Theoretical Computer Science 460 (2012) 94–103

Contents lists available at SciVerse ScienceDirect

Theoretical Computer Science

journal homepage: www.elsevier.com/locate/tcs

Note

# Extracting string motif bases for quorum higher than two

Simona E. Rombo *

*Institute for High Performance Computing and Networking (ICAR), National Research Council of Italy (CNR), Rende (CS), Italy*
*DEIS, Università della Calabria, Rende (CS), Italy*

ARTICLE INFO

*Article history:*
Received 22 September 2011
Received in revised form 29 March 2012
Accepted 18 June 2012
Communicated by M. Crochemore

ABSTRACT

Bases of generators of motifs consisting of strings in which some positions can be occupied by a don't care provide a useful conceptual tool for their description and a way to reduce the time and space involved in the discovery process.

In the last few years, a few algorithms have been proposed for the extraction of a basis, building in large part on combinatorial properties of strings and their autocorrelations. Currently, the most efficient techniques for binary alphabets and quorum $q = 2$ require time quadratic in the length of the host string.

The present paper explores properties of motif bases for quorum $q \geq 2$, both with binary and general alphabets, by also showing that important results holding for quorum $q = 2$ cannot be extended to this, more general, case. Furthermore, the extraction of motifs in which a bound is set on the maximum allowed number of don't cares is addressed, and suitable algorithms are proposed whose computational complexity depends on the fixed bound.



## 1. Introduction

The extraction of motif patterns made of intermixed solid and *don't care* characters has been shown to play a relevant role in molecular biology, data compression and other domains (cf., [6,1,13]). Usually, string patterns are considered to be *interesting* if they occur a number of times at least equal to the *quorum*, that is, a given threshold fixed a priori.

One of the most challenging problems in such a discovery process is that the number of candidate motifs grows exponentially with the size of the input string. Pioneered by Parida et al. [12,14], the notion of *string motif basis*, based on maximal saturation and irredundancy, has been introduced in order to alleviate such a growth. Motif bases are special classes of motifs with high informative content. They provide a useful conceptual tool for the description and generation of all the other motifs occurring in the input string $s$ w.r.t. a given quorum, yet growing linearly with the size of $s$ [2,10,16,18,19]. Most techniques proposed to extract motif bases exploit combinatorial properties of strings and their autocorrelations. The most efficient algorithms work for alphabets of finite size and require time at least quadratic in the length of the host string [4,5]. Such approaches apply only for quorum $q = 2$ and they do not allow for any control about the number of don't cares in the extracted motifs. Furthermore, their computational complexity depends on the size of the alphabet characterizing the input string, that is impractical for alphabets with large size (i.e., in the case of discretized time series or digital images).

In many applications such as, for example, time series and computational biology, most interesting motifs to be extracted are those featuring high frequencies [15,6]. Unfortunately, motif bases with quorum $q' > 2$ cannot be obtained from a basis $\mathcal{B}$ computed for quorum $q = 2$ on the same input string, by simply discarding from $\mathcal{B}$ those motifs occurring less than $q'$ times. Indeed, this would alter the strict dependency among the irredundant motifs composing the basis.

* Correspondence to: Institute for High Performance Computing and Networking (ICAR), National Research Council of Italy (CNR), Rende (CS), Italy. Tel.: +39 0984 494774.
*E-mail address:* srombo@deis.unical.it.

0304-3975/$ – see front matter © 2012 Elsevier B.V. All rights reserved.
doi:10.1016/j.tcs.2012.06.021

In this work, we study the problem of extracting string motif basis when the quorum is greater than two. In particular, we prove that results leading to optimal algorithms for quorum $q = 2$ cannot be extended for quorum $q' > 2$, and we also present several properties holding for any quorum. Previous work showed that the number of motifs in the basis increases with the quorum value (sse, e.g., [2,10]). This is because the number of don't cares in the motifs of the basis is, in general, larger for higher quorums. However, motifs with don't cares are characterized by some indeterminacy that increases when the number of don't cares is larger, with the chance of losing important informative content. To avoid such a situation, we propose suitable algorithms where a bound can be assumed on the total number of don't cares admitted in any element of the basis, by following the directions addressed for other classes of motifs [8,9]. A basis obtained this way includes only those maximal motifs containing a limited number of don't cares and that are irredundant each other. The algorithms we propose present the important peculiarity that their computational complexities depend also on the maximum number of don't cares admitted in each motif of the basis, and a more efficient extraction of motifs with low indeterminacy can be obtained. Finally and interestingly, such algorithms do not depend on the size of the alphabet of the input string.

## 2. Preliminaries

Let $s$ be a string of $n$ characters over a finite alphabet $\Sigma$. We denote by $s[j]$ the *j-th* character of $s$, if 1 is the first position of $s$. We also use $pref_i(x)$ and $suf_i(x)$ to denote, respectively, the *i-th* prefix (i.e., the first $i$ characters) and the *i-th* suffix (i.e., the one that starts at position $i$) of the string $x$. We exploit the compact notation $pref_i$ and $suf_i$ when referring to the string $s$. In addition to the *solid* characters from $\Sigma$, we also admit a special *don't care* character denoted by "$\circ$" and matching any other character in $\Sigma^+ = \Sigma \cup \{\circ\}$. For characters $\sigma_1$ and $\sigma_2$, the relation $\sigma_1 \preceq \sigma_2$ holds when $\sigma_1$ is a don't care or $\sigma_1 = \sigma_2$.

The following definitions are recaptured to deal with the problem of string motif extraction. In some cases, they match or extend the corresponding ones in, e.g., [2,4,10,19].

The operator $\oplus$ is defined on any pair of characters $\sigma_1$ and $\sigma_2$ in $\Sigma^+$ as:

$$\sigma_1 \oplus \sigma_2 = \begin{cases} \sigma_1, & \text{if } \sigma_1 = \sigma_2 \\ \circ, & \text{if } \sigma_1 \neq \sigma_2. \end{cases}$$

**Definition 1** (*Consensus, Meet*)**.** Let $m_1$ and $m_2$ be two strings on $\Sigma^+$. The *consensus* between $m_1$ and $m_2$ is the string $m = m_1 \oplus m_2$ such that $m[i] = m_1[i] \oplus m_2[i]$, for $1 \leq i \leq \min\{|m_1|, |m_2|\}$. Deleting all leading and trailing don't cares from $m_1 \oplus m_2$ yields a (possibly empty) string on $\Sigma^+$ that is the *meet* between $m_1$ and $m_2$, denoted by $[m_1 \oplus m_2]$.

The operator $\oplus$, as well as the induced notions of consensus and meet, extends naturally to more than two characters or strings.

The following definitions make implicit reference to a subject string $s$ of $n$ characters on $\Sigma$.

**Definition 2** (*Occurrence*)**.** Let $m$ be a string on $\Sigma^+$ such that $|m| \leq n$. A position $h$ of $s$ ($h \leq n - |m|$) is an *occurrence* of $m$ if and only if $m[i] \preceq s[i + h]$, $1 \leq i \leq |m|$. In this case, we call *occurrence* of $m$ also $pref_{|m|}(suf_h)$.

We are interested in the discovery of patterns that occur in $s$ at least $q$ times, where $q \leq n$ is a positive integer called the *quorum*.

**Definition 3** (*q-Motif*)**.** A *q-motif* of $s$ is a pair $(m, \mathcal{L}_m)$, where: *(i)* $m$ is a string on $\Sigma^+$ such that $|m| \geq 1$ and $m[1]$ and $m[|m|]$ are solid characters, and *(ii)* $\mathcal{L}_m = \{l_1, l_2, \ldots, l_p\}$, with $p \geq q$, is the exhaustive list of all the occurrences of $m$ in $s$.

When the value of the quorum is clear from the context, we refer to $q$-motifs simply as *motifs*.

**Definition 4** (*q-Autocorrelation*)**.** Let $I = \{i_1, i_2, \ldots, i_{q-1}\}$ be a set of positions of $s$ ($q \leq n$). The meet $[s \oplus suf_{i_1} \oplus suf_{i_2} \oplus \cdots \oplus suf_{i_{q-1}}]$ is a *q-autocorrelation* of $s$.

Note that a $q$-autocorrelation $m$, if not empty, is necessarily a *q-motif*. Indeed, the first condition of Definition 3 is automatically satisfied by the definition of meet (Definition 1). The second condition is satisfied as well since, if $m$ is not empty, there are at least $q$ positions in $s$ where it occurs, corresponding to the $q$ suffixes (one of them possibly coinciding with $s$) whose meet generates $m$. Furthermore, it is worth pointing out that a $q$-autocorrelation does not necessarily occur at the first position of $s$, as pointed out by the following example.

**Example 1.** Let $s = aaabcdbcdbccdbdd$ be the input string and $q = 3$ be the quorum. Then, to obtain the 3-autocorrelation generated by $s$, $s_4$ and $s_{11}$, three leading don't cares need to be deleted from the corresponding consensus, and the resulting meet is $m = b \circ d$ with $\mathcal{L}_m = \{4, 7, 14\}$.

**Definition 5** (*Submotif*)**.** Given two motifs $m_1$ and $m_2$ with $|m_1| \leq |m_2|$, $m_1 \preceq m_2$ holds if and only if there exists an integer $0 \leq h \leq (|m_2| - |m_1|)$ such that $m_1[i + h] \preceq m_2[i]$, $1 \leq i \leq |m_1|$. In this case, $m_1$ is a *submotif* of $m_2$, and $m_2$ *implies* or *extends* $m_1$.

**Definition 6** (*Maximal Motif*)**.** A motif $m$ is *maximal* if and only if there is no other motif $m'$ such that $m$ is a submotif of $m'$ and $|\mathcal{L}_m| = |\mathcal{L}_{m'}|$.

**Definition 7** (*q-Motif Basis*)**.** A maximal motif $m$ is *redundant* if and only if there exist $r$ motifs in $s$ such that $\mathcal{L}_m = \bigcup_{i=1}^{r}(\mathcal{L}_{m_i} + h_i)$, where: *(i)* $m \preceq m_i$, for each $m_i$ $(1 \leq i \leq r)$; *(ii)* each $h_i$ is a positive integer $(1 \leq i \leq r)$; *(iii)* $\mathcal{L}_{m_i} + h_i = \{l_1 + h_i, l_2 + h_i, \ldots, l_p + h_i\}$.
If a motif is not redundant, it is called *irredundant*. The set of all the irredundant $q$-motifs of $s$ is called $q$-*basis* of $s$ and it is unique (as proved in [3]).

## 3. General properties

In the following, we consider in our analysis only non-trivial motifs $m$ such that $|m| > 1$. We use $k$ to denote the maximum number of present/allowed don't cares in a motif, and denote by $d(m)$ the number of don't cares in a motif $m$. Furthermore, let $\mathcal{P}$ and $pos(m)$ be the list of the don't care positions and the sum of the numerical values of positions of don't cares in a motif $m$, respectively. As an example, if $m = a\circ\circ bcdaaa\circ add\circ b$, then $\mathcal{P} = \{2, 3, 10, 14\}$ and $pos(m) = 2+3+10+14 = 29$. Let $I = \{i_1, i_2, \ldots, i_h\}$ be a set of positions of $s$ $(h \leq n)$. The meet $[suf_{i_1} \oplus suf_{i_2} \oplus \cdots \oplus suf_{i_h}]$ is denoted by $[\oplus suf_I]$. In general, for a motif $m$ that is the meet of $h$ suffixes there may be more than one way to choose $h$ suffixes that produce $m$ as their meet. Here we make the convention that we always choose $h$ suffixes whose consensus has no leading don't cares. Thus, for example, for $s = abccaacabccabc$, $I = \{2, 5, 9\}$ and $m = [suf_2 \oplus suf_5 \oplus suf_9] = ca \circ c$, we use $suf_4$, $suf_7$ and $suf_{11}$ to indicate the generation of $m$ and thus write $m = [suf_4 \oplus suf_7 \oplus suf_{11}]$. Note that, by referring to such a convention, a $q$-autocorrelation can always be expressed as the meet among $q$ suffixes of $s$, where the first of such suffixes coincides with the whole $s$ only if no leading don't cares have been deleted from the corresponding consensus. The following results hold.

**Lemma 1.** *Let $m$ be a maximal $q$-motif such that $\mathcal{L}_m = \{i_1, \ldots, i_h\}$ $(h \geq q)$. Then $m = [suf_{i_1} \oplus \cdots \oplus suf_{i_h}]$.*

**Proof.** Suppose that there exists a motif $m' = [\oplus suf_I]$ s.t. $I = \{i_1, i_2, \ldots, i_h\}$ with $m \neq m'$. Then, one between $m \preceq m'$ and $m' \preceq m$ necessarily holds. If $m \preceq m'$, the condition of maximality for $m$ would be contradicted, since $|\mathcal{L}_m| = |\mathcal{L}_{m'}|$. If $m' \preceq m$, then there exists at least a position $j$ of the consensus among $suf_{i_1}, \ldots, suf_{i_h}$ that is a don't care, while $m[j]$ is a solid symbol. Let $suf_{i_k}$ and $suf_{i_l}$ be two suffixes s.t. $i_k, i_l \in I$ and $suf_{i_k}[j] \neq suf_{i_l}[j]$. This would lead, again, to a contradiction, since $m$ occurs at both $i_k$ and $i_l$, and $m[j]$ is solid. □

**Theorem 1.** *Each $q$-autocorrelation of $s$ is a maximal $q$-motif.*

**Proof.** Let $m = [suf_{i_1} \oplus \cdots \oplus suf_{i_q}]$ be a motif in $\mathcal{A}$ and suppose that $m$ is not maximal. Then, according to Definition 6, there exists a motif $m'$ such that $m \preceq m'$ and $|\mathcal{L}_m| = |\mathcal{L}_{m'}|$. Suppose that $m'$ is maximal (otherwise consider the maximal motif extending both $m$ and $m'$). Since $m \preceq m'$, each occurrence $i_j$ of $m$ is covered by an occurrence $i'_j$ of $m'$ s.t. $i'_j \leq i_j$. Furthermore, by Lemma 1, $m' = [suf_{i'_1} \oplus \cdots \oplus suf_{i'_h}]$, where $\mathcal{L}_{m'} = \{i'_1, \ldots, i'_h\}$ and $h \geq q$. Since $m$ is the meet among $s$ and $q-1$ of its suffixes, none of its occurrences at $i_1, \ldots, i_q$ can be covered by any occurrence of $m'$, thus $i_1 = i'_1, \ldots, i_q = i'_q$. Since the meet $[suf_{i_1} \oplus \cdots \oplus suf_{i_q}]$ cannot be a submotif of $[suf_{i_1} \oplus \cdots \oplus suf_{i_q} \oplus suf_{i'_{q+1}} \oplus \cdots \oplus suf_{i'_h}]$, then necessarily $m \equiv m'$. □

**Lemma 2.** *Let $m$ be a maximal $q$-motif such that $m \notin \mathcal{A}$, if $\mathcal{A}$ is the set of $q$-autocorrelations of $s$. Then $m$ is a $q'$-autocorrelation with $q' > q$.*

**Proof.** If $m \notin \mathcal{A}$ and $m$ is maximal, then there exists at least one motif $m_i \in \mathcal{A}$ such that $m \preceq m_i$ but $|\mathcal{L}_m| > |\mathcal{L}_{m_i}|$. This means that $|\mathcal{L}_m| > q$ thus, by Lemma 1 and by the definition of maximality, the claim is proved. □

**Lemma 3.** *Let $I = \{i_1, i_2, \ldots, i_h\}$ be a set of positions of $s$ $(h \leq n)$, and let $d(m)$ be the number of don't cares in the meet $m = [\oplus suf_I]$. Consider the $h$ meets that can be formed by taking suffixes in the set $\{suf_{i_1}, suf_{i_2}, \ldots, suf_{i_h}\}$ $h-1$ at a time. Let $m_j$ be the $|m|-th$ prefix of the $j-th$ such meet $(1 \leq j \leq h)$. Then, the following hold:*

$$\sum_{j=1}^{h} d(m_j) \leq h \cdot d(m) \tag{1}$$

$$m \preceq m_j, \quad j = 1, \ldots, h. \tag{2}$$

**Proof.** For any $i_j \in I$, consider the set of positions $I_j = I/\{i_j\}$, and let $m_j$ be the $|m|$-th prefix of the meet $[\oplus suf_{I_j}]$. Then, $m$ may be viewed as:

$$m = [m_j \oplus suf_{i_j}].$$

If the $|m|$-th prefix of $suf_{i_j}$ is an occurrence of $m_j$, then $d(m) = d(m_j)$. Otherwise, there is at least one position where $m_j$ and $pref_{|m|}(suf_{i_j})$ have a different solid symbol. Such a mismatch would imply that:

$$m \preceq m_j \tag{3}$$

$$d(m_j) \leq d(m). \tag{4}$$

Adding up both sides of the inequality (4) over all the $h$ meets yields the claim. □

Agreeing with the results already obtained for quorum $q = 2$ [3–5], the following theorem holds.

**Theorem 2.** *Let $\mathcal{A}$ be the set of $q$-autocorrelations and $\mathcal{B}$ be the $q$-basis of $s$. Then $\mathcal{B} \subseteq \mathcal{A}$.*

**Proof.** Let $m$ be a motif in $\mathcal{B}$ and suppose that $m$ is not a $q$-autocorrelation. By Lemma 2, $m$ is a $q'$-autocorrelation with $q' > q$. Let $\mathcal{M}$ the set of meets among the suffixes generating $m$, taken $q$ at a time. By Lemma 3 $m \preceq m_i$, for each $m_i \in \mathcal{M}$. Furthermore, $\mathcal{L}_m = \cup_i \mathcal{L}_{m_i}$ $(m_i \in \mathcal{M})$. But this would lead to a contradiction since $m$ was supposed to be irredundant. □

A $q$-basis extraction paradigm for all alphabet sizes unfolds along the following three main steps: *(1)* extract the set $\mathcal{A}$ of all the (non-empty) distinct $q$-autocorrelations of the input string; *(2)* compute the occurrence list of each $q$-autocorrelation; *(3)* discard from $\mathcal{A}$ the redundant motifs, obtaining the $q$-basis in $\mathcal{A}$.

When the quorum is $q = 2$, it is not difficult to implement Steps 1 and 3 in $O(n^2)$ time (cf. [2,16,18]). For a generic quorum $q$, each $q$-autocorrelation is the meet among $s$ and $q-1$ of its suffixes. Thus there are $\binom{n-1}{q-1}$ meets to be computed, each computing requiring $O(n)$ time. The resulting number of autocorrelations is bounded by $O\left(\frac{n^{q-1}}{q!}\right)$, and Step 1 can be solved in $O\left(\frac{n^q}{q!}\right)$ time. For what concerns Step 3, consider the collection of the location lists of all the motifs in $\mathcal{A}$. For each of such list $\mathcal{L}_m$, if $\mathcal{L}_m = \mathcal{L}_{m_1} \bigcup \mathcal{L}_{m_2} \cdots \bigcup \mathcal{L}_{m_h}$ up to some offsets, with $m_1, \ldots, m_h \in \mathcal{A}$ and $m_i \neq m$ $(i = 1, 2, \ldots, h)$, then $m$ is redundant. Otherwise, $m$ is irredundant and can be added to the final output basis. Such a test can be afforded in $O(n)$ time for each of the $O\left(\frac{n^{q-1}}{q!}\right)$ lists, by testing whether all occurrences in $\mathcal{L}_m$ falls into the "footprints" of some occurrence of a subgroup of the other motifs (cf., e.g., [10]). Thus, also Step 3 can be performed in $O\left(\frac{n^q}{q!}\right)$ time.

The bottleneck that influences the overall cost is Step 2. The occurrence list of a generic autocorrelation $m$ can be obtained by checking, for each position $i$ of $s$, if $i$ is an occurrence of $m$. Thus Step 2 can be performed in time $O(n|\mathcal{A}|\mu)$, where $|\mathcal{A}|$ is the number of autocorrelations and $\mu$ is the overall time necessary for the check.

In [10,17] algorithms have been proposed for quorum $q = 2$ such that the overall time bounds are $O(n^2 \log n \log |\Sigma|)$ and $O(|\Sigma| n^2 \log^2 n \log n \log\log n)$, respectively. However, the algorithms in [10,17] exploit the landmark string searching algorithm by Fischer and Paterson [7], based on the FFT, which is indirect and admittedly impractical in many cases. With binary alphabets, an implementation of Step 2 in $O(n^2)$ time is presented in [4,5], leading to an optimal algorithm since $\Theta(n^2)$ space is necessary to store all the autocorrelations. Unfortunately, such an approach cannot be adopted for quorum $q > 2$, as will be explained later in this paper. Furthermore, the computational complexity of all the approaches [4,5,10,17] depends on the size of the alphabet of the input string, and this can be ineffective for some applications where it can reach high values, such as discretized time series or digital images.

When $\Sigma$ is a binary alphabet, the two following lemmas hold.

**Lemma 4.** *Let $i$, $j$ and $h$ be three positions of $s$ and consider the meets $m = [suf_i \oplus suf_j \oplus suf_h]$, $m_{ij} = pref_{|m|}([suf_i \oplus suf_j])$, $m_{jh} = pref_{|m|}([suf_j \oplus suf_h])$ and $m_{ih} = pref_{|m|}([suf_i \oplus suf_h])$. Then,*

$$d(m_{ij}) + d(m_{jh}) + d(m_{ih}) = 2d(m). \tag{5}$$

**Proof.** Let $p$ be the position of a don't care in $m_{ij}$. Since the alphabet is binary, then the symbol in position $p$ of $pref_{|m|}(suf_h)$ necessarily agrees with one of the two symbols in position $p$ of $pref_{|m|}(suf_i)$ and $pref_{|m|}(suf_j)$. Thus, for each don't care in $m_{ij}$, there is a don't care in the same position in one between $m_{jh}$ and $m_{ih}$, and a solid symbol in the other one. Note that $m$ may also be obtained as the meet among $m_{ij}$, $m_{jh}$ and $m_{ih}$. Thus, each don't care of $m$ in position $p$ corresponds to a don't care in the same position in two of the generating prefixes, and to a solid symbol in the remaining one. □

Lemma 4, that can also be viewed as a specialization of (1) in Lemma 3, supports an alternative proof of a lemma at the core of the basis extraction approaches for quorum $q = 2$ and binary alphabets presented in [4,5], as shown below.

**Lemma 5.** *Let $i$, $j$ and $h$ be three positions of $s$ and consider the meets $m_{ij} = [suf_i \oplus suf_j]$, $m_{ih} = pref_{|m_{ij}|}([suf_i \oplus suf_h])$ and $m_{jh} = pref_{|m_{ij}|}([suf_j \oplus suf_h])$. Then: $h \in \mathcal{L}_{m_{ij}} \Leftrightarrow d(m_{ij}) = d(m_{ih}) + d(m_{jh})$.*

**Proof.** If $pref_{|m_{ij}|}(suf_h)$ is an occurrence of $m_{ij}$, then $m_{ij} = [suf_i \oplus suf_j \oplus suf_h]$, thus $d(m_{ij}) = d(m)$ and, from Lemma 4:

$$d(m_{ij}) + d(m_{ih}) + d(m_{jh}) = 2d(m_{ij}) \Rightarrow d(m_{ih}) + d(m_{jh}) = d(m_{ij}).$$

On the other hand, if $d(m_{ij}) = d(m_{ih}) + d(m_{jh})$ holds, then substituting in (5) the term $d(m_{ih}) + d(m_{jh})$ with $d(m_{ij})$, we obtain that $d(m_{ij}) = d(m)$. This means that the two meets $m_{ij} = [suf_i \oplus suf_j]$ and $m = [suf_i \oplus suf_j \oplus suf_h]$ are equal, and then $h$ is an occurrence of $m_{ij}$. □

Assume that the number of don't cares in every prefix of every meet between two suffixes of $s$ has been computed and stored. Then, given any such meet, Lemma 5 may be used to check whether or not it occurs at any assigned position of $s$, in constant time. This gives a handle in the extraction of 2-motif bases, leading to an optimal algorithm for binary alphabets and $q = 2$ [5]. For general alphabets and any quorum, the following two lemmas hold.

| | | | | | |
|---|---|---|---|---|---|
| $suf_i$ | a | b | b | a | b |
| $suf_j$ | a | b | a | a | b |
| $suf_h$ | a | a | b | b | b |
| $m_{ijh}$ | a | · | · | · | b |

| | | | | | |
|---|---|---|---|---|---|
| $suf_i$ | a | b | b | a | b |
| $suf_h$ | a | a | b | b | b |
| $suf_g$ | a | b | b | b | b |
| $m_{ihg}$ | a | · | b | · | b |

| | | | | | |
|---|---|---|---|---|---|
| $suf_i$ | a | b | b | a | b |
| $suf_j$ | a | b | a | a | b |
| $suf_g$ | a | b | b | b | b |
| $m_{ijg}$ | a | b | · | · | b |

| | | | | | |
|---|---|---|---|---|---|
| $suf_j$ | a | b | a | a | b |
| $suf_h$ | a | a | b | b | b |
| $suf_g$ | a | b | b | b | b |
| $m_{jhg}$ | a | · | · | · | b |

**Fig. 1.** Lemma 5 cannot be extended to $q = 3$.

**Lemma 6.** *Let $I = \{i_1, i_2, \ldots, i_h\}$ be a set of positions of $s$ ($h \leq n$) such that $m = [\oplus suf_I]$. Consider the set $\mathcal{M}$ of $|m|$-th prefixes of the $h-1$ meets that can be formed by the suffixes in the set $\{suf_{i_1}, suf_{i_2}, \ldots, suf_{i_h}\}$, taken $h-1$ at a time. Let $g$ be a position of $s$ such that $g \neq i_1, i_2, \ldots, i_h$. If $pref_{|m|}(suf_g)$ is an occurrence of a motif $m_j \in \mathcal{M}$, then it is also an occurrence of $m$.*

**Proof.** Immediate from the fact that, by Equation (3) in Lemma 3, $m \preceq m_j$, for each $m_j \in \mathcal{M}$. □

**Lemma 7.** *Let $I = \{i_1, i_2, \ldots, i_h\}$ be a set of positions of $s$ ($h \leq n$) such that $m = [\oplus suf_I]$. Let $g$ be a position of $s$ such that $g \neq i_1, i_2, \ldots, i_h$. Let $\mathcal{M}'$ be the set of $|m|$-th prefixes of the $h-1$ meets among $suf_g$ and the suffixes in $\{suf_{i_1}, suf_{i_2}, \ldots, suf_{i_h}\}$, taken $h-1$ at a time. Then, $pref_{|m|}(suf_g)$* is not *an occurrence of $m$ if at least one of the following two conditions is satisfied:*

1. *$d(m_f) > d(m)$, for each $m_f \in \mathcal{M}'$,*
2. *$d(m_f) = d(m)$ and $pos(m_f) \neq pos(m)$, for each $m_f \in \mathcal{M}'$.*

**Proof.** Condition 1 implies that $pref_{|m|}(suf_g)$ cannot be an occurrence of $m$, since there is a don't care in correspondence to the same solid character occurring in the same position of the $|m|$-th prefixes of $suf_{i_1}, \ldots, suf_{i_h}$. Similarly, this happens also when the second condition holds, since if the number of don't cares is the same but the sum of positions is not, then there is some don't care in the wrong position, that is, in correspondence to some solid symbol of $m$. □

Lemmas 6 and 7 enable us to test whether or not the prefix of a suffix is an occurrence of a given maximal $q$-motif under general alphabets, although in particular cases. This paves the way for identifying suitable cut-off rules allowing to make more efficient the selection of those positions of $s$ candidate to be occurrences of a specific autocorrelation, as will be discussed in Section 5.

## 4. Methods

Step 2 of the general paradigm for basis extraction can be performed as explained below. Let $\mathcal{A}$ be the set of $q$-autocorrelations extracted from $s$. Then the list of occurrences of the elements in $\mathcal{A}$ can be obtained by checking, for each autocorrelation $m$ in $\mathcal{A}$ and for each position $i$ in $s$, if $i$ is an occurrence of $m$.

In general, the number of autocorrelations is bounded by $O(\frac{n^{q-1}}{q!})$ and the number of positions in $s$ to consider for each $m$ in $\mathcal{A}$ is $n - |m|$. However, we will discuss later in this paper that the resulting $O(\frac{n^q}{q!})$ bound can be improved by suitable artifices.

What influences most the overall cost of Step 2 is checking if a position of $s$ is an occurrence of a given autocorrelation. As already recalled in the previous section, for binary alphabets and $q = 2$ Lemma 5 allows to perform this task in constant time. We seek here balancing properties similar to that of Lemma 5 for a quorum greater than 2. The following example shows that, even when the alphabet is binary, that lemma cannot be brutally extended even to $q = 3$.

**Example 2.** As displayed in Fig. 1, although $g$ is an occurrence of $m_{ijh}$, the sum of the number of don't cares in $m_{ijg}$, $m_{ihg}$ and $m_{jhg}$ is not equal to the number of don't cares in $m_{ijh}$.

This opens the quest for possible alternative properties relating maximal 3-motifs to their occurrences. One possibility consists in exploiting the inequality (1) of Lemma 3 in order to relate the number of don't cares in the meets of triplets of suffixes. For instance, let $m_1 = [suf_i \oplus suf_j \oplus suf_h]$, $m_2 = [suf_i \oplus suf_j \oplus suf_g]$, $m_3 = [suf_i \oplus suf_h \oplus suf_g]$, $m_4 = [suf_j \oplus suf_h \oplus suf_g]$ and $m' = [suf_i \oplus suf_j \oplus suf_h \oplus suf_g]$. If $pref_{|m_1|}(suf_g)$ of $s$ is an occurrence of $m_1$, then $d(m_1) = d(m')$ and the inequality (1) becomes:

$$d(m_2) + d(m_3) + d(m_4) \leq 3d(m_1). \tag{6}$$

Unfortunately, this inequality can be satisfied also when $g$ is not an occurrence of $m_1$, as shown by the following example.

**Example 3.** Let $suf_i = ababcdab$, $suf_j = abaccadb$ and $suf_h = aaaccddb$. Then, $m_1 = a \circ a \circ c \circ \circ b$. Let $suf_g = abacbddb$. Then, $m_2 = aba \circ \circ \circ \circ b$, $m_3 = a \circ a \circ \circ d \circ b$ and $m_4 = a \circ ac \circ \circ db$. Although $d(m_2) + d(m_3) + d(m_4) = 11$, which is less than $3d(m_1) = 12$, $g$ is not an occurrence of $m_1$.

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| $suf_i$ | a | b | a | b | b | a | a | b | b |
| $suf_j$ | a | a | b | b | b | a | b | a | b |
| $suf_h$ | a | b | a | a | b | a | b | a | b |
| $m$ | a | · | · | · | b | a | · | · | b |
| $suf_g$ | a | a | b | a | b | a | b | b | b |
| $s_1$ | a | b | a | b | b | a | a | b | b |
| $\oplus$ | a | · | · | · | b | a | · | b | b |
| $suf_{g'}$ | a | b | a | b | b | b | b | a | b |
| $s_1$ | a | b | a | b | b | a | a | b | b |
| $\oplus$ | a | b | a | b | b | · | · | · | b |

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| $s_1$ | a | b | a | b | b | a | a | b | b |
| $s_2$ | a | a | b | a | b | a | b | a | b |
| $m$ | a | · | · | · | b | a | · | · | b |
| $suf_g$ | a | a | b | a | b | a | b | b | b |
| $s_2$ | a | a | b | a | b | a | b | a | b |
| $\oplus$ | a | a | b | a | b | a | b | · | b |
| $suf_{g'}$ | a | b | a | b | b | b | b | a | b |
| $s_2$ | a | a | b | a | b | a | b | a | b |
| $\oplus$ | a | · | · | · | b | · | b | a | b |

**Fig. 2.** $g$ is an occurrence of $m$, whereas $g'$ is not.

In the following, we describe suitable strategies to solve the aforementioned problem first for binary and then for general alphabets.

### 4.1. Binary alphabets

**Lemma 8.** *Let $m = [suf_i \oplus suf_j \oplus suf_h]$, and $s_1$ and $s_2$ be any two strings of size $|m|$ on $\Sigma = \{a, b\}$ obtained by arbitrarily replacing each don't care of $m$ by an $a$ in $s_1$ and a $b$ in $s_2$, or vice versa. Then $m = [s_1 \oplus s_2]$.*

**Proof.** Immediate. □

**Lemma 9.** *Let $m = [suf_i \oplus suf_j \oplus suf_h]$, and $s_1$ and $s_2$ be two strings of size $|m|$ on $\Sigma = \{a, b\}$ obtained from $m$ as in Lemma 8. Let $m' = pref_{|m|}([suf_g \oplus s_1])$ and $m'' = pref_{|m|}([suf_g \oplus s_2])$. Then $g \in \mathcal{L}_m \Leftrightarrow d(m) = d(m') + d(m'')$.*

**Proof.** We first prove that if $g \in \mathcal{L}_m$, then $d(m) = d(m') + d(m'')$. From Lemma 8, $m = [s_1 \oplus s_2]$. If $g \in \mathcal{L}_m$, then:

1. each solid symbol of $m$ appears also at the corresponding position of $pref_{|m|}(suf_g)$;
2. in correspondence with every don't care of $m$, $pref_{|m|}(suf_g)$ agrees with only one between $s_1$ and $s_2$.

The first condition implies that both $m'$ and $m''$ have solid symbols in correspondence with solid symbols of $m$. From the second condition, it follows that for each don't care of $m$ there is a don't care in one of $m'$ and $m''$, and a solid symbol in the other. This establishes the direct implication.

Next, we prove by contradiction that if $d(m) = d(m') + d(m'')$ then $g \in \mathcal{L}_m$. In fact, assuming that under this condition $g$ is not in $\mathcal{L}_m$, then there must exist some position $i$ at which $m$ has a solid character that differs from the one occupying the same position in $pref_{|m|}(suf_g)$. This implies that both $m'$ and $m''$ have a don't care in position $i$, agreeing with the fact that $g$ is not an occurrence of $m$. Now, consider the positions of $m$ corresponding to don't cares. Let $i'$ be one such position. Two cases are possible:

1. $m[i'] = s_1[i']$. In this case, $m'$ has a solid symbol and $m''$ has a don't care at $i'$;
2. $m[i'] = s_2[i']$. Hence, $m''$ has a solid symbol and $m'$ has a don't care at $i'$.

Then, in correspondence with any don't care of $m$, only one between of $m'$ and $m''$ may have a don't care. This means that the number of don't cares of $m$ is covered by $d(m') + d(m'')$; but there are also don't cares at position $i$ of both $m'$ and $m''$, whence $d(m') + d(m'') > d(m)$, a contradiction. □

**Example 4.** Fig. 2 shows the balance of don't cares expressed in Lemma 9. Specifically, the figure displays two positions $g$ and $g'$ of the input string $s$ such that $g$ is an occurrence of $m$ and $g'$ is not.

**Corollary 1.** *Let $m = [suf_i \oplus suf_j \oplus suf_h]$, let $s_1$ be the string of size $|m|$ on $\Sigma = \{a, b\}$ obtained by replacing each don't care in position $l$ of $m$ by an $a$ if the symbol in position $l$ of $suf_i$ is a $b$, and by a $b$ otherwise. Let $m' = pref_{|m|}([suf_g \oplus s_1])$ and $m'' = pref_{|m|}([suf_g \oplus suf_i])$, respectively. Then:*

$$g \in \mathcal{L}_m \Leftrightarrow d(m) = d(m') + d(m'').$$

**Proof.** Immediate from Lemma 9, by substituting $suf_i$ to $s_2$. □

We note that the term $d(m'')$ in Corollary 1 is known once the don't cares in every prefix of every 2-autocorrelation of $s$ have been pre-computed as done for the case $q = 2$ [4], which requires $O(n^2)$ time.

We next discuss how $d(m')$ can be computed. As will be clear later, this represents the crucial step in our approach. We call the term $s_1$ in Corollary 1 the *complement* of $suf_i$ w.r.t. $m$. Consider the special case in which $pref_{|m|}(suf_i)$ is an occurrence of $pref_{|m|}([suf_j \oplus suf_h])$.

**Property 1.** *Let $i$ be an occurrence of $pref_{|m|}([suf_j \oplus suf_h])$. Let $n^j_{gi}$ be the number of positions $l'$ such that $pref_{|m|}(suf_g)[l'] \oplus pref_{|m|}(suf_i)[l']$ is a solid symbol while $pref_{|m|}(suf_g)[l'] \oplus pref_{|m|}(suf_j)[l']$ is a don't care, and $n^h_{gi}$ be the number of positions $l''$ such that $pref_{|m|}(suf_g)[l''] \oplus pref_{|m|}(suf_i)[l'']$ is a solid symbol while $pref_{|m|}(suf_g)[l''] \oplus pref_{|m|}(suf_h)[l'']$ is a don't care.*[1] *Then:*

$$d(m') = n^j_{gi} + n^h_{gi}.$$

**Proof.** Suppose that $d(m') \neq n^j_{gi} + n^h_{gi}$. This can be due to two different cases, to be analyzed separately:

1. $d(m') > n^j_{gi} + n^h_{gi}$,
2. $d(m') < n^j_{gi} + n^h_{gi}$.

A don't care in position $l$ of the meet $m'$ means that $pref_{|m|}(suf_g)$ has a different character than the complement of $suf_i$ in $l$. Since $|\Sigma| = 2$, then $pref_{|m|}(suf_g)$ and $pref_{|m|}(suf_i)$ have the same character at position $l$.

If Case 1 holds, then there exists at least one position $f$ such that $m'[f]$ is a don't care and $pref_{|m|}(suf_i)[f] = pref_{|m|}(suf_j)[f] = pref_{|m|}(suf_h)[f] = pref_{|m|}(suf_g)[f] = \sigma \in \Sigma$. Then, $s_1[f]$ would be different from all of $pref_{|m|}(suf_i)[f]$, $pref_{|m|}(suf_j)[f]$, and $pref_{|m|}(suf_h)[f]$, but this is a contradiction since $s_1$ is, by construction, equal to at least one of them at each position.

Let us now turn to Case 2, and assume that a position $f$ exists such that:

- $pref_{|m|}(suf_g)[f] = pref_{|m|}(suf_i)[f] = \sigma \in \Sigma$,
- $pref_{|m|}(suf_j)[f] = \sigma' \neq \sigma, \sigma' \in \Sigma$,
- $m'[f] = \sigma'' \in \Sigma$.

Since the alphabet is binary, then one between $\sigma'' = \sigma$ and $\sigma'' = \sigma'$ necessarily holds. If $\sigma'' = \sigma$, then $f$ would correspond to a solid symbol of $m$; but $pref_{|m|}(suf_j)[f] = \sigma'$, thus this would lead to a contradiction. If $\sigma'' = \sigma'$, then $m'[f]$ would be a don't care, since $pref_{|m|}(suf_g)[f] = \sigma$, which is, again, a contradiction. □

**Theorem 3.** *If $pref_{|m|}(suf_i)$ is an occurrence of $pref_{|m|}([suf_j \oplus suf_h])$, then $\mathcal{L}_m$ can be computed in $O(n)$ time.*

**Proof.** For each 3-autocorrelation, we store four integer values representing the number of don't cares and the number of times that two matching suffixes do not match the third one. Such values can be computed as part of the extraction without penalty. By Property 1, checking if $a$ occurs at a specific position of the input string takes constant time, thus $O(n)$ time overall. □

When none among $pref_{|m|}(suf_i)$, $pref_{|m|}(suf_j)$, $pref_{|m|}(suf_h)$ is an occurrence of the meet of the other two suffixes, it is still possible to identify and store some positions, that will be called *anchors*, such that between two anchors such a condition is satisfied for one of the three strings (we take also the first and the last character of the meet as anchors). Depending on how the first anchor is chosen, the overall number of anchors in the meet can be different. We always refer to the anchors configuration such that the overall number of anchors is minimum. The following theorem holds.

**Theorem 4.** *Let $m = [suf_i \oplus suf_j \oplus suf_h]$, let $k$ be the number of don't cares in $m$. The list of occurrences $\mathcal{L}_m$ can be computed in $O(nk)$ time.*

**Proof.** Let $f$ and $g$ be two positions of $m$ corresponding to two consecutive anchors, and suppose that between $f$ and $g$ we have that $pref_{|m|}(suf_i)$ is an occurrence of $pref_{|m|}(suf_j \oplus suf_h)$. Let $m_{fg} = m[f]\, m[f+1] \ldots m[g-1]\, m[g]$, and apply Theorem 3 on $m_{fg}$. Note that both the number of don't cares and the number of times that two agreeing suffixes do not agree with the third one between positions $f$ and $g$ can be obtained in constant time by subtraction from those stored for the corresponding 3-autocorrelation. Since this process has to be repeated for each pair of consecutive anchors, then the overall cost is $O(nk')$, where $k'$ is the number of anchors in $m$. In the worst case, the number of anchors equals the number $k$ of don't cares in the 3-meet, which proves the claim. □

Thus, the overall cost of this approach is $O(n^3k)$. For a general quorum $q$, it is easy to see that Step 2 of the general paradigm for basis extraction can be computed analogously in $O(\frac{n^q}{q!}k)$ time by storing, for each $q$-meet, the numbers of times that two matching suffixes do not match the $q-2$ remaining ones.

### 4.2. General alphabets

For ease of exposition, we consider at first the case $q = 2$ and then generalize it for higher quorums.

Let $k$ be the number of don't cares of the autocorrelation (or autocorrelations) containing the maximum number of don't cares.

[1] Note that such numbers may be computed in constant time during the extraction of the corresponding 3-meet.

**Lemma 10.** *Let $m = [suf_i \oplus suf_j]$ and $\mathcal{P}_{ij}$ be the list of the don't care positions in $m$. Furthermore, let $\mathcal{P}_{ih}$ and $\mathcal{P}_{jh}$ be the lists of the don't care positions $pref_{|m|}([suf_i \oplus suf_h])$ and $pref_{|m|}([suf_j \oplus suf_h])$, respectively.*

*Then for any given position $h$ of $s$:*

$$h \in \mathcal{L}_m \Leftrightarrow \mathcal{P}_{ih} \cup \mathcal{P}_{jh} = \mathcal{P}_{ij}.$$

**Proof.** Assume $h \in \mathcal{L}_m$ and $\mathcal{P}_{ih} \cup \mathcal{P}_{jh} \neq \mathcal{P}_{ij}$. Then, there exists at least one homologous don't care in $[suf_i \oplus suf_h]$ and $[suf_j \oplus suf_h]$ the position $f$ of which is not in $\mathcal{P}_{ij}$. Since $f$ is not in $\mathcal{P}_{ij}$, then $suf_i[f] = suf_j[f] = \sigma \in \Sigma$. At the same time, $suf_h[f] = \sigma' \in \Sigma$, $\sigma' \neq \sigma$, and $m[f] = \sigma$. Therefore, $h$ cannot be an occurrence of $m$, a contradiction.

For the second part of the proof, if $\mathcal{P}_{ih} \cup \mathcal{P}_{jh} = \mathcal{P}_{ij}$, then the meet between the $|m|$-th prefixes of $[suf_i \oplus suf_h]$ and $[suf_h \oplus suf_j]$ generates $m$, which proves the claim. □

**Corollary 2.** *Under the conditions of Lemma 10, verifying if $h$ is an occurrence of $m$ takes time $O(k)$.*

**Proof.** This is the time required to compute the union of the two sets $\mathcal{P}_{ih}$ and $\mathcal{P}_{jh}$ of size $O(k)$ as well as to perform the comparison with $\mathcal{P}_{ij}$. □

The following theorem holds.

**Theorem 5.** *The basis of irredundant 2-motifs containing at most $k$ don't cares each for a string $s$ of $n$ characters over a general alphabet $\Sigma$ can be computed in $O(kn^2)$ time.*

**Proof.** Extracting the $n$ autocorrelations and discarding the redundant motifs takes $O(n^2)$ time. For each position $j$ of $s$, verifying if $j$ is an occurrence of the autocorrelation $m$ is done in $O(k)$ time by Corollary 2. □

Lemma 10 and Theorem 5 can be generalized as follows.

**Lemma 11.** *Let $h$ be a position in $s$ and $m$ be a $q$-autocorrelation of $s$ generated by the suffixes $suf_{i_1}, suf_{i_2}, \ldots, suf_{i_q}$. Let $\mathcal{P}$ be the list of the don't care positions in $m$. Furthermore, let $\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_q$ be the lists of the don't care positions in the $|m|$-th prefixes $m_1, m_2, \ldots, m_q$ of the $q$ meets that can be formed between $suf_h$ and the $q$ suffixes $suf_{i_1}, suf_{i_2}, \ldots, suf_{i_q}$, taken $q-1$ at the time. Then:*

$$h \in \mathcal{L}_m \Leftrightarrow \mathcal{P}_1 \cup \mathcal{P}_2 \cup \cdots \mathcal{P}_q = \mathcal{P}.$$

**Proof.** Let $h \in \mathcal{L}_m$ and assume that $\mathcal{P}_1 \cup \mathcal{P}_2 \cup \cdots \mathcal{P}_q \neq \mathcal{P}$. Then, there exists at least one don't care in $m_1, m_2, \ldots, m_q$ whose position is not in $\mathcal{P}$. Let $f$ be the position of such a don't care. Since $f$ is not in $\mathcal{P}$, then $suf_{i_1}[f] = suf_{i_2}[f] = \cdots = suf_{i_q}[f] = \sigma \in \Sigma$. At the same time, $suf_h[f] = \sigma' \in \Sigma$, $\sigma' \neq \sigma$, and $m[f] = \sigma$. Thus, $h$ cannot be an occurrence of $m$, a contradiction.

Conversely, if $\mathcal{P}_1 \cup \mathcal{P}_2 \cup \cdots \mathcal{P}_q = \mathcal{P}$, then the meet between the $|m|$-th prefixes of $m_1, m_2, \ldots, m_q$ generates $m$. □

**Theorem 6.** *The basis of irredundant $q$-motifs containing at most $k$ don't cares each for a string $s$ of $n$ characters over a general alphabet $\Sigma$ can be computed in $O\left(kn^q \frac{q}{(q-1)!}\right)$ time.*

**Proof.** Extracting all the $q$-autocorrelations and discarding the redundant motifs takes $O\left(\frac{n^q}{q!}\right)$ time. For each position $j$ of $s$, verifying if $j$ is an occurrence of the autocorrelation $m$ is done in $O(kq^2)$ time by performing the union of $q$ sets of size $O(k)$. □

Theorems 4 and 6 pave the way for designing a technique to extract bases made of motifs with an a priori bounded number of don't cares. As already discussed in the Introduction, fixing such a bound can be useful to avoid situations where the motifs in the basis have too poor informative content for large quorums.

Let us assume a bound on the total number $k$ of don't cares admitted in any element of the basis. This requirement may be enforced by neglecting autocorrelations that exceed the bound in the first place, and then keeping only those ones that are irredundant with respect to the set of autocorrelations generated this way. Of course, there is no guarantee that such a basis can generate all the possible motifs occurring in the input string, differently from the basis where no bounds are fixed a-priori. On the other hand, such a bounded-basis can be viewed as a set of motifs satisfying a specific constraint (on the number of don't cares) together with the property to be irredundant each other. Notice that this is one case of a general, yet largely unexplored, problem consisting in defining motif bases constrained to feature specific characteristics.

Since each autocorrelation corresponds to a mutual shift between two identical copies of $s$, it is trivial to infer the number of leading don't cares in every consensus of the form $s \oplus suf_i$: just sweep over $s$ comparing $s[j]$ with $s[j+i]$ looking for the position of the earliest match. For each mutual displacement, this way it takes thus a linear number of character comparisons to identify the first matching pair, hence this is doable for all displacements in $O(n^2)$ overall. Once this is known, testing whether the rest of a consensus converts in an autocorrelation with no more than $k$ don't cares is accomplished in $O(k)$ steps by resort to a known application of lowest common ancestors on subword trees (see, e.g., [11]). In conclusion, the at most $(n-1)$ 2-autocorrelations abiding by the $k$ don't care bound can be extracted in $O(kn)$ overall time. For a general quorum $q$, this cost switches to $O\left(k\frac{n^{q-1}}{q!}\right)$. Thus, the overall cost for basis extraction is $O\left(kn^q \frac{q}{(q-1)!}\right)$ from the previous theorems, that becomes $O\left(n^q \frac{q}{(q-1)!}\right)$ in this case, since $k$ is a constant to be fixed a-priori.

## 5. Conclusive remarks

In the previous sections we showed useful approaches for basis extraction holding for any quorum with both binary and general alphabets. In particular, the technique based on anchors for binary alphabets requires $O(\frac{n^q}{q!}k)$ time, where $k$ is the maximum number of don't cares in any $q$-autocorrelation. The approach for general alphabets based on efficiently checking if a position of $s$ is an occurrence of a given autocorrelation takes $O\left(kn^q \frac{q}{(q-1)!}\right)$ time. Both computational complexities turn out to be impractical when $k, q \simeq n$. However, in practical applications, motifs with a high number of don't cares are meaningless and, as specified before, imposing that $k$ is fixed a-priori such that $k \ll n$ is in practice a sensible choice. This has important consequences on the speed of the proposed algorithms, since the number of $q$-autocorrelations with at most $k$ don't cares obviously decreases when the quorum increases and such a number crucially influences the efficiency of all the presented approaches.

Furthermore, in many application contexts, such as for example computational biology or time series analysis, the quorum $q$ is greater than two but not so large to have values close to $n$. In these cases, the contributions provided by this work are even more significant since, first of all, important properties previous approaches are based on work only for $q = 2$, as shown in the first part of this paper. Then, if $k, q \ll n$ the computational complexities become affordable in practice and, at the best of our knowledge, this is the first attempt to control the speed of the algorithms through the allowed level of indeterminacy.

We finally sketch how Step 2 of the basis extraction paradigm can be further optimized, when we deal with general alphabets. Let us partition the set $\mathcal{A}$ of $q$-autocorrelations into two subsets $\mathcal{A}'$ and $\mathcal{A}''$ such that: $\mathcal{A}'$ includes those autocorrelations $m = [suf_{i_1} \oplus suf_{i_2} \oplus \cdots \oplus suf_{i_q}]$ where at least one $suf_{i_j} \in \{suf_{i_1}, \ldots, suf_{i_q}\}$ is an occurrence of the meet yielding from the remaining suffixes that generate $m$; $\mathcal{A}''$ includes all the other autocorrelations. Let $m' \in \mathcal{A}'$ and suppose that $suf_{i_j}$ is an occurrence of the meet $m''$ among the other $q-1$ suffixes generating $m'$. Then, a position $h$ of $s$ is an occurrence of $m'$ if the $q$-autocorrelation $m''' = suf_h \oplus m''$ coincides with $m'$. Since both $m'$ and $m'''$ are computed during Step 1, and all autocorrelations can be checked for duplicate that time, this test can be performed in constant time for each autocorrelation in $\mathcal{A}'$. Thus, the only autocorrelations whose occurrence lists computation concurs to the overall cost are those in $\mathcal{A}''$. For each $m \in \mathcal{A}''$, the number of positions of $s$ to be checked can be pre-computed by exploiting Lemma 7 in order to discard those positions of $s$ that of course are not occurrences of $m$. Both the number of autocorrelations and the number of positions of $s$ to be checked to perform Step 2 can be reduced this way. Note that, the more a string is repetitive, the more the size of $\mathcal{A}''$ is smaller than $|\mathcal{A}'|$. The suggested optimization is thus particularly suitable in those contexts characterized by high repetitiveness.

Several challenges remain still open. Notably, among them, the study of other kinds of bases, whose irredundant motifs are made for example of several solid blocks that occur at a variable distance in the input string. Another interesting topic is the design of data structures suitable to make the discovery process more efficient.

## Acknowledgments

The author is grateful to A. Apostolico and L. Palopoli for interesting discussions and suggestions about the topic. This research was partially supported by the Italian Ministry for Education, University and Research under the grant PON 01_02477 2007-2013 - FRAME.

## References

[1] A. Apostolico, M. Comin, L. Parida, Mining, compressing and classifying with extensible motifs, Algorithms Mol. Biol. 1 (4) (2006).
[2] A. Apostolico, L. Parida, Incremental paradigms of motif discovery, J. Comp. Biol. 11 (1) (2004) 15–25.
[3] A. Apostolico, L. Parida, S.E. Rombo, Motif patterns in 2D, Theor. Comput. Sci. 390 (1) (2008) 40–55.
[4] A. Apostolico, C. Tagliacollo, Incremental discovery of the irredundant motif bases for all suffixes of a string in $O(n^2 \log n)$ time, Theor. Comput. Sci. 408 (2-3) (2008) 106–115.
[5] A. Apostolico, C. Tagliacollo, Optimal extraction of irredundant motif bases, Int. J. Found. Comput. Sci. 21 (6) (2010) 1035–1047.
[6] A. Brazma, I. Jonassen, I. Eidhammer, D. Gilbert, Approaches to the automatic discovery of patterns in biosequences, J. Comp. Biol. 5 (2) (1998) 277–304.
[7] M.J. Fisher, M.S. Paterson, String matching and other products, in: Proc. of Complexity of Comp. (SIAM-AMS), 1974, pp. 113–125.
[8] R. Grossi, A. Pietracaprina, N. Pisanti, G. Pucci, E. Upfal, F. Vandin, MADMX: a novel strategy for maximal dense motif extraction, in: Algorithms in Bioinformatics, 9th International Workshop, WABI 2009, 2009, pp. 362–374.
[9] R. Grossi, A. Pietracaprina, N. Pisanti, G. Pucci, E. Upfal, F. Vandin, MADMX: a strategy for maximal dense motif extraction, J. Comp. Biol. 18 (4) (2011) 535–545.
[10] R. Grossi, N. Pisanti, M. Crochemore, M.-F. Sagot, Bases of motifs for generating repeated patterns with wild cards, IEEE/ACM Trans. Comp. Biol. Bioinform. 2 (1) (2005) 40–50.
[11] D. Gusfield, Algorithms on strings, trees, and sequences: computer science and computational biology, Camb. Univ. Press, NY, USA, 1997.
[12] L. Parida, Algorithmic techniques in computational genomics, Ph.D. Thesis, Department of Computer Science, New York University, 1998.
[13] L. Parida, Pattern Discovery in Bioinformatics: Theory & Algorithms, first ed., Chapman & Hall/CRC, 2007.
[14] L. Parida, I. Rigoutsos, A. Floratos, D. Platt, Y. Gao, Pattern discovery on character sets and real-valued data: linear bound on irredundant motifs and polynomial time algorithms, in: Proceedings of ACM-SIAM Symposium on Discrete Algorithms, SODA 2000, 2000, pp. 297–308.
[15] P. Patel, E.J. Keogh, J. Lin, S. Lonardi, Mining motifs in massive time series databases, in: Proccedings of the 2002 IEEE International Conference on Data Mining, ICDM 2002, 2002, pp. 370–377.
[16] J. Pelfrêne, S. Abdeddaïm, J. Alexandre, Extracting approximate patterns, in: Proc. of CPM, 2003, pp. 328–347.

[17] J. Pelfrêne, S. Abdeddaïm, J. Alexandre, Extracting approximate patterns, J. Discrete Algorithms 3 (2–4) (2005) 293–320.
[18] N. Pisanti, M. Crochemore, R. Grossi, M.-F. Sagot, A basisi of tiling motifs for generating repeated patterns and its complexity for higher quorum, in: Proc. of Math. Found. Comp. Sc., MFCS, 2003, pp. 622–632.
[19] N. Pisanti, M. Crochemore, R. Grossi, M.-F. Sagot, A comparative study of bases for motif inference, in: C. Iliopoulos, T. Lecroq (Eds.), String Algorithmics, KCL Publications, 2004, pp. 195–226.