Searching for flexible repeated patterns using a non transitive similarity relation.

Henri Soldano,
Laboratoire d'Informatique de Paris Nord, URA-CNRS 1507, Institut Galilée, Université Paris Nord, Av. J.B. Clément, 93430 Villetaneuse, France, and Atelier de Bio-Informatique, section Physique-Chimie, Institut Curie, 11 rue Pierre et Marie Curie, 75005 Paris, France

Alain Viari,
Laboratoire de Physique et Chimie Biomoléculaires, URA-CNRS 198 and Atelier de Bio-Informatique, section Physique-Chimie, Institut Curie.

Marc Champesme
Laboratoire d'Informatique de Paris Nord, URA-CNRS 1507, Institut Galilée, Université Paris Nord, and Atelier de Bio-Informatique, section Physique-Chimie, Institut Curie.

Abstract.

Given a reflexive and symmetric, but not necessarily transitive, similarity relation defined on an alphabet of symbols, two objects of size k are related if, in each position, their symbols are related. Then, given a set of objects we are interested in maximal subsets of related objects. We give some general properties of these subsets and we propose algorithms for identifying them in the particular case of k-length substrings in a string. These algorithms derive from the Karp, Miller and Rosenberg algorithms for identification of repeated patterns.

Introduction.

In a previous work Karp and al. (1972) have proposed various algorithms, hereafter referred to as KMR algorithms, to identify repeated patterns in a structure of size N (string, array or tree). In their approach the patterns to identify correspond to exact matches between objects. For instance, two k-length substrings match if, in each position, the same symbol is present in both substrings. However some situations require a more flexible matching and patterns corresponding to similar, rather than strictly identical, objects are searched for. As an example, in molecular biology similar fragments of amino acid sequences may exhibit a similar 3D-structure or biological
function. In this work we suppose that a symmetric and reflexive relation $R$, expressing a similarity, is defined on the alphabet. This leads to a reflexive and symmetric relation $R_k$ on structured objects of size $k$, defined as follows: two objects are similar if, in each position, the corresponding symbols are related by $R$. Now, given a set of objects, our purpose is to find all the maximal cliques of the relation $R_k$, i.e., the maximal subsets in which any pair of objects $(x,y)$ is such that $x$ and $y$ are related. Then each maximal clique of $R_k$ defines a pattern.

This paper is divided in two sections. Section I presents some definitions and properties concerning the maximal cliques of $R_k$. This includes some constructive properties for structured objects of size $k$. Section II presents some algorithms, implementation, and experiments on random strings addressing the following problems:

1) find all maximal cliques of $R_k$, which size is greater than 2 (i.e., repeated patterns) when the objects are the $k$-length substrings of a $N$-length string.
2) find the largest $k$ for which such repeated patterns exist in a $N$-length string.

I - Properties of the relation $R_k$

I-1 Notations

Throughout this paper, we will use the following notations:

$A = \{s_1, \ldots, s_s\}$ is an alphabet of $s$ symbols.

$R$ is a symmetric and reflexive (but not necessarily transitive) relation, defined on $A$, which represents a similarity between the symbols of $A$.

An "object" of size $k$ is a labeled and oriented structure composed of $k$ elements of $A$. Each element is assigned to a certain "position".

For instance, a $k$-length string is a structure of size $k$, and an $l$-array is a structure of size $k=l.c$.

It is important to distinguish between the name of an object and its "value" (i.e., the components of its structure).

More precisely, if $x$ is an object of size $k$, we will note its value:

$V(x) = (V_1(x), V_2(x), \ldots, V_k(x))$ , where $V_i(x)$ belongs to $A$
For example, consider a string \( S = (s_1, s_2, \ldots, s_N) \) of length \( N \).

The set \( X = \{1, 2, \ldots, y, \ldots, N-k+1\} \) of positions in the string is a set of \( N-k+1 \) objects representing all the \( k \)-length substring of \( S \). An object (substring) of size \( k \) will be referred to as \( j \) (its starting index) and its value is \( V(j) = (s_j, s_{j+1}, \ldots s_{j+k-1}) \).

Unless specified, we will denote by \( X \) the set of all objects of size \( k \) we are considering and by \( N \) the size of \( X \).

I-2 Definitions

The comparison between two objects of the same size will result from the comparisons of symbols in each position. To this purpose, we define the following relation:

\[ R_k \text{ stands for the symmetric and reflexive relation on } X, \text{ defined as:} \]
\[ x R_k y \text{ if } i \in \{1, 2, \ldots, k\} \quad V_i(x) R V_i(y) \quad (\text{def. 1}) \]

A clique of the relation \( R_k \) on \( X \) is a subset of \( X \) in which any pair of elements \( (x,y) \) is such that \( x R_k y \). A clique is called maximal if by adding any other element to it, the resulting subset is no more a clique.

Hereafter, we denote by \( C_X \) a clique of the relation \( R_k \) on \( X \).

In the same manner, one can consider the cliques of the relation \( R \) on \( A \). To distinguish them from the cliques of \( R_k \), we will use a lowercase \( c \), for such cliques and an uppercase \( C \) for the cliques of \( R_k \).

Let \( \{c_1, c_2, \ldots, c_k\} \) be a set of \( k \) cliques of \( R \). We will call the Cartesian product \( c = c_1 \times c_2 \times \cdots \times c_k \) a cliques product of \( R \).

Moreover, if all the \( c_i \) are maximal cliques, then \( c \) will be called a maximal cliques product.

We will use the following example throughout the paper:

Let \( A = \{a, b, c, d, e\} \). \( R \) is a symmetric and reflexive relation whose graph is represented on figure 1. The maximal cliques of \( R \) are the sets:
Let us consider a string \( S = adbeb \), and the set \( X = \{1, 2, 3\} \) of the 3-length substrings of \( S \) represented by their starting indexes. Then we have, for instance, \( 1R_33 \) since \( aRb, dRe, \) and \( bRb \). Moreover \( \{1,3\} \) is a maximal clique of \( R_3 \) since it cannot be extended.

**I-3 Properties of \( R_k \)**

We give now several properties of maximal cliques of \( R_k \).

Let \( f \) be the function that relates any clique \( C_X \) of \( R_k \) to the Cartesian product \( e_1 \circ e_2 \circ \ldots \circ e_k \), where each \( e_i \) stands for the subset of \( A \) defined as:

\[
e_i = \bigcup_{x \in C_X} V_i(x) \quad \text{(def. 2)}
\]

**Proposition -1-**

\( f(C_X) \) is a cliques product of \( R \). \hspace{1cm} \text{(prop. 1)}

**Proof** For each position \( i \), the values of \( V_i \) corresponding to the elements of \( C_X \) are related by \( R \) (def. 1). As a consequence \( e_i \) is a clique of \( R \).

**Proposition -2-**

Given a maximal cliques product \( C \) of \( R \), there exists at most one maximal clique \( C_X \) of \( R_k \) such that \( f(C_X) \subseteq C \). \hspace{1cm} \text{(prop. 2)}

**Proof** Let \( C_X \) and \( C_X' \) be two maximal cliques of \( R_k \), such that:

\[
f(C_X) \subseteq C \quad \text{and} \quad f(C_X') \subseteq C
\]

then for each position \( i \), we obtain (def. 2):

\[
e_i \circ e_i' = \bigcup_{x \in C_X \cap C_X'} V_i(x) \subseteq C_i
\]
This means that $C_X \sqcap C'_X$ is a clique of $R_k$, and, since $C_X$ and $C'_X$ are both maximal cliques, this necessarily implies that $C_X = C'_X$.

From these propositions it comes:

**Proposition-3.**

Let $f_\_\_be the restriction of $f$ to the set of maximal cliques of $R_k$, then $f_\_\_$ is one to one. (prop. 3)

**Proof.** Let $C_X$ and $C'_X$ be two maximal cliques of $R_k$ such that $f(C_X) = f(C'_X)$. Since $f(C_X)$ is a clique product (prop. 1), there exists at least one maximal cliques product $\square$ such that $f(C_X) = f(C'_X) \sqcap \square$. Then it follows (prop. 2) that $C_X = C'_X$.

**Proposition-4.**

Given $e_k$ : the number of maximal cliques of $R_k$  
$m$ : the number of maximal cliques of $R$  
$g$ : the maximum number of maximal cliques of $R$ containing the same symbol of $A$  
$\square_k$ : the set of maximal cliques of $R_k$

We have the following upper bounds:

a) $\square \card(C_X) \leq N \cdot g^k$ (prop. 4a)  

b) $e_k \leq \min(N \cdot g^k, m^k)$ (prop. 4b)

**Proof.**

a) By definition of $g$, each element $x$ of $X$ is such that $V(x)$ belongs to less than $g^k$ maximal cliques products. Then, according to proposition 2, $x$ belongs to less than $g^k$ maximal cliques of $R_k$. Since $N$ is the size of $X$, the inequality is proved.

b) The number of maximal cliques products is equal to $m^k$. Then, according to proposition 2, we have $e_k \leq m^k$. Furthermore, since each maximal clique of $R_k$ contains at least one element, it follows from (4a) that $e_k \leq N \cdot g^k$. 
In the particular case where $R$ is an equivalence relation, $g = 1$ and these inequalities simply mean that the classes of $R_k$ form a partition of $X$, and that the number of classes is less than $\min(N, m^k)$. The number $g$, hereafter referred to as the "degeneracy" of $R$, represents a discrepancy from $R$ to an equivalence relation. One could also consider, as a more realistic measure in practical cases, the averaged degeneracy $\overline{g}$, obtained by assigning an a priori probability to each symbol. Then, as an example, for a random $N$-length string the averaged quantities in (4a-b) become of order $O(N, \overline{g}^k)$.

**Proposition-5.**

Let $C_X$ be a clique of $R_k$, and $C'_X$ be a maximal clique of $R_k$, then we have:

$$C_X \sqsubseteq C'_X \quad f(C_X) \sqsubseteq f(C'_X) \quad \text{(prop. 5)}$$

**Proof**  "$\sqsubseteq$": follows from the definition of $f$.

"$\sqsubseteq$": Let $x$ be an element of $C_X$, then, by definition of $f$, $V(x) \sqsubseteq f(C_X)$ and thus $V(x) \subseteq f(C'_X)$. In the same manner, for any element $x'$ of $C'_X$ we have $V(x') \subseteq f(C'_X)$. Hence $V(x)$ and $V(x')$ belong to the same clique product $f(C'_X)$. It follows from definition 1, that $x R_k x'$. Since $C'_X$ is a maximal clique of $R_k$, then $x$ must belong to $C'_X$, and then $C_X \sqsubseteq C'_X$.

Hence, $f$ appears as a "characteristic function" of cliques of $R_k$. It allows us to compare maximal cliques (prop. 3), and to check whether a clique is included into a
given maximal clique (prop. 5) in a constant number of operations with respect to N. Furthermore we give upper bounds both for the total number of objects in the whole set of maximal cliques of $R_k$, and for the size $e_k$ of this set. These bounds are linear with respect to N, and depend on the degeneracy g of R.

The previous properties are quite general in the sense that they do not depend upon a particular structure but only upon its size. We will now focus on other properties allowing step by step constructions of maximal cliques of $R_k$, for particular structures.

**I-4 Constructing maximal cliques of $R_k$ for k-length substrings of a N-length string**

In a previous work, Karp et al. (1972) have addressed the problem of identifying repeated substructures of fixed size in a structure of size N (string, array or tree). In their work, the relation $R$ between the symbols was the identity. In the case of repeated substrings in an N-length string, the algorithms they have proposed, were based on two lemmas:

- **KMR Lemma-1**: $x R_{a+b} y$ if $x R_a y$ and $x+b R_{a} y+b$ (with $b \leq a$)
- **KMR Lemma-2**: $x R_{a+b} y$ if $x R_{a} y$ and $x+a R_{b} y+a$

These lemmas allow the construction of the $R_{a+b}$ relation starting from the $R_a$ (and $R_b$ in lemma-2) relation. When $R$ is not transitive, these lemmas still hold (since they only involve pairwise comparisons).

**I-4-1 Constructing maximal cliques of $R_1$**

In the case of an N-length string $S = (s_1, s_2, ... , s_N)$, and considering the 1-length substrings, $X$ is the set $\{1, 2, ..., y, ..., N\}$ of all positions in $S$. (see section I-1).

Let us denote by:

- $\{C_1, C_2, ..., C_m\}$ the m maximal cliques of $R$,
- $\{C_X^1, C_X^2, ..., C_X^m\}$ the subsets of $X$ defined as:
  
  $C_X^i = \{j \in X \mid s_j \in C_i^1\}$
Then we have the following properties:

**Proposition 6**

- \( C_X^i \) is a clique of \( R_1 \). \hspace{1cm} (prop. 6a)

- The set of maximal cliques of \( R_1 \) is included in \( \{ C_X^1, C_X^2, \ldots, C_X^m \} \). \hspace{1cm} (prop. 6b)

**Proof** (6a) For any \( x, y \) in \( C_X^i \), we have \( s_x R s_y \) and thus \( x R_1 y \).

(6b) Let \( C \) be a maximal clique of \( R_1 \), then \( c = f(C) \) is a clique of \( R \) (prop. 1). Thus, there exists a maximal clique \( C^i \) of \( R \), such that \( C \uplus C^i \). Then, since \( C_X^i \) contains all the elements \( j \) such that \( s_j \uplus C^i \), it follows that \( C \uplus C_X^i \). But, since \( C \) is a maximal clique of \( R_1 \), this necessarily implies that \( C = C_X^i \).

Hence, the set of maximal cliques of \( R_1 \) is obtained by constructing \( \{ C_X^1, C_X^2, \ldots, C_X^m \} \), and by eliminating all the \( C_X^i \) which are included in any other \( C_X^j \).

In our previous example (section I-2) we have with \( S = abdeeb \):

\[
C_X^1 = \{1, 3, 5\}, \quad C_X^2 = \{2, 3, 5\}, \quad C_X^3 = \{2, 4\}, \quad C_X^4 = \{1, 4\}
\]

Note that, in that case all the \( C_X^i \) are maximal.

**I-4-2 Step-by-step construction of maximal cliques of \( R_k \)**

Given a set \( E \) of indices, let us denote \( E + d \) (resp. \( E - d \)) the set obtained by adding (resp. subtracting) the integer \( d \) to each index in \( E \). Consider a clique \( C_X \) of \( R_a \), then any pair \( \{x, y\} \) belonging to \( (C_X)_b \) is such that \( x + b R_a y + b \).

The following properties are established from the KMR lemmas:

**Proposition 7a (resp. proposition 7b)**

Let

- \( C_X \) be a clique of \( R_a \)
- \( C'_X \) be a clique of \( R_a \) (resp. \( R_b \))

then
\( C_X \uplus (C'_X)_{-b} \) (resp. \( C_X \uplus (C'_X)_{-a} \)) is a clique of \( R_{a+b} \) (prop. 7a, (resp. 7b))

**Proof** proposition 7a:
For any pair \( \{x, y\} \subseteq C_X \uplus (C'_X)_{-b} \), we have:
- \( x R_a y \) since \( \{x,y\} \subseteq C_X \)
- \( x+b R_a y+b \) since \( \{x,y\} \subseteq (C'_X)_{-b} \)

Then, according to KMR-lemma-1, it comes that \( x R_{a+b} y \).

As a consequence \( C_X \uplus (C'_X)_{-b} \) is a clique of \( R_{a+b} \).

( proof of proposition 7b is similar.)

**Proposition 8a (resp. proposition 8b)**

Let \( C''_X \) be a maximal clique of \( R_{a+b} \), then there exists a maximal clique \( C_X \) of \( R_a \) and a maximal clique \( C'_X \) of \( R_a \) (resp. \( R_b \)) such that:

\[
C''_X = C_X \uplus (C'_X)_{-b} \quad \text{(resp. } C_X \uplus (C'_X)_{-a} \text{) } \quad \text{(prop. 8a, (resp. 8b))}
\]

**Proof** proposition 8a:
For any pair \( \{x, y\} \subseteq C''_X \), we have \( x R_{a+b} y \) and thus, according to KMR lemma 1, we have \( x R_a y \) and \( x+b R_a y+b \).

From \( x R_a y \), it follows that \( C''_X \) is a clique of \( R_a \) and thus there exists a maximal clique \( C_X \) of \( R_a \) such that \( C''_X \subseteq C_X \).

From \( x+b R_a y+b \) it follows that \( (C''_X)_{-b} \) is a clique of \( R_a \) and thus there exists a maximal clique \( C'_X \) of \( R_a \) such that \( (C''_X)_{-b} \subseteq C'_X \) and therefore \( C''_X \subseteq (C'_X)_{-b} \).

As a consequence we have \( C''_X \subseteq C_X \subseteq (C'_X)_{-b} \).

However, according to proposition 7a, \( C_X \uplus (C'_X)_{-b} \) is a clique of \( R_{a+b} \) and since \( C''_X \) is supposed to be maximal, the inclusion necessarily implies that \( C''_X = C_X \uplus (C'_X)_{-b} \).

( proof of proposition 8b is similar.)

Starting from the set of maximal cliques of \( R_a \) (resp. \( R_{a+b} \)), we will obtain all the maximal cliques of \( R_{a+b} \) by performing all the intersections mentioned above.
Some intersections will produce non maximal cliques which can be eliminated by checking if there exists any other clique which includes them.

Let us consider again our previous example (section I-2).

When \( a = b = 1 \), the two lemmas are equivalent, and we will use the maximal cliques \( C_X \) of \( R_1 \): \[
\{1, 3, 5\}, \{2, 3, 5\}, \{2, 4\}, \{1, 4\}\] together with their corresponding \( (C_X)^{-1} \): \[
\{0, 2, 4\}, \{1, 2, 4\}, \{1, 3\}, \{0, 3\}\].

Performing all the intersections and removing non maximal cliques yields:
\[
\{2, 4\}, \{1, 4\}, \{1, 3\}\] as the maximal cliques of \( R_2 \). They correspond to the substrings \{db, eb, ad, eb, ad, be\}.

This clearly leads to a step-by-step algorithm which takes as input an N-length string and a relation R and which produces all maximal cliques of \( R_k \).

II Algorithms, implementation and experiments.

II-1 Algorithms.

In this section, we address the problem of finding all the repeated k-length substrings in a string, according to a relation R between symbols. In the previous section, we have proposed a method to construct all maximal cliques of \( R_k \) in an N-length string. This clearly solves the problem since the repeated k-length substrings are the maximal cliques of \( R_k \) whose size are larger than two. The repeated patterns are constructed by using the previous method except that, at the end of each step, the maximal cliques which size is less than two are removed. This strategy is justified since if a given maximal clique \( C_X \) of \( R_k \) is of size less than two, then its intersection with any set, will also be of size less than two and thus will not produce a longer repeated substring.

The first algorithm solves the following problem:

**Problem 1:** Given an alphabet A, a symmetric and reflexive relation R, and a string S, find all its repeated substrings of length k.

This problem generalises, by using a similarity relation R rather than the identity, a problem solved by KMR.

**Algorithm 1:**
(1) - Construct the set of maximal cliques of R.
(2) - Construct the set $L_1$ of repeated substrings of length 1 as seen in §I-4-1 except that all cliques of size less than two are removed before removing non maximal cliques of $R_1$.

Initialise $i$ to 1.

(3) - Repeat while $2i \leq k$

Construct the set $L_{2i}$ of repeated substrings of length $2i$, by using $L_i$ as seen in §I-4-2: KMR lemma-1 is used with $a = b = i$. The intersections mentioned in proposition-7a concern only the maximal cliques of $R_i$ belonging to $L_i$.

Remove the resulting cliques of $R_{2i}$ whose size is less than two.

Remove non maximal cliques by using an inclusion test.

Multiply $i$ by 2.

(4) - if $k = 2i$

return $L_{2i}$ as result.

else

construct $L_k$ as in step (3) by using KMR lemma-1 with $a = 2i$, $b = k-2i$.

return $L_k$.

The second algorithm solves the following problem:

Problem 2: With the same inputs as in problem 1, find the largest integer $k_{\text{max}}$ such that it exists at least one repeated substring of length $k_{\text{max}}$, and return these substring(s).

As above, this generalises a problem solved by KMR.

Algorithm-2:

(1) - same step (1) as in algorithm 1
(2) - same step (2) as in algorithm 1
(3) - Repeat while $L_i$ is not empty

same operations as step (3) in algorithm 1

(4) - Perform a binary search by using KMR lemma-1. Namely: use $L_{i/2}$ to construct $L_{3i/4}$. If this latter is not empty then use it to construct $L_{7i/8}$ else use $L_{i/2}$ to construct $L_{5i/8}$, and so on.
II-2 Implementation.

Because of the similarity of the following implementation to those described in Karp (1972), we have followed the same notations whenever possible.

At a given step, say that $R_a$ has $e_a$ maximal cliques labeled from 1 to $e_a$. The occurrence of these cliques on the string is represented as an $N-a+1$ place vector of lists $v_a = (v_a(1), v_a(2), \ldots, v_a(N-a+1))$, where each $v_a(i)$ is the list of all the labels of the maximal cliques of $R_a$ to which position $i$ belongs.

These lists are actually implemented as pushdown stacks with the traditional 'Push' and 'Pop' operations. However at some steps, we need to read down the content of a stack without actually popping the values. For this purpose, we provide the stack with an additional 'Downread' operation.

Now we have to construct $R_{a+b}$ from $R_a$, by using KMR lemma-1 and the above representation. Assume that we have at our disposal the vector $v_a$ stored as indicated above, and two $e_a$ place vectors of stacks initially empty available. Call them $P = (P(1), P(2), \ldots, P(e_a))$ and $Q = (Q(1), Q(2), \ldots, Q(e_a))$.

**II-2-1 Pseudocode.**

```
// Step-1 Built P
For i = 1 to N-a+1
    While $v_a(i)$ is not exhausted
        c <-- Downread($v_a(i)$)
        Push i into P(c)
    EndWhile
EndFor
// P gives an explicit representation of the maximal cliques of $R_a$,
// one maximal clique in each P(c). This is the dual representation
// of $v_k$ which gives, for each position, the maximal cliques
// labels at this position.

// Step-2 Built Q
For j = 1 to $e_a$
    While P(j) is not empty
        s <-- Pop(P(j))
```

If $s + b \leq N-a+1$ then

While $v_a(s+b)$ is not exhausted

\[ c \leftarrow \text{Downread}(v_a(s+b)) \]

Push $s$ into $Q(c)$

Push $j$ into $Q(c)$  // also push $P$-origin into $Q$

EndWhile

EndIf

EndWhile

EndFor

// This gives us the maximal cliques of $R_a$ shifted so that all pairs of integers $(x, y)$ occurring on the same $Q$-stack are such that $x+b = y+b$. This means that, to a given $Q(c)$, there corresponds a maximal clique $C'$ of $R_a$, such that $Q(c)$ represents the set $(C')_{-b}$ mentioned in §I-4-2. The reason why $j$ (the label of $P$ stack where $s$ comes from) is also pushed into $Q$ will appear on the next step.

//Step-3  Construct $v_{a+b}$

\[ e_{a+b} \leftarrow 0 \]  // a clique counter

For $j=1$ to $e_a$

previous $\leftarrow 0$  // initialized to a dummy $P$ label

While $Q(j)$ is not empty

\[ c \leftarrow \text{Pop}(Q(j)) \]  // the $P$ label of $s$

\[ s \leftarrow \text{Pop}(Q(j)) \]

If $c \neq$ previous

\[ e_{a+b} \leftarrow e_{a+b} + 1 \]  // start a new clique of $R_{a+b}$

previous $\leftarrow c$

EndIf

Push $e_{a+b}$ into $v_{a+b}(s)$

EndWhile

EndFor

// $v_{a+b}$ is a representation of the set of cliques of $R_{a+b}$ obtained by performing the intersections mentioned in proposition 7a. Note that $e_{a+b}$ is incremented each time a new $Q(j)$ is considered or each time a new value of $c$ appears. It should be pointed out that, since only repeated patterns are searched for rather than all maximal cliques, the previous pseudocode should be modified so that only the labels corresponding to cliques whose size is greater or equal to two are actually pushed into $v_{a+b}$
Step-4 Remove non maximal cliques of $R_{a+b}$

First construct a vector $T = T(1), \ldots, T(e_{a+b})$ of $e_{a+b}$ stacks. Each $T$-stack will contains one of the clique of $R_{a+b}$ represented in $v_{a+b}$.

In addition, the status of each $T$-stack is flagged as ‘+’, ‘-‘ or ‘?’.

‘+‘: means ‘clique is maximal’; ‘-‘ means ‘clique is not maximal’

and ‘?’ means ‘status of clique is unknown’.

Built $T$ from $v_{a+b}$ as for $P$ in Step 1.

Mark all $T$-stacks as ‘?’

For $i = 1$ to $N - (a+b) + 1$

We will fix once for all the status of all $T$-stacks at position $i$.

While $v_{a+b}(i)$ is not empty

$c \leftarrow \text{Pop}(v_{a+b}(i))$

If $T(c)$ is not marked as ‘-‘

Check $T(c)$ against all remaining $T$-stacks

While $v_{a+b}(i)$ is not exhausted and $T(c)$ is not marked as ‘-‘

$d \leftarrow \text{DownRead}(v_{a+b}(i))$

If $T(d)$ is not marked as ‘-‘

If $T(c)$ is marked as ‘?’

If $T(c) \not\subseteq T(d)$ then mark $T(c)$ as ‘-‘

EndIf

EndIf

If $T(d)$ is marked as ‘?’

If $T(d) \not\subseteq T(c)$ then mark $T(d)$ as ‘-‘

EndIf

EndIf

EndWhile

Now the final status of $T(c)$ is known: if it is not included in any $T(d)$ at this position then it is maximal.

If $T(c)$ is not marked as ‘-‘ then mark it as ‘+‘

EndWhile

EndFor

Now the status of each $T$-stack is determined.

Rebuilt $V_{a+b}$ from $T$ by using only the $T$-stacks marked as ‘+‘

II-2-1 Complexity.

Now, we will consider the complexities of each separate step.
Step 1 At the beginning of step 1, we have at most $N.g^a$ indices in $v_a$ (proposition 4a). Thus, step 1 requires $O(N.g^a)$ operations.

Step 2 This step requires $O(N.g^{a+b})$ operations. The proof of this bound is more difficult. Let us consider, for the sake of simplicity, the cases of lemma 2 or the case of lemma 1 with $a = b$ (which is similar), since these are the cases mostly used in algorithms 1 and 2 above. Let us consider now a particular index $s$ popped from $P(i)$ and its associated a-length substring at position $s$. Each symbol of this substring cannot belong to more than $g$ maximal cliques of $R$ (by definition of $g$), thus, this a-length substring cannot belong to more than $g^a$ maximal cliques products and, hence, to more than $g^a$ maximal cliques of $R_a$ (this follows from proposition 3). Therefore, the index $s$ cannot appear more than $g^a$ times in P-stacks. Each time $s$ is popped from a P-stack, it will be pushed in the Q-stacks indicated in $v_b(s+a)$. Thus, let us consider, in the same manner, the b-length word at the position $s+a$. The same argument as above, shows that this word cannot belong to more than $g^b$ maximal cliques of $R_b$ and, hence, $s$ cannot be pushed into more than $g^b$ Q-stacks. Thus, a given index $s$ cannot appear more than $g^a$. $g^b = g^{a+b}$ times in all Q-stacks. Finally, there is at most $N.g^{a+b}$ elements in all the Q-stacks and the whole step has required $O(N.g^{a+b})$ operations. Analogous arguments with lemma 1 and $a \neq b$ would lead to $O(N.g^{2a})$ instead of $O(N.g^{a+b})$.

Step 3 This step requires the same number of operations as step 2 since at most $O(N.g^{a+b})$ indices have been pushed into Q-stacks, as previously stated. Note that, since a clique contains at least one element (actually 2 for repeated substrings) $e_{a+b} \leq N.g^{a+b}$.

Step 4 The construction of the T-stacks requires $O(N.g^{a+b})$ operations (this is the same operation as in step 1 but with now at most $N.g^{a+b}$ elements).

Then we perform all necessary inclusion tests by using $v_{a+b}$ and the T-stacks. First, remember that we have previously stated that each index $s$ cannot appear more than $g^{a+b}$ times in the Q-stacks and thus each $v_{a+b}(d)$ cannot contain more than $g^{a+b}$ cliques. Therefore, we have to perform at most $O(N.(g^{a+b})^2)$ inclusion tests.

Finally, we need to evaluate how many operations are required by one inclusion test. This depends on how the inclusion test (between two T-stacks representing two cliques) is conducted: there are basically two ways to do this. The easiest way is to compare each position indicated in the two T-stacks. Since the indices are ordered in each clique, this will take at most $O(N)$ operations. This first method would thus lead to
O(N^2.(g^{a+b})^2) operations for the overall step. There is, however, another way to perform the inclusion test. Remember that proposition 5 states that any inclusion test on cliques of R_k (here k = a+b) can be performed on cliques products. Thus, if all the cliques products are assumed to be known, one inclusion test would thus require O(k) operations (considering Card(A) as constant). The explicit representation of all the \( c_f \) cliques products can be conducted separately and requires O(k. N.g^k) (this follows from definition 2). Finally, this second method leads to O(N.k.g^{2k}) for the overall step.

Note, however, that for most practical cases (i.e., with non biased strings), the first method is linear with respect to N (This comes from the fact that when many cliques have to be considered their size is actually very small) and is much simpler to implement. Note also that for g=1 there is only 1 clique (class) in each \( v_{a+b}(d) \) and hence no inclusion test at all is performed. Hence the algorithm behaves, in this particular case, like the original KMR algorithm.

Finally, whole worst-case complexity of steps 1 to 4, i.e., of construction of maximal cliques of R_k , is of order O(N.k. g^{2k}) and thus the overall algorithm is of order O(N.k_{max}. g^{2k_{max}}).log(k_{max}) (k_{max} being either a chosen length (problem 1) or the maximum possible length (problem 2)).

Now we present some experimental results obtained on random sequences.

**II-3 Experiments.**

The following experiments concern the algorithm 1. The program is written in C and is run on a SUN Sparcstation (28 Mips). We present two sets of experiments, corresponding to two alphabets of respectively 10 (experiment 1) and 26 (experiment 2) latin letters. In both cases, we are searching for repeated patterns of length k = 4, using four different relations R between symbols. The first relation is the identity, i.e., has a degeneracy g = 1, in that case the algorithm is equivalent to the original KMR algorithm. The three other relations R are designed to have averaged degeneracies \( \bar{g} \) equal to 1.5, 2, and 2.5 with m= 10 (exp. 1) and m = 26 (exp. 2). This is obtained by using the following circular relation R : in experiments with \( \bar{g} = 2 \), the maximal cliques of R are \{a,b\}, \{b,c\}, ..., \{i,j\}, \{j,a\} (for exp. 1) and \{a,b\}, \{b,c\}, ..., \{y,z\}, \{z,a\} (for exp. 2); in experiment with \( \bar{g} = 1.5 \) half of the previous maximal cliques contains two elements as before and the other half contains only one element (e.g. \{a,b\}, \{b,c\}, ..., \{m,a\}, \{n\}, ..., \{x\}, \{y\}, \{z\}) for exp. 2); in experiment with \( \bar{g} = 2.5 \), half of the maximal cliques
contains two elements and the others contain three elements (e.g. \{a,b,c\}, \{b,c,d\}, \ldots, \{l,m,a\}, \{m,a,b\}, \{n,a\}, \ldots, \{y,z\}, \{z,n\}) for exp. 2). For each of these cases, we built a random string of size \(N\) increasing from 500 to 20000 by step \(\Delta N = 500\). For each \(N\), we record the CPU time \(t\) and the total number of indices \(n_S\) where 4-length repeated substrings where found. \(n_S\) is thus the actual number of indices to output in order to give the result. These values are plotted against \(N\) in figures 2a, b, c and d. Figures 2a and 2c correspond to the case \(m=10\). They clearly show that both \(t\) and \(n_S\) vary linearly with \(N\), at least for high values of \(N\). As mentioned in property 4a, \(n_S\) is bounded by \(N.g^k\). Note that the behavior is not linear for small values of \(N\). This can be seen more clearly in figures 2b and 2d (\(m=26\)). During the evaluation of the boundaries, we have made the assumption that, in the worst case, all cliques appear on the string. In fact, for small \(N\), only a few number of cliques did actually appear. This number depends upon \(N\) and the variation of \(n_S\) and \(t\) is no more linear, although still linearly bounded (see fig. 2b). For the same value of \(N\), more cliques are obtained for \(m=10\) than for \(m=26\) and thus the asymptotic linear behavior is observed sooner.

One may also notice in figures 2b and 2d that the variation of \(t\) does not follow strictly that of \(n_S\) (actually for \(m=26\), in that range of \(N\), the variation of \(t\) is linear for \(g=1\) and 1.5, and is less than \(N^2\) for \(g=2\) and \(g=2.5\)). This is shown in figure 3, where we plot the “relative time”, defined as the ratio of time to the number of indices, versus \(N\) for \(m=10\). We notice that, for a given \(g\), the relative time decreases with \(N\), as more and more cliques appear. Conversely, and for the same reason, at a given \(N\), the relative time decreases with \(g\). In many practical cases, \(R\) will be derived from a distance defined on the alphabet (e.g. \(xRy\) iff \(\text{distance}(x,y) \leq \square\), where \(\square\) is a given threshold). In that case large values of the averaged degeneracy \(\frac{\square}{g}\) are very unlikely.

### III Extensions to other problems.

Two extensions may be considered. The first one concerns repeated patterns in other structures such as arrays or trees. This has been considered by Karp et al. (1972) in the particular case where \(R\) is the identity. As an example, in order to find repeated \(k\times k\) patterns in an \(N\times N\) array, a variant of lemma 1 was presented in which the right side was the conjunction of four assertions instead of two. This leads, in our framework, to variants of propositions 6, 7a and 8a, in which four intersections have to be performed in order to obtain a set of cliques. As before, non maximal cliques must then be removed. In that problem, each clique is represented by a set of index pairs.
The second extension concerns the use of more general relations between objects. Let us consider a set \( X \) of objects and suppose a recurrence relation expressed with 3 symmetric and reflexive relations \( R_1, R_2 \) and \( R_3 \):

\[
x R_3 y \quad \quad \quad x R_1 y \quad \text{and} \quad x R_2 y
\]

Then, one can show, that the maximal cliques of the relation \( R_3 \) are obtained by computing all the intersections of pairs \((C1, C2)\) of maximal cliques of \( R_1 \) and \( R_2 \) and by removing non maximal cliques, as previously described. From this point of view, a variant of KMR-lemma 1 is obtained by considering the relations \( R_1, R_2 \) and \( R_3 \) defined as follows:

\[
R_1=R_a \quad : \quad x R_a y \quad i \quad i \in \{1, 2, \ldots, a\} \quad V_i(x) \quad R \quad V_i(y)
\]

\[
R_2=R_a^b \quad : \quad x R_a^b y \quad i \quad i \in \{1, 2, \ldots, a\} \quad V_{i+b}(x) \quad R \quad V_{i+b}(y)
\]

\[
R_3=R_{a+b} \quad : \quad x R_{a+b} y \quad i \quad i \in \{1, 2, \ldots, a+b\} \quad V_i(x) \quad R \quad V_i(y)
\]

This variant allows us to search for maximal cliques of \( R_k \) in a dictionary of short strings, \( x \) standing for one string in the dictionary, rather than inside a \( N \)-length string.

**Conclusion.**

In this paper we give a framework, some results, and some algorithms to address the problem of finding the maximal cliques of a relation \( R_k \) defined on labeled objects of size \( k \), and derived from a symmetric and reflexive relation \( R \) on the alphabet. The proposed algorithms will find repeated \( k \)-length flexible patterns in a string and are extensions of the KMR algorithms. The significant parameter is the averaged degeneracy \( \overline{g} \) of \( R \), i.e., the averaged number of maximal cliques of \( R \) to which a symbol belongs. The theoretical worst case time complexity is linear with the size \( N \) of the string. It should be pointed out that these algorithms apply to the particular case of an alphabet containing \( s \) symbols plus a “don’t care” symbol ‘#’ which matches any other symbol (in this case \( \overline{g} = 2s/s+1 \) if all symbols including ‘#’ have an equal probability). We have already used successfully these algorithms in the field of molecular biology, where \( R \) is derived from amino-acid substitution matrices and the strings represent protein sequences. As previously published in Landraud (1989), it is
relatively easy to add some constraints in the KMR algorithms in order, for instance, to look for patterns shared by several distinct sequences (e.g. Landraud (1989)) or to look for dyad symmetries in DNA sequences (e.g. Martinez 1983). Of course, the same is true for the algorithms proposed here. We also hope that this framework will be useful in other pattern recognition problems such as image analysis, speech recognition and time series analysis.

References.


Figure-1

Graph of a symmetric and reflexive relation. The edges due to reflexivity are omitted.

Figure-2

-a: Averaged number $n_s$ of repeated 4-length substrings plotted against the length $N$ of random sequences. $m=10$ and $g$ varies from 1 to 2.5

-b: Averaged number $n_s$ of repeated 4-length substrings plotted against the length $N$ of random sequences. $m=26$ and $g$ varies from 1 to 2.5

-c: Averaged CPU time $t$ plotted against the length $N$ of random sequences. $m=10$ and $g$ varies from 1 to 2.5

-d: Averaged CPU time $t$ plotted against the length $N$ of random sequences. $m=26$ and $g$ varies from 1 to 2.5

Figure-3

Averaged $t/n_s$ ratio plotted against the length $N$ of random sequences, $k=4$, $m=26$ and $g$ varies from 1 to 2.5.