MADMX: A Novel Strategy for Maximal Dense Motif Extraction

Roberto Grossi[1], Andrea Pietracaprina[2,⋆], Nadia Pisanti[1], Geppino Pucci[2,⋆], Eli Upfal[3,⋆,⋆⋆], and Fabio Vandin[2,⋆,⋆⋆⋆]

[1] Dipartimento di Informatica, Università di Pisa, Pisa, Italy.
{grossi,pisanti}@di.unipi.it
[2] Dipartimento di Ingegneria dell'Informazione, Università di Padova, Padova, Italy.
{capri,geppo,vandinfa}@dei.unipd.it
[3] Department of Computer Science, Brown University, Providence RI, USA.
eli@cs.brown.edu

Abstract We develop, analyze and experiment with a new tool, called MADMX, which extracts frequent motifs, possibly including don't care characters, from biological sequences. We introduce *density*, a simple and flexible measure for bounding the number of don't cares in a motif, defined as the ratio of solid (i.e., different from don't care) characters to the total length of the motif. By extracting only *maximal dense motifs*, MADMX reduces the output size and improves performance, while enhancing the quality of the discoveries. The efficiency of our approach relies on a newly defined combining operation, dubbed *fusion*, which allows for the construction of maximal dense motifs in a bottom-up fashion, while avoiding the generation of nonmaximal ones. We provide experimental evidence of the efficiency and the quality of the motifs returned by MADMX.

1 Introduction

The discovery of frequent patterns (*motifs*) in biological sequences has attracted wide interest in recent years, due to the understanding that sequence similarity is often a necessary condition for functional correlation. Among other applications, motif discovery proves an important tool for identifying regulatory regions and binding sites in the study of functional genomics. From a computational point of view, a major complication for the discovery of motifs is that they may feature some sequence variation without loss of function. The discovery process must therefore target *approximate motifs*, whose occurrences are similar but not necessarily identical. Approximate motifs are often modeled through the use of

⋆ Support for these authors was provided, in part, by the European Union under the FP6-IST/IP Project AEOLUS.
⋆⋆ Supported in part by NSF awards IIS-0325838 and DMI-0600384, and ONR Award N000140610607.
⋆⋆⋆ Contact Author. This work was done, in part, while the author was visiting the Department of Computer Science of Brown University.

the *don't care* character in certain positions, which is a wild card matching all characters of the alphabet, called *solid characters* [10].

Finding interesting approximate motifs is computationally challenging. As the number of don't cares increases and/or the minimum frequency threshold decreases, the output may explode combinatorially, even if the discovery targets only maximal motifs—a subset of the motifs which implicitly represents the complete set. Moreover, even when the final output is not too large, partial data during the inference of target motifs might lead to memory saturation or to extensive computation during the intermediate steps.

A large body of literature in the last decade has dealt with efficient motif discovery [9,3,12,4,16,8,6,5,2], and an excellent survey of known results can be found in the book [10]. In order to alleviate the computational burden of motif extraction and to limit the output to the most promising or interesting discoveries, some works combine the traditional use of a frequency threshold with restrictions on the flexibility of the extracted motifs, often captured by limitations on the number of occurring don't cares.

In a recent work, Apostolico et al. [2] study the extraction of *extensible motifs*, comprising standard don't cares and extensible wild cards. The latter are spacers of variable length that can take different size (within pre-specified limits) in each occurrence of the motif. An efficient tool, called VARUN, is devised in [2] for extracting all maximal extensible motifs (according to a suitable notion of maximality defined in the paper) which occur with frequency above a given threshold σ and with upper limits D on the length of the spacers. VARUN returns the extracted motifs sorted by decreasing z-score, a widely adopted statistical measure of interestingness. The authors demonstrate the effectiveness of their approach both theoretically, by proving that each maximal motif features the highest z-score within the class of motifs it represents, and experimentally, by showing that the returned top-scored motifs comprise biologically relevant ones when run on protein families and DNA sequences .

A slightly more general way of limiting the number of don't cares in a motif has been explored in [13]. The authors define $\langle L, W\rangle$ motifs, for $L \leq W$, where at least L solid characters must occur in each substring of length W of the motif. They propose a strategy for extracting $\langle L, W\rangle$ motifs which are also maximal, although their notion of maximality is not internal to the class of $\langle L, W\rangle$ motifs. As a consequence, the algorithm is not complete, since it disregards all those $\langle L, W\rangle$ motifs that are subsumed by a maximal non-$\langle L, W\rangle$ one.

Our results. Our work focuses on the discovery of *rigid motifs*, which contain blocks of solid characters (solid blocks) separated by one or more don't cares. We propose a more general approach for controlling the number of don't cares in rigid motifs. Specifically, we introduce the notion of *dense motif*, a frequent pattern where the fraction of solid characters is above a given threshold. Our density notion is more flexible and general than the one considered in [10,2], since it allows for arbitrarily long runs of don't cares as long as the fraction of solid characters in the pattern is above the threshold. We define a natural notion of *maximality* for dense patterns and devise an efficient algorithm, called

MADMX (pronounced *Mad Max*), which performs complete MAximal Dense Motif eXtraction from an input sequence, with respect to user-specified frequency and density thresholds.

The key technical result at the core of our extraction strategy is a closure property which affords the complete generation of all maximal dense motifs in a breadth-first fashion, through an *apriori*-like strategy [1], starting from a relatively small set of solid blocks, and then repeatedly applying a suitable combining operator, called *fusion*, to pairs of previously generated motifs. In this fashion, our strategy avoids the generation and consequent storage of intermediate patterns which are not in the output set, which ensures time and space complexities polynomial in the combined size of the input and the output.

We performed a number of experiments on MADMX to assess the biological significance of maximal dense motifs and to compare MADMX against its most recent and close competitor VARUN. For the first objective, we used MADMX to extract maximal dense motifs from a number of human DNA fragments. We compared the output set against those in `RepBase` [7], the largest repository of repetitive patterns for eukaryotic species, using REPEATMASKER [15], a popular tool for masking repetitive DNA. The experiments show that all of our returned motifs are occurrences of patterns in `RepBase`, and *fully* characterize the family of SINE/ALU repeats (and partially the LINE/L1 family). This provides evidence that the notion of density, when applied to rigid motifs, captures biological significance.

Next we compared the z-score performance of MADMX and VARUN. We ran both algorithms on several families of DNA fragments, limiting VARUN to the generation of rigid motifs and setting the parameters so as to obtain comparable output sizes, with motifs listed by decreasing z-score. The experiments show that the top-m highest-ranking motifs returned by MADMX almost always feature higher z-scores than the corresponding top-m ones returned by VARUN, even for large values of m, with only a modest increase in running time, which may be partly due to the fact that coding of MADMX is yet to be optimized. In fairness, we must remark that VARUN deals also with extensible motifs while MADMX only targets rigid motifs.

The paper is organized as follows. In Section 2 several technical definitions and properties of motifs with don't cares are given. Section 3 proves the closure property at the base of MADMX and provides a high-level description of the algorithm. In Section 4, the experimental validation of MADMX is presented.

2 Preliminary Definitions and Properties

Let Σ be an alphabet of m characters and let $s = s[0]s[1]\dots s[n-1]$ be a string of length n over Σ. We use $s[i\dots j]$ to denote the substring $s[i]\ s[i+1]\cdots s[j]$ of s, for $i \leq j$. Characters in Σ are also called *solid characters*. We use $\circ \notin \Sigma$ to denote a distinguished character called *wild card* or *don't care* character. Let ϵ denote the empty string. A *pattern* x is a string in $\{\epsilon\} \cup \Sigma \cup \Sigma(\Sigma \cup \{\circ\})^*\Sigma$.

However, whenever necessary, we will assume that patterns are implicitly padded to their left and right with arbitrary sequences of don't care characters.

Given two patterns x, y we say that y is *more specific* than x, and write $x \preceq y$, iff for every $i \geq 0$ either $x[i] = y[i]$ or $x[i] = \circ$. Given two patterns x, y we say that x *occurs in* y *at position* ℓ iff $x \preceq y[\ell \dots \ell + |x| - 1]$: we also say that y *contains* x. For a string s, the *location list* $\mathcal{L}_x$ of a pattern x in s is the complete set of positions at which x occurs in s. We refer to $f(x) = |\mathcal{L}_x|$ as the *frequency* of pattern x in s. (Note that $f(\epsilon) = n$.) As in [16], the *translated representation* of the location list $\mathcal{L}_x = \{l_0, l_1, l_2, \dots, l_k\}$ is $\tau(\mathcal{L}_x) = \{l_1 - l_0, l_2 - l_0, \dots, l_k - l_0\}$. Given two patterns x, y, we say that y *subsumes* x in s if $f(x) = f(y)$ and y contains x. As a consequence, if y subsumes x then $\tau(\mathcal{L}_x) = \tau(\mathcal{L}_y)$. A pattern x is *maximal* if it is not subsumed by any other pattern y. (We observe that this notion of maximality coincides with that of [12].) Given a pattern x, its *maximal extension* $\mathcal{M}(x)$ is the maximal pattern that subsumes x, which can be shown to be unique [12].

In what follows, we call *solid block* a string in Σ^+ and a *don't care block* a string in $\circ^+$. Furthermore, given a pattern x, $\mathrm{dc}(x)$ denotes the number of don't care characters contained in x.

Definition 1. *The* density $\delta(x)$ *of* x *is:* $\delta(x) = 1 - dc(x)/|x|$. *Given a (density) threshold* ρ, $0 < \rho \leq 1$, *we say that a pattern* x *is dense if* $\delta(x) \geq \rho$.

Note that a solid block is a dense pattern with respect to every threshold ρ.

It is reasonable to concentrate the attention on dense patterns that are not subsumed by any other dense pattern, since they are the most interesting dense representatives in the equivalence classes induced by "sharing" the same translated representation; these representatives are defined below.

Definition 2. *A dense pattern* x *is a* maximal dense pattern *in* s *if it is not subsumed by any other dense pattern* $x' \neq x$.

Observe that a maximal dense pattern x needs not be a maximal pattern in the general sense, since $\mathcal{M}(x)$ might be a nondense pattern. However, every dense pattern x is subsumed by *at least* one maximal dense pattern. In fact, all of the maximal dense patterns that subsume x are dense substrings of $\mathcal{M}(x)$, namely, those that contain x and are not substrings of any other dense substring of $\mathcal{M}(x)$. We want to stress that there might be several maximal dense patterns that subsume x. As an example, for $\rho = 2/3$, the dense pattern $x = \mathtt{B}$ in the string $S = \mathtt{AdBeCfAgBhC}$ is subsumed by maximal dense patterns $\mathtt{A} \circ \mathtt{B}$ and $\mathtt{B} \circ \mathtt{C}$, while $\mathcal{M}(x) = \mathtt{A} \circ \mathtt{B} \circ \mathtt{C}$ is not dense.

Definition 3. *Given a frequency threshold* σ *and a density threshold* ρ, *a pattern* x *is a* dense maximal motif *in* s *if* x *is a maximal dense pattern in* s *with respect to* ρ, *and* $f(x) \geq \sigma$. *A dense maximal motif for* $\rho = 1$ *is also referred to as* maximal solid block.

Problem of interest. We are given an input string s, a frequency threshold σ, and a density threshold ρ. Find all the maximal dense motifs in s.

In the rest of the paper, we will omit referencing the input string s when clear from the context. An important property of maximal dense patterns, which we will exploit in our mining strategy, is that all of their solid blocks are maximal solid blocks. This property is stated in the following proposition whose proof, omitted for brevity, extends a similar result holding for arbitrary maximal patterns [16,11].

Proposition 1. *Let x be a maximal dense pattern with respect to a density threshold ρ, and let $b = x[i \dots j]$ be a solid block in x such that $x[i-1] = x[j+1] = \circ$ and $j \geq i$. Then, b is a maximal solid block.*

3 An Algorithm for MAximal Dense Motif eXtraction

In this section we describe our algorithm, called MADMX (pronounced *Mad Max*), for MAximal Dense Motif eXtraction. The algorithm adopts a breadth-first *apriori*-like strategy [1], similar in spirit to the one developed in [2], using maximal solid blocks as building blocks by Proposition 1. MADMX operates by repeatedly combining together, in a suitable fashion, pairs of maximal dense motifs, and extracting from the combinations less frequent maximal dense motifs.

A key notion for the algorithm, underlying the aforementioned combining operations, is the *fusion* of characters/patterns.

Definition 4. *Given three characters $c, c_1, c_2 \in \Sigma \cup \{\circ\}$, we say that c is the fusion of c_1 and c_2, and write $c = c_1 \bigtriangledown c_2$, if one of the following holds:*

1. $c = c_1 = c_2$;
2. $c_1 = \circ$, $c = c_2 \neq \circ$;
3. $c = c_1 \neq \circ$, $c_2 = \circ$.

The above notion of fusion generalizes to patterns as follows.

Definition 5. *Given three patterns x, y, z and an integer d, we say that z is the d-*fusion of x and y, *and write $z = x \bigtriangledown_d y$, if z can be obtained by removing the leading and trailing don't care characters from the pattern m defined as $m[i] = x[i+d] \bigtriangledown y[i]$, for all indices i.*

The breadth-first strategy adopted by our algorithm crucially relies on the following theorem, which highlights the structure of dense motifs:

Theorem 1. *Let x be a maximal dense motif with $dc(x) > 0$. Then:*

(a) there exists a maximal solid block b in x such that $\mathcal{M}(x) = \mathcal{M}(b)$, or
(b) there exist two maximal dense motifs y_1, y_2 such that:
- *$\mathcal{M}(x) = \mathcal{M}(y_1 \bigtriangledown_d y_2)$, for some d;*
- *there are two maximal solid blocks b_1, b_2 in x and an integer $\hat{d} > 0$ such that b_1 is a maximal solid block in y_1, b_2 is a maximal solid block in y_2, and $b_1 \circ^{\hat{d}} b_2$ is contained in $y_1 \bigtriangledown_d y_2$;*
- *$f(x) < \min\{f(y_1), f(y_2)\}$;*

For the proof of Theorem 1 we need to define another type of pattern combination, namely the operation of *merge* between two patterns, which is similar to the one introduced in [12]. Given two characters c_1, c_2, we define the operator $\oplus$ between them such that $c_1 \oplus c_2 = \circ$, if $c_1 \neq c_2$, and $c_1 \oplus c_2 = c_1 = c_2$, otherwise.

Definition 6. *Given two patterns x, y and an integer d, the d-merge of x and y is the pattern $z = x \oplus_d y$ which can be obtained by removing all leading and trailing don't cares from the pattern m defined as $m[i] = x[i + d] \oplus y[i]$ for all i.*

We want to stress the difference between the notions of merging and fusion: the merge of two patterns x, y is always well defined and more general than x, y, while the fusion of x, y may not exist and, if it does, is more specific than x, y.

For the proof of Theorem 1 we also need the property established by the following lemma.

Lemma 1. *Let x and y be maximal patterns, and d be an integer such that $z = x \oplus_d y \neq \epsilon$. Then z is a maximal pattern. Moreover, if $z \neq x$ (resp., $z \neq y$) then $f(z) > f(x)$ (resp., $f(z) > f(y)$).*

Proof. First we prove that z is maximal. By contradiction, suppose that this is not the case. Then, there exists a position i such that $z[i] = \circ$ and we can replace the $\circ$ with a solid character c without decreasing the frequency of the pattern. (Note that the position of the substitution can be to the left of the first character in z or to the right of the last character in z.) Since x and y are more specific than z, to every occurrence of x and y in the string corresponds an occurrence of z. Hence, every occurrence of x (resp., y) in the string, contains c in its $i + d$th (resp., ith) position. Therefore, by maximality of x and y, it must be $z[i] = x[i + d] = y[i] = c$, which is a contradiction. The relations between the frequencies of x, y and z follow trivially by their maximality. □

We are now ready to prove the theorem.

Proof (Theorem 1). Given a pattern x and two nonnegative integers $i \leq j$, we let $x^*[i \dots j]$ denote the pattern obtained by removing all the leading and trailing don't care characters from $x[i \dots j]$. Since x is a maximal dense pattern and $dc(x) > 0$, it is easy to see that there exist two dense patterns x_1, x_2 and an integer $d > 0$ such that $x = x_1 \circ^d x_2$, hence there exists an index $s_1 > 0$ such that $x^*[0 \dots s_1 - 1]$ and $x^*[s_1 + 1 \dots |x| - 1]$ are dense. We call these two patterns the *level-1 decomposition* of x (observe that many such decompositions may exist). Also, we let $\ell_1 = 0$ and $r_1 = |x| - 1$. Now, consider the following iterative process:

1. If in the level-i decomposition of x both $x^*[\ell_i \dots s_i - 1]$ and $x^*[s_i + 1 \dots r_i]$ have frequency strictly greater than $f(x)$, *or* at least one of $x^*[\ell_i \dots s_i - 1]$ and $x^*[s_i + 1 \dots r_i]$ is a solid block with frequency equal to $f(x)$, then terminate;
2. Otherwise, let $y = x^*[\ell_{i+1} \dots r_{i+1}]$ be (an arbitrary) one of $x^*[\ell_i \dots s_i - 1]$ or $x^*[s_i + 1 \dots r_i]$ which is not a solid block and has frequency equal to $f(x)$. Since y is dense, there exists an index s_{i+1}, $\ell_{i+1} < s_{i+1} < r_{i+1}$ such that $x^*[\ell_{i+1} \dots s_{i+1} - 1]$ and $x^*[s_{i+1} + 1 \dots r_{i+1}]$ are both dense. Call these two patterns the level-$(i + 1)$ decomposition of x. Set $i = i + 1$ and go to Step 1.

Assume that the decomposition process ends by finding a solid block b that is a solid block in x and has $f(b) = f(x)$. Then, $\mathcal{M}(b) = \mathcal{M}(x)$ and the theorem follows. Otherwise, at the last level j of the decomposition, we have that $f(x) < \min\{f(x^*[\ell_j \dots s_j - 1]), f(x^*[s_j + 1 \dots r_j])\}$. In this latter case, as explained in Section 2 (after Definition 2), we can determine two maximal dense patterns y_1, y_2 such that y_1 contains $x^*[\ell_j \dots s_j - 1]$, y_2 contains $x^*[s_j + 1 \dots r_j]$, and with $\mathcal{M}(y_1) = \mathcal{M}(x^*[\ell_j \dots s_j - 1])$ and $\mathcal{M}(y_2) = \mathcal{M}(x^*[s_j + 1 \dots r_j])$. Since $f(y_1) = f(x^*[\ell_j \dots s_j - 1])$ and $f(y_2) = f(x^*[s_j + 1 \dots r_j])$, we have that $f(x) < \min\{f(y_1), f(y_2)\}$. Observe that by construction there must exist two solid blocks b_1, b_2 in x and an integer $\hat{d}$ such that b_1 is a solid block in y_1, b_2 is a solid block in y_2, and $b_1 \circ^{\hat{d}} b_2$ is a sequence of two solid blocks in x. In fact, b_1 (resp., b_2) is the last (resp., the first) solid block of $x^*[\ell_j \dots s_j - 1]$ (resp., $x^*[s_j + 1 \dots r_j]$).

Next, we show that there exists a d such that the d-fusion $y_1 \bigtriangledown_d y_2$ is well defined, contains $b_1 \circ^{\hat{d}} b_2$, and $\mathcal{M}(y_1 \bigtriangledown_d y_2) = \mathcal{M}(x)$. We proceed as follows. Let us "align" $\mathcal{M}(x)$ and y_1 so to match the occurrences of b_1 in both patterns. Then, for a certain integer p, $\mathcal{M}(x)[i + p]$ corresponds to $y_1[i]$. Assume, for the sake of contradiction, that there exists an index j such that $\mathcal{M}(x)[j + p]$ is not more specific than $y_1[j]$. Then, Lemma 1 implies that $z = \mathcal{M}(x) \oplus_p \mathcal{M}(y_1) \neq \mathcal{M}(y_1)$, which contains $x^*[\ell_j \dots s_j - 1]$, is maximal and has frequency strictly greater than $f(y_1)$, which is impossible because we have chosen y_1 such that $\mathcal{M}(x^*[\ell_j \dots s_j - 1]) = \mathcal{M}(y_1)$ and therefore $f(x^*[\ell_j \dots s_j - 1]) = f(y_1)$. Therefore, $\mathcal{M}(x)$ contains y_1. A similar argument shows that $\mathcal{M}(x)$ contains y_2.

Since y_1 and y_2 are contained in $\mathcal{M}(x)$, there must exist a d such that $y_1 \bigtriangledown_d y_2$ is well defined and can be aligned with $\mathcal{M}(x)$ in such a way to match the blocks b_1 and b_2 of y_1 and y_2 with the corresponding blocks in $\mathcal{M}(x)$. Moreover, $\mathcal{M}(x)$ contains $y_1 \bigtriangledown_d y_2$, hence $f(y_1 \bigtriangledown_d y_2) \geq f(\mathcal{M}(x)) = f(x)$. However, since $y_1 \bigtriangledown_d y_2$ contains both $x^*[\ell_j \dots s_j - 1]$ and $x^*[s_j + 1 \dots r_j]$, it contains also $x^*[\ell_j \dots r_j]$, which, by the decomposition process, has frequency equal to $f(x)$. Therefore, $f(y_1 \bigtriangledown_d y_2) \leq f(x)$, and the theorem follows since $f(y_1 \bigtriangledown_d y_2) = f(x)$. □

In essence, Theorem 1 guarantees that we can find any maximal dense motif x either within $\mathcal{M}(b)$, for some maximal solid block b, or by d-fusing two higher-frequency maximal dense motifs y_1, y_2, for some d, finding $z = \mathcal{M}(y_1 \bigtriangledown_d y_2)$ and then possibly "trimming" z on both sides to obtain x.

Algorithm MADMX, whose pseudocode is reported in Figure 1, implements the strategy inspired by Theorem 1. It employs three (initially empty) sets *previous*, *current*, and *next*. In Line 2, the algorithm first stores the maximal solid blocks b in s for the given frequency in the set *blocks* (see Section 2). Then, it extracts all of the appropriate maximal dense motifs from $\mathcal{M}(b)$ in Lines 3–6, using the function extractMaximalDense, as implied by Theorem 1(a). Finally, Lines 7–16 implement the strategy as implied by Theorem 1(b). (In Line 10 a d-fusion $y_1 \bigtriangledown_d y_2$ is considered *valid* if it satifies the second property of Theorem 1(b).)

An important issue for the efficiency of MADMX is that it needs to compute the exact frequency of each generated pattern. For what concerns the fusion operation of two patterns x_1, x_2 in Line 10, observe that a simple computation

Algorithm MADMX()

Input: String s, frequency threshold σ, density threshold ρ
Output: Maximal dense motifs

```
1  previous ← ∅, current ← ∅, next ← ∅ ;
2  blocks ← maximal solid blocks of s with frequency ≥ σ;
3  for each b ∈ blocks do
4     find M(b) ;
5     DM ← extractMaximalDense(M(b));
6     for each x ∈ DM do current ← current ∪ {x};
7  while current ≠ ∅ do
8     for each x1 ∈ current do
9        for each x2 ∈ previous ∪ current do
10          for each d s.t. z = x1 ▽d x2 is a valid fusion do
11             find M(z);
12             DM ← extractMaximalDense(M(z));
13             for each x ∈ DM do
14                if f(x) ≥ σ and x ∉ previous ∪ current then next ← next ∪ {x};
15    previous ← previous ∪ current;
16    current ← next; next ← ∅;
17 return previous;
```

Figure 1. Pseudocode of algorithm MADMX.

on the pairs $(\ell_1, \ell_2) \in \mathcal{L}_{x_1} \times \mathcal{L}_{x_2}$ is sufficient to yield the frequencies of all the valid fusions of two patterns. However, given $z = x_1 \bigtriangledown_d x_2$, for a maximal dense pattern w which does not contain z in its entirety, we can only conclude that $f(w) \geq f(z)$. We then label the motifs for which the exact frequencies are known as *final*, and those for which only a lower bound to their frequencies is known as *tentative*, and update the lower bounds and the labels during the execution of the algorithm. Whenever the set *current* contains no final motifs, we can label as final the tentative motif in *current* with the highest lower bound to its frequency, and continue with the generation. The proof of the correctness of this assumption and further details on the implementation of the algorithm will be provided in the full version of this extended abstract. A crude upper bound on the running time of MADMX can be derived by observing that, for each pair of dense maximal motifs in output, the time spent during all the operations concerning that pair is (naively) $O\left(n^3\right)$, where n is the length of the input string. If P patterns are produced in output, the overall time complexity is $O\left(n^3P^2\right)$.

4 Experimental Validation of MADMX

We developed a first, non-optimized, implementation of MADMX in `C++` also including an additional feature which eliminates, from the set of initial maximal solid blocks, those shorter than a given threshold min_ℓ. The purpose of this latter heuristics is to speed up motif generation driving it towards the discovery of (possibly) more significant motifs, with the exclu-

sion of spurious, low-complexity ones. (The code is available for download at `http://www.dei.unipd.it/wdyn/?IDsezione=4534`.)

We performed two classes of experiments to evaluate how significant is the set of motifs found using our approach. The first class of experiments, described in Section 4.1, compares our motifs with the known biological repetitions available in `RepBase` [7], a very popular genomic database. The second class of experiments, described in Section 4.2, aims at comparing the motifs extracted by MADMX with those extracted by VARUN using the same z-score metric employed in [2] for assessing their relative statistical significance.

4.1 Evaluating significance by known biological repetitions

`RepBase` [7] is one of the largest repositories of prototypic sequences representing repetitive DNA from different eukaryotic species, collected in several different ways. `RepBase` is used as a reference collection for masking and annotation of repetitive DNA through popular tools such as REPEATMASKER [15]. REPEATMASKER screens an input DNA sequence s for simple repeats and low complexity portions, and interspersed repeats using `RepBase`. Sequence comparisons are performed through Smith-Waterman scoring. REPEATMASKER returns a detailed annotation of the repeats occurring in s, and a modified version of s in which all of the annotated repeats are masked by a special symbol (`N` or `X`). With the current version of `RepBase`, on average, almost 50% of a human genomic DNA sequence will be masked by the program [15].

Most of the interspersed repeats found by REPEATMASKER belong to the families called SINE/ALU and LINE/L1: the former are *Short INterspersed Elements* that are repetitive in the DNA of eukaryotic genomes (the Alu family in the human genome); the latter are *Long Interspersed Nucleotide Elements*, which are typically highly repeated sequences of 6K–8K bps, containing RNA polymerase II promoters. The LINE/L1 family forms about 15% of the human genome.

We have conducted an experimental study using MADMX and REPEATMASKER on *Human Glutamate Metabotropic Receptors* HGMR 1 (410277 bps) and HGMR 5 (91243 bps) as input sequences. We have downloaded the sequences from the March 2006 release of the UCSC Genome database (`http://genome.ucsc.edu`). REPEATMASKER version was open-3.2.7, sensitive mode, with the query species assumed to be homologous; it ran using `blastp` version 2.0a19MP-WashU, and `RepBase` update 20090120.

The experiments to assess the biological significance of the maximal dense motifs extracted by MADMX involved three separate stages. In the first stage, we ran REPEATMASKER on the input sequences HGMR 1 and HGMR 5, searching for interspersed repeats using `RepBase`. One of the output files (`.out`) of REPEATMASKER contains the list of found repeats, and provides, for each occurrence, the substring $s[i \dots j]$ of the input sequence s which is locally aligned with (a substring of) the repeat.

In the second stage, we ran MADMX on the same DNA sequences, with density threshold $\rho = 0.8$, frequency threshold $\sigma = 4$, and $\min_\ell = 15$. In order to filter out simple repeats and low complexity portions, which are dealt with by

REPEATMASKER without resorting to `RepBase`, we modified MADMX eliminating periodic maximal solid blocks (with short periods), which are the seeds of simple repeats. Then, we identified the occurrences of the motifs returned by MADMX in the input sequences, using REPEATMASKER as a pattern matching tool (i.e., replacing `RepBase` with the set of motifs returned by MADMX as the database of known repeats). The underlying idea behind this use of REPEATMASKER was to employ the same local alignment algorithms, so to make the comparison fairer.

In the third stage, we cross-checked the intervals associated with the occurrences of the `RepBase` repeats against those associated with the occurrences of our motifs. Surprisingly, MADMX was able to identify and characterize *all* of the intervals of the known SINE/ALU repeats in HGMR 1 and HGMR 5 (respectively, 56 repeats plus an extra unclassified for HGMR 1, and 20 plus an extra unclassified for HGMR 5). The remaining occurrences of the motifs permitted to identify 29 repeats out of 78 of the LINE/L1 family in HGMR 1. (A more detailed account of the whole range of experiments conducted using REPEATMASKER and the data sets by Tompa et el. and Sandve et al. will be provided in the full version.)

4.2 Evaluating significance by statistical z-score ranking

The z-score is the measure of the distance in standard deviations of the outcome of a random variable from its expectation. Consider a DNA sequence s of length n as if it was generated by a stationary, i.i.d. source with equiprobable symbols; an approximation to the z-score for a motif of length m that contains c solid characters and appears f times in s is given by $Z = \frac{f-(n-m+1)\times p}{\sqrt{(n-m+1)\times p\times(1-p)}}$, where $p = (1/4)^c$. This metric was used in [2] to assess the significance of the motifs extracted by VARUN and to rank them in the output.

We employed the code for VARUN provided by the authors to extract the rigid motifs from the DNA sequences analyzed in [2]. We then ran MADMX on the same sequences using the same frequency parameters, and setting the minimum density threshold ρ in such a way to obtain a comparable yet smaller output size. In this fashion, we tested the ability of MADMX to produce a succinct yet significant set of motifs, by virtue of its more flexible notion of density.

The results are shown in Table 1. For VARUN we used $D = 1$, thus allowing at most one don't care between two solid characters, and ran MADMX with $min_\ell = 1$, so to obtain the *complete* family of maximal dense motifs. In the table, there is a row of the table for each sequence (identified in the first column). Each sequence, whose total length is reported in the second column, is obtained as the concatenation of a number of smaller subsequences, reported in the third column. On each sequence, both tools were run with the same frequency threshold σ, and the table reports for both the output size in terms of the number of motifs returned and the execution time in seconds. Also, for MADMX, the table reports the density threshold ρ used in each experiment.

For each experiment, we compared the best top-m z-scores, with $m = 10, 50$, and 100, as follows. Note that, in general, the top-m motifs found by MADMX and VARUN differ. Thus, we let z^i_M (resp., z^i_V) be the z-score of the ith motif

name	length	#	σ	VARUN		MADMX			best top-m z-scores				
				|output|	time	ρ	|output|	time	m=10	m=50	m=100	m^*	$\hat{m}$
ace2	500	1	2	1866	3s	0.7	1762	18s	10	50	100	1571	1067
ap1	500	1	2	1555	1s	0.7	1304	5s	10	50	100	392	13
gal4	3000	6	4	9764	12s	0.67	7606	67s	10	49	99	16	16
gal4$^{(*)}$	3000	6	4	9764	12s	0.65	11733	191s	10	50	100	9764	301
uasgaba	1000	2	2	4586	30s	0.70	4194	90s	10	50	100	175	175

Table 1. Results of the comparison with VARUN.

in decreasing z-score order obtained by MADMX (resp., VARUN). For each m, the table reports how many times it was $z^i_M \geq z^i_V$, for $1 \leq i \leq m$. Also, column m^* (resp., column $\hat{m}$) gives the maximum m such that $z^i_M \geq z^i_V$ (resp., $z^i_M > z^i_V$) for every $1 \leq i \leq m$.

Even when MADMX is calibrated to yield a slightly smaller output, the quality of the motifs extracted, as measured by the z-score, is higher than those output by VARUN. Indeed, for sequences ace2 and uasgaba a very large prefix of the top-ranked motifs extracted by MADMX features strictly greater z-scores of the corresponding top-ranked ones extracted by VARUN. In fact, for all of the four sequences, at least the thirteen top-ranked motifs enjoy this property. To shed light on the slightly worse performance of MADMX on gal4, we re-ran MADMX with a different density threshold, so to obtain a slightly larger output (see row gal4$^{(*)}$). In this case, the top-301 motifs extracted by MADMX have z-score strictly greater than the corresponding motifs extracted by VARUN, while the execution time remains still acceptable.

For all runs, the top z-score of a motif discovered by MADMX is considerably higher than the one returned by VARUN. Specifically, on ace2 our best z-score is 387 763 vs. 12 027 of VARUN; on ap1, we have 12 027 vs. 1 490; on gal4 it is 75 vs. 28; on gal4$^{(*)}$ it is 150 vs. 28; on uasgaba we have 134 532 vs. 67 059. This reflects the high selectivity of MADMX, which is to be attributed mostly to adoption of a more flexible density constraint.

We must remark that MADMX (in its current nonoptimized version) is slower than VARUN, but it still runs in time acceptable from the point of view of a user. To further investigate the tradeoff between execution time and significance of the discovered motifs, we repeated the experiments running MADMX with $\min_\ell = 2$ and $\rho = 0.65$, for all sequences. The running time of MADMX was almost halved, while the small output produced still featured high quality. In fact, for sequences ace2, ap1, and uasgaba the top-100 motifs extracted by MADMX have z-score greater or equal than the corresponding ones returned by VARUN.

We also have attempted a comparison between VARUN and MADMX on longer sequences (such as HGMR 1) at higher frequencies (since, unfortunately, VARUN does not seem to be able to handle low frequencies on very long sequences). Even allowing a higher number of don't cares between solid characters ($D = 2$) for the motifs of VARUN, all of the top-m z-scores featured by the motifs extracted by

MADMX are greater than or equal to the corresponding scores in the ranking of VARUN, with m reaching the size of VARUN's output. In fairness, we remark that VARUN was designed to work at its best on protein sequences, while MADMX's main target are DNA sequences. Hence, these two tools should be regarded as complementary. Moreover, VARUN has the advantage of retrieving flexible motifs, while MADMX focuses only on rigid ones.

Acknowledgments The authors wish to thank Alberto Apostolico and Matteo Comin for providing the code and giving valuable insights on VARUN, Ben Raphael for suggesting the use of REPEATMASKER, and Roberta Mazzucco and Francesco Peruch for coding MADMX.

References

1. R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In *Proc. of 20th VLDB*, pages 487–499, 1994.
2. A. Apostolico, M. Comin, and L. Parida. VARUN: discovering extensible motifs under saturation constraints. *IEEE Trans. on Computational Biology and Bioinformatics*, 2009. To appear.
3. A. Apostolico and L. Parida. Incremental paradigms of motif discovery. *Journal of Computational Biology*, 11(1):15–25, 2004.
4. A. Apostolico and C. Tagliacollo. Optimal offline extraction of irredundant motif bases. In *Proc. of 13th COCOON*, LNCS 4598, pages 360–371, 2007.
5. A. Apostolico and C. Tagliacollo. Incremental discovery of the irredundant motif bases for all suffixes of a string in $O\left(n^2 \log n\right)$ time. *Theoretical Computer Science*, 408(2-3):106–115, 2008.
6. H. Arimura and T. Uno. Mining maximal flexible patterns in a sequence. In *Proc. of 21st JSAI*, LNCS 4914, pages 307–317, 2007.
7. J. Jurka, V.V. Kapitonov, A. Pavlicek, P. Klonowski, O. Kohani, and J. Walichiewicz. Repbase Update, a database of eukaryotic repetitive elements. *Cytogenet. Genome Res.*, 110:462–467, 2005.
8. M. Morris, F. Nicolas, and E. Ukkonen. On the complexity of finding gapped motifs. *CoRR*, abs/0802.0314, 2008.
9. L. Parida. Some results on flexible-pattern discovery. In *Proc. of 11th CPM*, LNCS 1848, pages 33–45, 2000.
10. L. Parida. *Pattern discovery in bioinformatics*. Mathematical and Computational Biology Series. Chapman & Hall / CRC, Boca Raton, FL, 2008.
11. N. Pisanti. *Segment-based distances and similarities in genomic sequences*. PhD thesis, University of Pisa, Italy, 2002.
12. N. Pisanti, M. Crochemore, R. Grossi, and M.F. Sagot. Bases of motifs for generating repeated patterns with wild cards. *IEEE Trans. on Computational Biology and Bioinformatics*, 2(1):40–50, 2005.
13. I. Rigoutsos and A. Floratos. Combinatorial pattern discovery in biological sequences: the TEIRESIAS algorithm. *Bioinformatics*, 14(1):55–67, 1998.
14. S. Saha, S. Bridges, Z.V. Magbanua, and D.G. Peterson. Empirical comparison of *ab initio* repeat finding programs. *Nucleic Acids Res.*, 36(7):2284–2294, 2008.
15. A.F.A. Smit, R. Hubley, and P. Green. *RepeatMasker Open-3.0*. `http://www.repeatmasker.org`, 1996–2004.
16. E. Ukkonen. Structural analysis of gapped motifs of a string. In *Proc. of 32nd MFCS*, LNCS 4708, pages 681–690, 2007.