

# Incremental Paradigms of Motif Discovery

ALBERTO APOSTOLICO<sup>1</sup> and LAXMI PARIDA<sup>2</sup>

## ABSTRACT

We examine the problem of extracting maximal irredundant motifs from a string. A combinatorial argument poses a linear bound on the total number of such motifs, thereby opening the way to the quest for the fastest and most efficient methods of extraction. The basic paradigm explored here is that of iterated updates of the set of irredundant motifs in a string under consecutive unit symbol extensions of the string itself. This approach exposes novel characterizations for the base set of motifs in a string, hinged on notions of partial order. Such properties support the design of ad hoc data structures and constructs, and lead to develop an  $O(n^3)$  time incremental discovery algorithm.

Key words:

AU1

## 1. INTRODUCTION

STRING PATTERNS AKIN TO REPETITIVE STRUCTURES are variously pursued in string processing applications (Apostolico and Galil, 1997). Notable among such structures are repetitions, cadences, palindromes, and other redundancies. In molecular biology and genomics studies, a prominent role has been gained by the notion of a motif, which in loose terms consists of a subsequence of intermittently solid characters interspersed with wild characters and appearing more or less frequently in an input sequence. Because motifs seem to be implicated in many string manipulations and biological functions, their discovery is of particular interest. We refer to the quoted literature (Apostolico and Galil, 1997; Apostolico and Atallah, 2002; Rigoutsos *et al.*, 2000; Parida *et al.*, 2000; Wang *et al.*, 1999) for more comprehensive discussion. In general, the discovery process is made particularly difficult by the fact that the number of candidate motifs in a string grows exponentially with the length of that string. Attempts at circumventing this are made by introducing special classes of motifs subject to certain special conditions. For instance, it seems natural to exclude from consideration motifs that could be enriched in terms of solid characters without prejudice in the corresponding frequency. Along these lines, a class of motifs called *irredundant* can be identified that grows linearly with input size. This opens the way to the quest for fast and efficient methods of extraction for such motifs.

AU2

AU3

AU2

We examine here the problem of discovering all maximal irredundant motifs from a string. The basic paradigm explored is that of iterated updates of the set of base motifs in a string under consecutive unit symbol extensions of the string itself. This approach exposes novel characterizations for the base set of

---

<sup>1</sup>Dipartimento di Ingegneria dell' Informazione, Università di Padova, Padova, Italy, and Department of Computer Sciences, Purdue University, West Lafayette, IN.

<sup>2</sup>IBM Thomas J. Watson Research Center, Yorktown Heights, NY 10598.

motifs in a string, hinged on notions of partial order, which may be of independent interest. Such properties support the development of ad hoc data structures and constructs, leading to an  $O(n^3)$  incremental discovery algorithm.

This paper is organized as follows. In the next section, we recall some basic definitions and properties and derive the combinatorial facts subtending to our construction. Section 3 is devoted to the design and description of our algorithm. The last section lists conclusions and plans of future work.

## 2. PROPERTIES

Let  $s = s_1s_2 \dots s_n$  be a *string* of length  $s = |n|$  over an alphabet  $\Sigma$ . We use  $suf_i$  to denote the suffix  $s_i s_{i+1} \dots s_n$  of  $s$ . A character from  $\Sigma$ , say  $\sigma$ , is called a *solid* character and “.” is called a *don't care* character.

**Definition 1 ( $\sigma_1 \prec, =, \preceq \sigma_2$ ).** *If  $\sigma_1$  is a don't care character, then  $\sigma_1 \prec \sigma_2$ . If both  $\sigma_1$  and  $\sigma_2$  are identical characters in  $\Sigma$ , then  $\sigma_1 = \sigma_2$ . If either  $\sigma_1 \prec \sigma_2$  or  $\sigma_1 = \sigma_2$  holds, then  $\sigma_1 \preceq \sigma_2$ .*

**Definition 2 ( $p$  occurs at  $l$ , cover).** *A string,  $p$ , on  $\Sigma \cup \text{“.”}$ , occurs at position  $l$  in  $s$  if  $p[j] \preceq s[l+j-1]$  holds for  $1 \leq j \leq |p|$ . String  $p$  is said to cover the interval  $[l, l + |p| - 1]$  on  $s$ .*

A *motif* is any element of  $\Sigma$  or any string on  $\Sigma \cdot (\Sigma \cup \{.\})^* \cdot \Sigma$ .

**Definition 3 (motif  $m$ , location list  $\mathcal{L}_m$ ).** *Given a string  $s$  on alphabet  $\Sigma$  and a positive integer  $k$ ,  $k \leq |s|$ , a string  $m$  on  $\Sigma \cup \text{“.”}$  is a motif with location list  $\mathcal{L}_m = (l_1, l_2, \dots, l_p)$ , if all of the following hold: (1)  $m[1], m[|m|] \in \Sigma$ , (2)  $p \geq k$ , and (3) there does not exist a location  $l$ ,  $l \neq l_i$ ,  $1 \leq i \leq p$  such that  $m$  occurs at  $l$  on  $s$  (the location list is of maximal size).*

The first condition ensures that the first and last characters of the motif are solid characters; if don't care characters are allowed at the ends, the motifs can be made arbitrarily long in size without conveying any extra information. The third condition ensures that any two distinct location lists must correspond to distinct motifs.

In the following, a motif that occurs at least  $k$  times will be called a *k-motif*. Consider  $s = abcdabcd$ . Using the definition of motifs, the different 2-motifs are as follows:  $m_1 = ab$  with  $\mathcal{L}_{m_1} = \{1, 5\}$ ,  $m_2 = bc$  with  $\mathcal{L}_{m_2} = \{2, 6\}$ ,  $m_3 = cd$  with  $\mathcal{L}_{m_3} = \{3, 7\}$ ,  $m_4 = abc$  with  $\mathcal{L}_{m_4} = \{1, 5\}$ ,  $m_5 = bcd$  with  $\mathcal{L}_{m_5} = \{2, 6\}$ , and  $m_6 = abcd$  with  $\mathcal{L}_{m_6} = \{1, 5\}$ .

Notice that  $\mathcal{L}_{m_1} = \mathcal{L}_{m_4} = \mathcal{L}_{m_6}$  and  $\mathcal{L}_{m_2} = \mathcal{L}_{m_5}$ . Using the notation  $\mathcal{L}+i = \{x+i \mid x \in \mathcal{L}\}$ ,  $\mathcal{L}_{m_5} = \mathcal{L}_{m_6}+1$  and  $\mathcal{L}_{m_3} = \mathcal{L}_{m_6}+2$  hold. We call the motif  $m_6$  *maximal* as  $|m_6| > |m_1|, |m_4|$  and  $|m_5| > |m_2|$ . Motifs  $m_1, m_2, m_3, m_4$ , and  $m_5$  are nonmaximal motifs.

We give the definition of maximality below. In intuitive terms, a motif  $m$  is *maximal* if we cannot make it more specific or longer while retaining the list  $\mathcal{L}_m$  of its occurrences in  $s$ .

**Definition 4 ( $m_1 \preceq m_2$ ).** *Given two motifs  $m_1$  and  $m_2$  with  $|m_1| \leq |m_2|$ ,  $m_1 \preceq m_2$  holds if for some  $d$ ,  $0 \leq d \leq |m_2| - |m_1|$ ,  $m_1[j] \preceq m_2[j+d]$ ,  $1 \leq j \leq |m_1|$ .*

We also say in this case that  $m_1$  is a *submotif* of  $m_2$  and that  $m_2$  *implies* or *extends* or *covers*  $m_1$ . If, moreover, the first characters of  $m_1$  and  $m_2$  match, then  $m_1$  is also called a *prefix* of  $m_2$ .

For example, let  $m_1 = ab..e$ ,  $m_2 = ak..e$ , and  $m_3 = abc.e.g$ . Then  $m_1 \preceq m_3$ , and  $m_2 \not\preceq m_3$ . The following lemma is straightforward to verify.

**Lemma 1.** *If  $m_1 \preceq m_2$ , then  $\mathcal{L}_{m_1} \supseteq \mathcal{L}_{m_2} + d$ , and if  $m_1 \preceq m_2$ ,  $m_2 \preceq m_3$ , then  $m_1 \preceq m_3$ .*

**Definition 5 (maximal motif).** *Let  $m_1, m_2, \dots, m_k$  be the motifs in a string  $s$ . A motif  $m_i$  is maximal in composition if and only if there exists no  $m_l$ ,  $l \neq i$  with  $\mathcal{L}_{m_i} = \mathcal{L}_{m_l}$  and  $m_i \preceq m_l$ . A motif  $m_i$ , maximal in composition, is also maximal in length if and only if there exists no motif  $m_j$ ,  $j \neq i$ , such that  $m_i$  is a submotif of  $m_j$  and  $|\mathcal{L}_{m_i}| = |\mathcal{L}_{m_j}|$ . A maximal motif is maximal both in composition and in length.*

Clearly, the intent in requiring maximality in composition and length is to limit the number of motifs that may be usefully extracted and accounted for in a string. However, the notion of maximality alone does not suffice to bound the number of such motifs. It can be shown that there are strings that have an unusually large number of maximal motifs without conveying extra information about the input.

A maximal motif  $m$  is *irredundant* if  $m$  and the list  $\mathcal{L}_m$  of its occurrences cannot be deduced by the union of a number of lists of other maximal motifs. Conversely, we call a motif  $m$  *redundant* if  $m$  (and its location list  $\mathcal{L}_m$ ) can be deduced from the other motifs *without* knowing the input string  $s$ . More formally:

**Definition 6 (redundant motif).** *A maximal motif  $m$ , with location list  $\mathcal{L}_m$ , is redundant if there exist maximal submotifs  $m_i$ ,  $1 \leq i \leq p$ , such that  $\mathcal{L}_m = \mathcal{L}_{m_1} \cup \mathcal{L}_{m_2} \dots \cup \mathcal{L}_{m_p}$  (i.e., every occurrence of  $m$  on  $s$  is already covered by one of the motifs  $m_1, m_2, \dots, m_p$ ).*

**Definition 7 (irredundant motif).** *A maximal motif that is not redundant is called an irredundant motif.*

We use  $\mathcal{B}_i$  to denote the set of irredundant motifs in  $su f_i$ . Set  $\mathcal{B}_i$  is called the *basis* for the motifs of  $su f_i$ . Thus, in particular, the basis  $\mathcal{B}$  of  $s$  coincides with  $\mathcal{B}_1$ .

**Definition 8 (basis).** *Given a sequence  $s$  on an alphabet  $\Sigma$ , let  $\mathcal{M}$  be the set of all maximal motifs on  $s$ . A set of maximal motifs  $\mathcal{B}$  is called a basis of  $\mathcal{M}$  iff the following hold: (1) for each  $m \in \mathcal{B}$ ,  $m$  is irredundant with respect to  $\mathcal{B} - \{m\}$ , and (2) letting  $\mathbf{G}(\mathcal{X})$  be the set of all the redundant maximal motifs generated by the set of motifs  $\mathcal{X}$ , then  $\mathcal{M} = \mathbf{G}(\mathcal{B})$ .*

In general,  $|\mathcal{M}| = \Omega(2^n)$ . The natural attempt now is to obtain as small a basis as possible. Before getting to that, we examine some basic types of maximality.

**Lemma 2.** *Let  $m$  be a maximal motif with no don't care characters and  $|\mathcal{L}_m| = 1$ , then  $m = s$ .*

**Proof.** Any motif with those properties can be completed into  $s$ , by the notion of maximality. ■

**Lemma 3.** *Let  $m$  be a maximal motif with at least one don't care character, then  $|\mathcal{L}_m| \geq 2$ .*

**Proof.** Under the hypothesis, it must be  $|m| > 1$ . The rest is a straightforward consequence of the notion of maximality. ■

Lemmas 2 and 3 tell us that, other than the string  $s$  itself and the characters of the alphabet, the only maximal motifs of interest have more than one occurrence. Solid motifs, i.e., motifs that do not contain any don't care symbol, enjoy a number of nice features that make it pedagogically expedient to consider them separately. Let the equivalence relation  $\equiv_s$  be defined on a string  $s$  by setting  $y \equiv_s w$  if  $\mathcal{L}_y = \mathcal{L}_w$ . Recall that the *index* of an equivalence relation is the number of equivalence classes in it. The following well known fact from Blumer *et al.* (1985) shows that the number of maximal motifs with no don't care characters is linear in the length of the text string. It descends from the observation that for any two substrings  $y$  and  $w$  of  $s$ , if  $\mathcal{L}_y \cap \mathcal{L}_w$  is not empty, then  $y$  is a prefix of  $w$  or vice versa.

**Fact 1.** *The index  $k$  of the equivalence relation  $\equiv_x$  obeys  $k \leq 2n$ .*

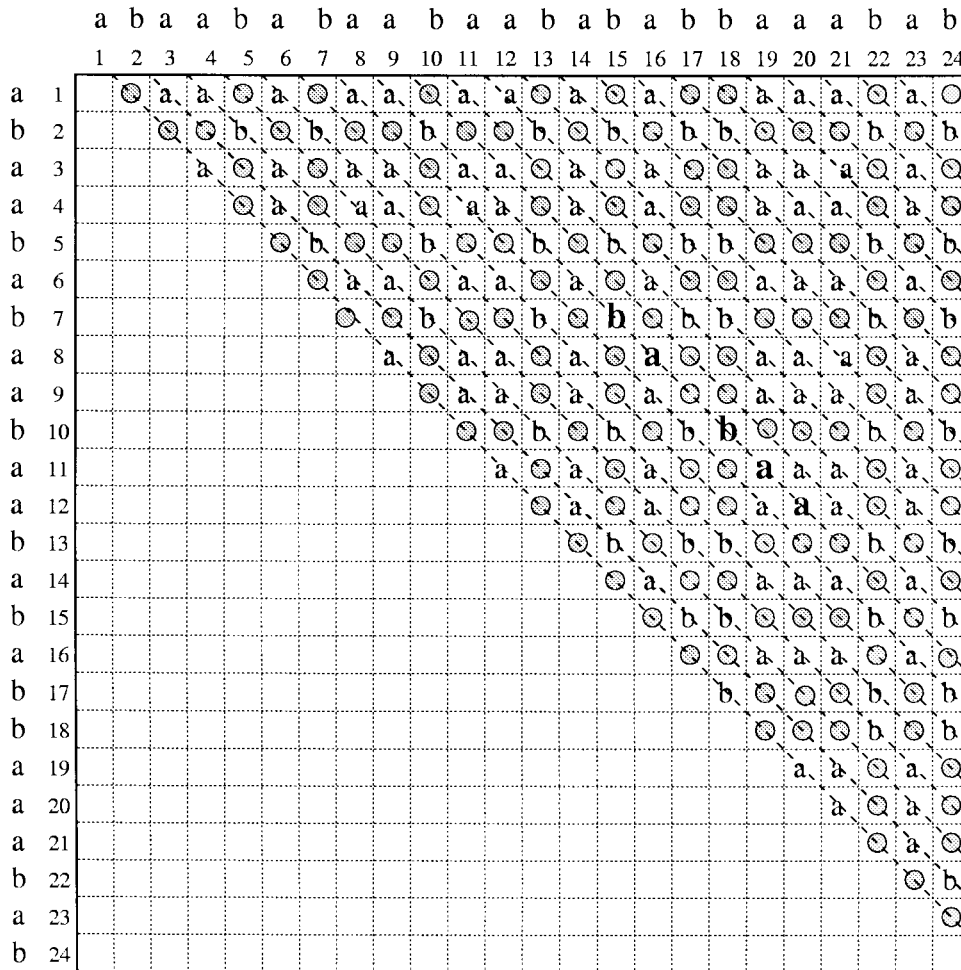
Now let  $x$  and  $y$  be two strings with  $m = |x| \leq |y| = n$ . The *consensus* of  $x$  and  $y$  is the string  $z_1 z_2 \dots z_m$  on  $\Sigma \cup \text{"."}$  defined as  $z_i = x_i$  if  $x_i = y_i$  and  $z_i = \text{"."}$  otherwise ( $i = 1, 2, \dots, m$ ). Deleting all leading and trailing don't care symbols from  $z$  yields a (possibly empty) motif that is called the *meet* of  $x$  and  $y$ . The following general property shows that the irredundant 2-motifs are to be found among the pairwise meets of all suffixes of  $s$ , which will play a central role in our constructions.

**Theorem 1.** Every irredundant 2-motif in  $s$  is the meet of two suffixes of  $s$ .

**Proof.** Let  $m$  be a 2-motif in  $\mathcal{B}$ , and  $\mathcal{L}_m = (l_1, l_2, \dots, l_p)$  be its occurrence list. The claim is true for  $p = 2$ . Indeed, let  $i = l_1$  and  $j = l_2$  and consider the meet  $m'$  of  $\text{suf}_i$  and  $\text{suf}_j$ . By the maximality in composition of  $m$ , we have that  $m' \preceq m$ . On the other hand, for any motif  $\hat{m}$  with occurrences at  $i$  and  $j$ , it must be  $\hat{m} \preceq m'$ , whence, in particular,  $m \preceq m'$ . Thus,  $m = m'$ . Assume now  $p \geq 3$  and that there is no pair of indices  $i$  and  $j$  in  $\mathcal{L}_m$  such that  $m$  is the meet of  $\text{suf}_i$  and  $\text{suf}_j$ . Again, for any choice of  $i$  and  $j$  in  $\mathcal{L}_m$ , we must have that  $m \preceq m'$ , where  $m'$  denotes as before the meet of  $\text{suf}_i$  and  $\text{suf}_j$ . Therefore, we have that  $m \preceq m'$  but  $m \neq m'$  for all choices of  $i$  and  $j$ . Assume now one such choice is made. By the maximality of  $m$ , it cannot be that  $m'$  is the meet of all suffixes with beginning in  $\mathcal{L}_m$ . Therefore, there must be at least one index  $k$  such that  $m'$  differs either from the meet of  $\text{suf}_k$  and  $\text{suf}_i$  or from the meet of  $\text{suf}_k$  and  $\text{suf}_j$ , or from both. To fix the ideas, let  $m''$  be this second meet. Since  $m \preceq m''$  and  $m \preceq m'$ , then  $\mathcal{L}_{m'}$  and  $\mathcal{L}_{m''}$  are sublists of  $\mathcal{L}_m$ , by Lemma 1. In other words,  $\mathcal{L}_m$  can be decomposed into two or more lists of maximal motifs such that their union implies  $m$  and its occurrences. But this contradicts the assumption that  $m$  is irredundant. ■

A convenient graphical representation of the pairwise consensus of all suffixes of  $s$  is offered by the matrix  $M$  of matching character pairs for the input string, as exemplified in Fig. 1. It is enough to fill  $M[i, j]$  for  $i = 1, 2, \dots, n - 1$  and  $j > i$ , and we set there  $M[i, j] = s_i$  if  $s_i = s_j$ , and  $M[i, j] = \text{"."}$  (represented by a solid circle in the figure) elsewhere. As is easily seen, the diagonal beginning at column

**F1**



**FIG. 1.** An example match matrix showing pairwise suffix meets.

$j$ , ( $2 \leq j \leq n$ ) in the matrix  $M$  is the consensus between  $s = suf_1$  and  $suf_j$ , and the set of diagonals incorporates all meets: for any  $i < n$  and  $j > i$ , following the diagonal from  $M[i, j]$  to the end yields the consensus of  $suf_i$  and  $suf_j$ , and deleting don't cares at the beginning and end of such a consensus gives the corresponding meet. As an illustration, the meet of  $suf_7$  and  $suf_{15}$  is represented in bold in our figure. We find it convenient to assume that for each meet  $m$ , the sorted list  $\mathcal{L}_m$  is appended to the entry that corresponds to the first match of  $m$  in the rightmost diagonal where  $m$  appears. For example, in Fig. 1, the match  $M[6, 21]$  defines  $m_1 = aba$  and points to the list  $\mathcal{L}_{m_1} = (1, 4, 6, 9, 12, 14, 21)$ . The match  $M[1, 20]$  identifies  $m_2 = a..ab$ , and similarly yields to the list  $\mathcal{L}_{m_2} = (1, 3, 6, 9, 11, 20)$ .

Theorem 1 supports an immediate and natural linear bound for the cardinality of our set of irredundant 2-motifs, which gives the notion somewhat of a graceful axiomatic dignity: by maximality, these motifs are just some of the  $n - 1$  meets of  $s$  with its own suffixes. The proof given below bundles this with the linearity of a special subset of the set of  $O(n^2)$  suffixes of irredundant 2-motifs.

**Theorem 2.** *The number of irredundant 2-motifs in a string  $x$  of  $n$  characters is  $O(n)$ .*

**Proof.** Consider the set  $\{z^{(k)}\}$  of strings obtained by taking the *reverse* of the meet of  $suf_1$  and  $suf_k$ ,  $k = 2, 3, \dots, n$ . In our figure, the  $z^{(k)}$ 's are seen by reading the diagonals of the matrix  $M$  bottom-up, beginning at the lowest and stopping at the highest match in each diagonal. Thus, for any  $j > i$ , the reverse of the meet of  $suf_i$  and  $suf_j$  consists of a prefix of  $z^{(k)}$  where  $k = j - i$ . With % a special endmarker not in  $\Sigma$ , insert now the strings  $z^{(k)}\%$  ( $k = 2, 3, \dots, n$ ) in a compact trie (digital search tree)  $\mathcal{T}$ . Note that each  $z^{(k)}$  is a string over the alphabet  $\Sigma \cup \text{"."}$ , i.e., the don't care is treated as just any other character. The number of leaves in the trie  $\mathcal{T}$  is clearly linear in  $n$ , and so is the number of internal nodes and arcs. By Theorem 1, for any irredundant motif  $m$  in  $s$ , the end of the reverse  $m^r$  of  $m$  occurs at some node or arc on a path from the root of  $\mathcal{T}$ . It is clear that the only irredundant motif that can possibly end at a node in this way is that spelled out by the labels of the path from that node to the root. We claim that no two reverse irredundant motifs may end on a same arc of  $\mathcal{T}$ . Indeed, assume for a contradiction that the paths of  $m_1^r$  and  $m_2^r$  end on the same arc, and to fix the ideas assume that the string  $m_1^r$  is a prefix of  $m_2^r$ , i.e.,  $m_2^r = m_1^r v$  with  $|v| = h$ . Let  $m_2^r$  be the reverse meet for of  $suf_i$  and  $suf_j$ . Then,  $m_1^r$  is the reverse meet of  $suf_{i+h}$  and  $suf_{j+h}$  ( $m_1$  begins  $h$  positions below the start of  $m_2$  on the same diagonal in  $M$ ). Clearly, any occurrence of  $m_2^r$  in  $s^r$  is also an occurrence of  $m_1^r$ ; thus, by the irredundancy of  $m_1$ , there must be at least one occurrence of  $m_1^r$  in  $s^r$  that does not complete into an occurrence of  $m_2^r$ . Now, among such occurrences of  $m_1$ , there must be at least one, say, the one at position  $g$  in  $s$ , such that the meet of  $suf_g$  and  $suf_{i+h}$ , or that of  $suf_g$  and  $suf_{j+h}$  (or both), coincides precisely with the meet of  $suf_{i+h}$  and  $suf_{j+h}$ . Otherwise, by an argument already used in Theorem 1,  $m_1$  would be implied by other motifs and thus would be redundant. Assume the meet of  $suf_g$  and  $suf_{i+h}$  coincides with  $m_1$  and hence with the meet of  $suf_{i+h}$  and  $suf_{j+h}$ . Taking w.l.o.g.  $g < i + h$ , the reverse of the meet of  $suf_g$  and  $suf_{i+h}$  is a prefix of  $z^{(i+h-g)}\%$ . The latter is in the trie. But then  $m_1^r$  is a prefix of  $z^{i+h-g}\%$  while  $m_2^r$  is not, whence the paths of  $z^{j-i}\%$  and  $z^{(i+h-g)}\%$  must diverge between the endpoints of  $m_1^r$  and  $m_2^r$ . ■

We leave it as an exercise to extend the notion of meet and Theorem 1 to arbitrary  $k$ , to the effect that every irredundant  $k$ -motif is the meet of  $k$  suffixes.

Now let  $m$  be a motif,  $|m| > 1$ . Since  $m$  must start with some solid character, call it  $\sigma$ , it is always possible to encode  $m$  as  $m = \sigma dm'$  where  $d \geq 0$  and  $m'$  is some other motif or its encoding,  $|m'| > 0$ . Through the rest of the discussion, we are primarily interested in motifs  $m$  with at least two solid characters. We are specifically interested in the evolution of the set of irredundant motifs when a symbol is prepended to a string. More specifically, we are interested in the mechanics of the transformation of  $\mathcal{B}_{i+1}$  into  $\mathcal{B}_i$  in the transition from  $suf_{i+1}$  to  $suf_i$ ,  $i = n - 1, n - 2, \dots, 1$ . For the remainder of this section, we examine this problem in general terms, before we focus on the case  $k = 2$  in the next section.

**Lemma 4.** *For any  $m \in \mathcal{B}_i$  not in  $\mathcal{B}_{i+1}$ ,  $|m| \geq 2$ , there are  $\sigma \in \Sigma$ ,  $d \geq 0$  and a maximal motif  $m'$  such that  $m = \sigma dm'$  where  $m'$  is (the prefix of) a motif in  $\mathcal{B}_f$ ,  $f = i + 1, i + 2 \dots i + d$ .*

**Proof.** Since we have by hypothesis  $|\mathcal{L}_m| \geq 2$  in  $suf_i$  (by Lemma 3 and the assumption that  $m$  is gapped), then we also have  $|\mathcal{L}_{m'}| \geq 2$  in  $suf_{i+1}$ . We show first that  $m'$  fulfills the conditions of maximality.

Observe first that  $m'$  must be maximal in composition, for if it were possible to specify more solid characters in the occurrences of  $m'$  without altering the list  $\mathcal{L}_{m'}$  in  $\text{suf}_{i+1}$ , then this would apply in particular to all positions preceded at a distance of  $d$  characters by character  $\sigma$ , whence  $m$  would subsume that change. By the same argument, it must be also maximal in length, since assuming that this is not the case then there must be a motif  $m''$  implying  $m'$  and such that  $\bar{m} = \sigma d' m''$  implies  $m$ , which contradicts the assumption that  $m \in \mathcal{B}_i$ . The claim is thus established if  $m'$  is irredundant. On the other hand, if  $m'$  is redundant, then, since  $m'$  is maximal, the list  $\mathcal{L}_{m'}$  must be the union of lists of other motifs of which  $m'$  is a prefix. In particular, there must be two motifs  $\bar{m}$  and  $\hat{m}$  implying  $m'$  and such that one occurrence of  $m$  is found at  $d$  characters from an element of the list  $\mathcal{L}_{\hat{m}}$  while the other is found analogously per  $\mathcal{L}_{\bar{m}}$ . ■

Lemma 4 gives us a general format for motifs at  $i$ . We examine next the impact of the introduction of a new motif in  $\mathcal{B}_i$  on the set  $\mathcal{B}_{i+1}$ . Clearly, a motif of  $\mathcal{B}_{i+1}$  is propagated to  $\mathcal{B}_i$  as long as its irredundancy was preserved. Therefore, we are interested specifically in the conditions that might cause a motif of  $\mathcal{B}_{i+1}$  to lose its irredundancy. Recall that a motif  $\hat{m}$  of  $\text{suf}_i$  is redundant if its list can be deduced by the union of the lists of other, irredundant motifs. This means that *every* occurrence of  $\hat{m}$  in  $\text{suf}_i$  must be extended by some motif of  $\mathcal{B}_i$ . Since this condition was not met in  $\mathcal{B}_{i+1}$ , this requires that the new motifs extend the occurrence of  $\hat{m}$  at  $i$  and also every other occurrence of  $\hat{m}$  not extended by any other motif of  $\mathcal{B}_{i+1}$  (in the following, such occurrences of  $\hat{m}$  will be called *maximal*—but note that here maximality refers to an occurrence, not a motif—and Motif  $\hat{m}$  will be said to be *exposed*). We also say that the new motifs *obliterate*  $\hat{m}$ . Thus, while  $i$  varies from  $n$  to 1, a motif like  $\hat{m}$  can be discovered and obliterated a number of times. At the generic stage  $i$ ,  $\hat{m}$  will be a term of  $\mathcal{B}_{i+1}$  only if at least one of its occurrences is maximal. As for the motifs  $m$  in the preceding lemma, these motifs might themselves have been discovered a number of times before and now again at  $i$ , in which case precisely one of their occurrences in  $\text{suf}_i$  must be maximal, and this would have to be the occurrence at  $i$ . Note, however, that every time a list  $\mathcal{L}_m$  is created, this must recapture all previous occurrences of  $m$ .

In loose terms, this discussion suggests that motif occurrences are arranged in a partial order based on the containment or prefix relationship. Along these lines, any new candidate motif  $m$  at  $i$  must be considered from two complementary perspectives, respectively entailing the motifs that  $m$  extends and those that are instead extensions of it. Since the existence of any motif in this second class precludes further consideration of  $m$ , this condition should be checked first. Note that an extension of  $m$  might be any motif in  $\mathcal{B}_i$ . As for those in the first class, each one of them risks being dislodged from  $\mathcal{B}_i$  if the new motifs  $m$ , together with other motifs of  $\mathcal{B}_{i+1}$  that stay in  $\mathcal{B}_i$ , prove capable of extending its every occurrence in  $\text{suf}_i$ . On the other hand, notice that the expulsion of a motif  $\hat{m}$  from the basis on grounds of redundancy does not affect the irredundancy of the motifs whose lists are now tributaries to the list of  $\mathcal{L}_{\hat{m}}$ .

**Lemma 5.** *Let  $m \notin \mathcal{B}_{i+1}$  be a maximal motif and a prefix of  $\text{suf}_i$ ; then  $m \in \mathcal{B}_i$  if and only if the occurrence of  $m$  at  $i$  is maximal (i.e., if  $m$  becomes exposed in  $\text{suf}_i$ ).*

**Proof.** Observe that by Lemma 3 there must be at least two occurrences of  $m$  in  $\text{suf}_i$ , hence at least one occurrence of  $m$  in  $\text{suf}_{i+1}$ . We consider first the case where there is exactly one occurrence of  $m$  in  $\text{suf}_{i+1}$ . Assume for a contradiction that the occurrence of  $m$  at  $i$  is not maximal, and let  $\hat{m}$  be an element of  $\mathcal{B}_i$  of which  $m$  is a prefix. If  $\hat{m}$  also covers the occurrence of  $m$  in  $\text{suf}_{i+1}$ , this contradicts the hypothesis that  $m \in \mathcal{B}_i$ . Since  $\hat{m}$  must occur twice in  $\text{suf}_i$ , however, then  $m$  must appear at least three times, which again contradicts the assumption.

Assume now that  $m$  occurs more than once in  $\text{suf}_{i+1}$ . Since  $m \in \mathcal{B}_i$  but  $m \notin \mathcal{B}_{i+1}$ , however, then each occurrences of  $m$  in  $\text{suf}_{i+1}$  must be implied by an occurrence of some other motif. Consider any such motif, say,  $\hat{m}$ . The assertion follows if  $\hat{m} \in \mathcal{B}_i$ , since by the hypothesis of  $m$  having a maximal occurrence at  $i$  we have that  $\hat{m}$  cannot have an occurrence at  $i$ . On the other hand, if  $\hat{m} \notin \mathcal{B}_i$ , then every occurrence of  $\hat{m}$  in  $\text{suf}_{i+1}$  is covered by some motif of  $\mathcal{B}_i$  which was not in  $\mathcal{B}_{i+1}$ . But clearly any such motif must have an occurrence at  $i$  and thus cover the occurrence of  $m$  at  $i$ . This would make  $m$  redundant and contradicts the assumption that  $m \in \mathcal{B}_i$ . ■

Our discussion is summarized in the following statement.

**Theorem 3.** *The set  $\mathcal{B}_i$  is the union of the exposed motifs in  $\text{suf}_i$ .*

Lemma 5 and Theorem 3 suggest a general pattern of approach to the discovery of irredundant motifs. Along these lines, the motifs of  $\mathcal{B}_i$  can be produced following two steps.

1. **FIND:** Find all motifs—i.e., whether included already in  $\mathcal{B}_{i+1}$  or newly discovered—having occurrences at  $i$ . In the first case, there is need only to extend the corresponding lists; in the second, they have to be (re-)threaded. Let  $\hat{\mathcal{B}}$  be the set that results from subjecting  $\mathcal{B}_{i+1}$  to this treatment.
2. **DISCARD:** Discard all motifs of  $\hat{\mathcal{B}} \cup \mathcal{B}_{i+1}$  obliterated by the new motifs in  $\hat{\mathcal{B}}$ . This entails checking that the newly introduced motifs or motif occurrences actually extend all occurrences of the obliterated ones. The result is  $\mathcal{B}_i$ .

The implementation of this paradigm for general  $k$  is demanding. For  $k = 2$ , however, Theorems 1 and 2 provide a handle. In view of Theorem 1, Lemma 5 states that the when  $k = 2$  new motifs  $m$  in  $\mathcal{B}_i$  may be found by taking the meet of  $\text{suf}_i$  with any suffix  $\text{suf}_j$  with  $j > i$  and then checking which one of these is an exposed motif in  $\text{suf}_i$ . By Theorem 2, we can also state the following.

**Theorem 4.** *The cardinality of the union of the sets  $\mathcal{B}_i$ 's ( $i = 1, 2, \dots, n$ ) is  $O(n^2)$ .*

From now on, we assume implicitly that  $k = 2$  and refer to 2-motifs simply by “motif,” unless specified otherwise.

### 3. THE TOP LEVEL AND ITS ANALYSIS

We are now ready to outline a presentation and analysis of the top level of our procedure. As mentioned, the outermost cycle iterates on the index  $i$  from  $n$  to 1. It is convenient to visualize the computation as centered on the match matrix  $M$ , which is filled row-by-row bottom-up during the main cycle, a new row being introduced for every new suffix. Thus, prior to considering  $\text{suf}_i$ , the matrix is filled from row  $i + 1$  to row  $n$ . Recall that each match in the matrix represents a pairwise suffix meet, and let now specifically  $m^{[k,l]}$  be the one such meet starting at match  $M[k, l]$ .

**Lemma 6.** *Assume that  $\mathcal{L}_{m^{[k,l]}}$  is available for all  $k > i$  and  $l > k$ . Then  $\mathcal{L}_{m^{[k,l]}}$  can be computed for all  $k \geq i$  and  $l > k$  in overall  $O(n^2)$  time.*

**Proof.** We leave base and border cases for exercise and concentrate on the inductive step. For this, we first take care of initializing all newly introduced suffix meets, i.e., those starting with a match on row  $i$ . Specifically, we show that, assuming  $\mathcal{L}_{m^{[i+1,j]}}$  is available for all  $j = i + 2, \dots, n$ ,  $\mathcal{L}_{m^{[i,j]}}$  can be computed for all  $j = i + 1, \dots, n$  in overall  $O(n^2)$  time. Consider the generic entry  $M[i, j]$  and assume this is a match. Let  $M[i + d, j + d]$  be the closest diagonal match, and let  $\mathcal{L}_{m^{[i+d,j+d]}} = (l_1, l_2, \dots, l_p)$  the corresponding list. Then  $\mathcal{L}_{m^{[i,j]}}$  is the subset of  $(l_1 - d, l_2 - d, \dots, l_p - d)$  that corresponds to occurrences of the character  $s_i$ . This set is trivially extracted in linear time while scanning  $\mathcal{L}_{m^{[i+d,j+d]}}$  on  $s$ .

We must now update the lists of entries on rows of index larger than  $i$ . At the beginning, we have by hypothesis that the list of each matching entry  $M[k, l]$  correctly reports all occurrences of  $m^{[k,l]}$  in  $\text{suf}_{i+1}$ . Hence, we need only to check whether  $m^{[k,l]}$  has an occurrence at  $i$  and in that case add  $i$  to the list. We will do so by scanning the matrix  $M$  bottom-up and spending constant time on the update of the list associated with each entry. At the generic step, we have updated all rows up to  $k + 1$  and are considering row  $k$ . Let  $M[k + d, l + d]$  be the closest diagonal match to  $M[k, l]$ . Then clearly  $i$  must be added to the list of  $m^{[k,l]}$  if and only if  $s_i$  matches the first character of  $m^{[k,l]}$  and  $i + d$  is in the list  $\mathcal{L}_{m^{[k+d,l+d]}}$ . Each condition is testable in constant time. ■

For instance, assuming that only the rows from row 6 on had been computed in the matrix of Fig. 1, we would have, for motif  $ab$ ,  $\mathcal{L}_{m^{[6,23]}} = (6, 9, 12, 14, 16, 21, 23)$ . When row 5 is introduced, the new meet  $M[5, 22]$ , corresponding to motif  $bab$ , has list  $\mathcal{L}_{m^{[5,22]}} = (5 = 6 - 1, 13 = 14 - 1, 22 = 23 - 1)$ . Lemma 6 ensures that the computation of all  $\mathcal{L}$ -lists involved in our process can be afforded in overall cubic time and that, moreover, whenever a new meet is introduced, the list of the newcomer is synthesized in linear time.

We can now concentrate on the management of the procedures FIND and DISCARD at the generic iteration. To avoid clutter in our notation, it is useful to think of the elements of any of the  $\mathcal{B}$ -sets below interchangeably as a set of motifs as well as specific motif occurrences (typically, the occurrence at  $i$ ), whenever this causes no confusion. Towards this end, we stipulate the following.

**Invariant 1.** *In  $\mathcal{B}_{i+1}$ , each motif is positioned at (represented by) its leftmost occurrence in  $s$ .*

Recall that the task of FIND is to find all motifs—i.e., whether included already in  $\mathcal{B}_{i+1}$  or newly discovered—having (perhaps maximal) occurrences at  $i$ , so as to produce the intermediate set  $\hat{\mathcal{B}}$ . Accordingly, we consider FIND composed of the two constituents FINDOLD and FINDNEW.

We examine FINDOLD first. This procedure expects to be handed the list of entries in the matrix  $M$  that corresponds to the motifs in  $\mathcal{B}_{i+1}$ .

Let us call  $\bar{\mathcal{B}}$  the subset of  $\mathcal{B}_{i+1}$  which will result from FINDOLD. Having initialized  $\bar{\mathcal{B}}$  to be the empty set, the action of the procedure consists of matching  $suf_i$  against the motifs of  $\mathcal{B}_{i+1}$ , one at a time. Whenever motif  $m$  is fully matched,  $i$  is inserted in  $\mathcal{L}_m$ , and  $m$  is removed from  $\mathcal{B}_{i+1}$  and inserted in  $\bar{\mathcal{B}}$ . At this point,  $\bar{\mathcal{B}}$  contains all old motifs of which a new occurrence at  $i$  was discovered. Note that, as a by-product, this propagates Invariant 1 for all old motifs that will eventually stay in  $\mathcal{B}_i$ . The remaining details of the procedure are straightforward.

**Lemma 7.** *FINDOLD takes  $O(n^2)$  time.*

**Proof.** There are  $O(n)$  motifs in a base, each requiring  $n$  comparisons to be matched against  $suf_i$ , hence a total cost  $O(n^2)$ . The cost of every other operation is absorbed in these bounds. ■

Before proceeding further, we need to take care of the possibility that some old motifs become redundant as a result of the discovery of a new occurrence at  $i$  of other old motifs (i.e., of the old motifs now in  $\bar{\mathcal{B}}$ ). The following lemma proves that no such situation may arise, thereby establishing the correctness of this part of the procedure.

**Lemma 8.** *The occurrence at  $i$  of an old motif cannot obliterate another old motif.*

**Proof.** Let  $m$  be the old motif occurring at  $i$ . Note first that the occurrence of an old motif at  $i$  cannot obliterate another old motif similarly occurring at  $i$  (or we would have that, already in  $suf_{i+1}$ , all occurrences of this second motif were contained in occurrences of some other motif of  $\mathcal{B}_{i+1}$ ) prior to the  $i$ th iteration. Assume then that the motif being obliterated does not have an occurrence at  $i$ . It is clear that only the leftmost occurrence of an old motif could be obliterated by  $m$  in this way. Letting  $m = \sigma dm'$ , then clearly any old motif  $\bar{m}$  in  $suf_{i+1}$  now implied by  $m$  must have been implied by  $m'$ , too. Since the implication of  $\bar{m}$  by  $m$  is all is needed to obliterate  $\bar{m}$ , then all other occurrences of  $\bar{m}$  in  $suf_{i+1}$  were also already implied by some motif in  $\mathcal{B}_{i+1}$  prior to iteration  $i$ . The question is then why  $m'$ —which being a suffix of  $m$  must appear at least twice in  $suf_{i+1}$ —was not itself a motif of  $\mathcal{B}_{i+1}$ . Clearly,  $m'$  must be redundant. But then so is  $\bar{m}$ , since any motif that implies  $m'$  will imply  $\bar{m}$  as well. ■

It is now the task of FINDNEW to transform  $\bar{\mathcal{B}}$  into  $\hat{\mathcal{B}}$ , the set of old and new motifs occurring at  $i$ . In view of the structure of the match matrix  $M$  and Lemma 6, we can let FINDNEW start by comparing the first symbols of  $suf_i$  and  $suf_j$  pairwise for all  $j > i$ , thereby generating the family of new meets  $\tilde{\mathcal{B}} = \{m^{[i,j]}\}$  and the associated lists (note that  $\tilde{\mathcal{B}}$  contains only the subdiagonals of Matrix  $M$  that start with a match on row  $i$ ). FINDNEW also updates all other lists in the matrix. Lemma 6 guarantees that this part of the work is bounded by  $O(n^2)$  time, so that we do not need to belabor this further.

Luckily, it is not necessary to ask whether some among the motifs in  $\tilde{\mathcal{B}}$  extend other such motifs.

**Lemma 9.** *No motif in  $\tilde{\mathcal{B}}$  can obliterate another motif in  $\tilde{\mathcal{B}}$ .*

**Proof.** Assume that  $m$  is one of the motifs obliterated by the other motifs in  $\tilde{\mathcal{B}}$  and that  $m$  results from the meet of  $suf_i$  with  $suf_j$ . This means that each occurrence in  $\mathcal{L}_m$  is extended by some occurrence of

another motif. Take, in particular, the occurrence of  $m$  at  $j$ , and let  $m'$  be the motif of  $\tilde{\mathcal{B}}$  extending  $m$  at  $j$ . Then  $m'$  itself occurs both at  $j$  and  $i$ ; i.e.,  $m'$  implies the meet  $m$  of  $\text{suf}_i$  and  $\text{suf}_j$ , a contradiction. ■

We can list the following.

**Lemma 10.** *FINDNEW, whence also FIND, takes  $O(n^2)$  time.*

**Proof.** From the above discussion. ■

This concludes our outline of Step 1, and we now turn to Step 2 and the procedure DISCARD. Recall that the task of DISCARD is to eliminate from  $\hat{\mathcal{B}} \cup \mathcal{B}_{i+1}$  all motifs that are extended by some new motif at  $i$ . Recall also that the motifs extended at  $i$  only by some old motifs in  $\mathcal{B}_{i+1}$  must have some exposed occurrence somewhere in  $\text{suf}_{i+1}$  which stays exposed also in  $\text{suf}_i$ . Such motifs are thus unaffected. To recapitulate, *at this point* we have the following sets, where motifs within each set are mutually maximal:

- the subset  $\bar{\mathcal{B}}$  of  $\hat{\mathcal{B}}$  that consists of old motifs having an occurrence at  $i$ ;
- the set  $\mathcal{B}_{i+1}$ , currently consisting of old motifs not having an occurrence at  $i$ ;
- the subset  $\tilde{\mathcal{B}}$  of  $\hat{\mathcal{B}}$  that consists of the meets of  $\{m_{[i,j]}\}$  (which have thus an occurrence at  $i$ ).

While we have seen that no element in a set can obliterate another element in the same set, we still need to examine how elements in one set may possibly obliterate the elements of another set. Since, in addition,  $\mathcal{B}_{i+1}$  and  $\bar{\mathcal{B}}$  are mutually compatible (see Lemma 8), we are left with the comparisons between these sets and  $\tilde{\mathcal{B}}$ . The first observation to be made follows in the lines of Lemma 9.

**Lemma 11.** *For any  $i$ , no element of  $\mathcal{B}_i$  with an occurrence at  $i$  can extend a meet  $m_{[i,j]}$ .*

**Proof.** By definition, the meet of  $\text{suf}_i$  with any other suffix is a maximal motif. ■

In conclusion, no element of  $\mathcal{B}_{i+1} \cup \bar{\mathcal{B}}$  can obliterate an element of  $\tilde{\mathcal{B}}$ , and only the reciprocal is possible: that is, new motifs may dislodge an old one, provided they extend all occurrences of the latter in  $\text{suf}_i$ . We could check this by fully matching the meets in  $\tilde{\mathcal{B}}$  against the old motifs one by one, or *vice versa*. However, there is a quadratic number of motif pairs to be considered. At a cost of  $O(n)$  character comparisons per pair, this would result in  $O(n^3)$  time for this iteration,  $O(n^4)$  in total. Fortunately, Lemma 11 tells us that such an exhaustive matching is not needed: in intuitive terms, since an old motif cannot contain a new one, then the character-by-character comparison of a new and an old motif can stop at the first mismatch. In fact, such a mismatch could mean only that the old motif is contained in the new one.

To make the argument more precise, assume that the motifs in  $\tilde{\mathcal{B}}$ —each represented as a string on  $\Sigma \cup \text{"."}$ —are stored in a compact trie. Assume further that, beginning with the  $\mathcal{L}$  lists at the bottom of the trie and proceeding bottom-up, we produce at each node the list of all occurrences of the motifs which are found in the subtree rooted at that node.

Note that the motifs in  $\tilde{\mathcal{B}}$  are positioned at  $i$ , while old motifs are positioned at some  $i' \geq i$ . We discuss first the matching of a motif  $m$  positioned at  $i$  (i.e., an old motif currently in  $\tilde{\mathcal{B}}$ ). To match  $m$ , we follow the path in the trie in response to consecutive characters in  $m$ . By what was said, to a solid character of  $m$  there cannot correspond a “.” in the trie, but the reverse can happen. To deal with the general case, assume that this process stops after comparing the  $k$ -th character. That is, a don’t care in  $m$  “mismatched” with a solid character in the trie. Let  $\mu$  be the node of the trie which is found at the end of the arc where this mismatch occurred. Then all motifs that are found in the subtree rooted at  $\mu$  extend  $m$ . We merge  $\mathcal{L}_m$  with the list precomputed at  $\mu$  and, in case these lists coincide, mark  $m$  “redundant” and eliminate it from  $\tilde{\mathcal{B}}$ .

Finally, we consider the comparisons between the motifs that are still left in  $\mathcal{B}_{i+1}$  and those in  $\tilde{\mathcal{B}}$ . Note that if the occurrence of  $m$  at  $i$  obliterates a motif  $m'$  having its leftmost occurrence at  $i' = i + d$ , then the occurrence at  $i'$  of a suffix of  $m$ , a meet in itself, would have obliterated already  $m'$  during iteration  $i'$ . Thus, the only case to be addressed concerns the possibility that a motif in  $\tilde{\mathcal{B}}$  is actually a lengthened version of one in  $\mathcal{B}_{i+1}$ . That is,  $m \in \tilde{\mathcal{B}}$  can be written as  $vm'$  where  $m' \in \mathcal{B}_{i+1}$ . It is easily seen that a motif such as  $m$  is spotted and its list synthesized (simply inherited from that of  $m'$ ) as a by-product of the construction of Lemma 6. This concludes the description of our construction.

**Lemma 12.** DISCARD takes  $O(n^2)$  time.

**Proof.** The construction of the trie of  $O(n)$  motifs, each at most  $n$  characters long, takes  $O(n^2)$  time by straightforward methods. The trie has at most  $O(n)$  leaves and thus  $O(n)$  branching internal nodes. Merging the  $\mathcal{L}$  lists at each node takes linear time, hence  $O(n^2)$  time in total. (Perform the fusions on a binary tree equivalent to the trie, in which each original node of the trie has been transformed into a small binary tree: the number of leaves does not change, and the same holds for the bound on the branching nodes.) Subsequent matches of each motif take  $O(n)$  time each, and so does the list merge involved in checking each motif. All other operations are absorbed in this bound. ■

Taking the union of the  $\mathcal{B}$ 's at the outset yields  $\mathcal{B}_i$ . We have thus established the following.

**Theorem 5.** *The set of irredundant 2-motifs in a string of  $n$  characters can be computed incrementally in  $O(n^3)$  time.*

## CONCLUSION

We examined the problem of extracting maximal, irredundant motifs that occur at least twice in a string. The notion of irredundancy for motifs with at least  $k$  occurrences for general  $k$  had been introduced and studied in previous work (Parida *et al.*, 2000). Here, we give a linear upper bound for the number of irredundant 2-motifs in a string. It is easy to adapt arguments such as that of Theorem 1 so that they generalize to  $k$ -motifs, however, the extension of our constructions to the general case seems more involved.

The motif discovery paradigms explored here are based on a close scrutiny of the process of iterative update of the set of motifs in a string under consecutive unit symbol extensions of the string itself. This line of attack exposes characterizations of possible intrinsic interest for the set of motifs in a base and also for the set of motif occurrences at any given position of the input string.

It is possible that these results lead to a family of algorithms for the extraction of irredundant motifs with a typical performance better than previously available, in particular, susceptible to graceful specialization to cases of interest involving controlled numbers of gaps, density constraints, etc. Additional open issues concern the derivation of lower-bound arguments for the number of motifs and related aspects of algorithmic complexity.

Finally, while the linear bound on irredundant 2-motifs brings about remarkably synthetic descriptors for the motif structure of a string, this class of irredundant motifs may not exhaust the significant motifs in any given context. Along these lines, one may wish to design efficient ways to derive more or less restricted classes of redundant motifs from the irredundant ones.

## ACKNOWLEDGMENTS

Work by the first author was performed in part while on leave at IBM T.J. Watson Center. Additional support was provided in part by NATO Grant PST.CLG.977017, by the Italian Ministry of University and Research under the National Projects "Bioinformatics and Genomics Research" and FIRB RBNE01KNFP, by the Research Program of the University of Padova, and by an IBM Faculty Partnership Award.

## REFERENCES

- Apostolico, A. 2003. Pattern discovery and the algorithmics of surprise. *Proc. NATO ASI on Artificial Intelligence and Heuristic Methods for Bioinformatics*, 111–127.
- Apostolico, A., and Atallah, M.J. 2002. Compact recognizers of episode sequences. *Information and Computation* 174, 180–192.
- Apostolico, A., and Galil, Z. eds. 1997. *Pattern Matching Algorithms*, Oxford University Press, New York.
- Blumer, A., Blumer, J., Ehrenfeucht, A., Haussler, D., Chen, M.T., and Seiferas, J. 1985. The Smallest Automaton Recognizing the Subwords of a Text. *Theoret. Comput. Sci.* 40, 31–55.

- Parida, L., Rigoutsos, I., Floratos, A., Platt, D., and Gao, Y. 2000. Pattern discovery on character sets and real-valued data: Linear bound on irredundant motifs and polynomial time algorithms. *Proc. 11th ACM-SIAM Symposium on Discrete Algorithms (SODA 2000)*, 297–308.
- Rigoutsos, I., Floratos, A., Parida, L., Gao, Y., and Platt, D. 2000. The emergence of pattern discovery techniques in computational biology. *J. Metabolic Engineering* 2(3), 159–177.
- Wang, J., Shapiro, B., and Shasha, D. eds. 1999. *Pattern Discovery in Biomolecular Data: Tools, Techniques, and Applications*, Oxford University Press, Oxford, UK.
- Waterman, M. 1995. *Introduction to Computational Biology*, Chapman and Hall, London.

Address correspondence to:

*Alberto Apostolico*  
*Department of Computer Sciences*  
*Purdue University*  
*Computer Sciences Bldg.*  
*West Lafayette, IN 47907*

*E-mail: axa@cs.purdue.edu*

### **AU1**

**Provide “Key words” here.**

### **AU2**

**Reference citations okay as added per electronic file? Only Refs. [4] and [5] were shown on manuscript. Okay as now shown? Or should these citations/Refs. be deleted?**

### **AU3**

**Change to “interspersed” correct?**