1972) : "Rapid identification of repeated patterns in strings, Trees and arrays", **4th ACM Symposium. Theory of computing**, pages 125-136.

[7] Mannila, H., Toivonen, H and Verkamo. A. I., (1997) "Discovery of frequent episodes in event sequences" .**Data Mining and Knowledge Discovery**, 1(3), 259-289, 1997.

[8] Sagot M F., Viari and Soldano H.(1997) "Multiple sequence comparison: a peptide matching approach", **Theor. Comp. Sci.** 180 115-137

[9] Sagot M. F., Pothier J., Viari A., Soldano H. (1995): "Finding Flexible patterns In A Text – An Application To 3D Molecular Matching". **Cabios**, 11:59-70..

[10] Soldano, H., Viari, A. and Champesme, M. (1995) "Searching for flexible repeated patterns using a non-transitive similarity relation." **Pattern Recog. Lett**., 16, 243-246.