AliBiMotif: Integrating alignment and biclustering to unravel Transcription Factor Binding Sites in DNA sequences

Joana P. Gonçalves*

Knowledge Discovery and Bioinformatics (KDBIO),
INESC-ID Computer Science Department, IST,
Technical University of Lisbon,
Rua Alves Redol 9, 1000-029 Lisboa, Portugal
E-mail: jpg@kdbio.inesc-id.pt
*Corresponding author

Yves Moreau

BIOI, Electrical Engineering Dept. (ESAT-SCD), KULeuven,
Kasteelpark Arenberg 10, 3001 Heverlee, Belgium
E-mail: yves.moreau@esat.kuleuven.be

Sara C. Madeira

Knowledge Discovery and Bioinformatics (KDBIO),
INESC-ID Computer Science Department, IST,
Technical University of Lisbon,
Rua Alves Redol 9, 1000-029 Lisboa, Portugal
E-mail: smadeira@kdbio.inesc-id.pt

Abstract: Transcription Factors (TFs) control transcription by binding to specific sites in the promoter regions of the target genes, which can be modelled by structured motifs. In this paper we propose AliBiMotif, a method combining sequence alignment and a biclustering approach based on efficient string matching techniques using suffix trees to unravel approximately conserved sets of blocks (structured motifs) while straightforwardly disregarding non-conserved stretches in-between. The ability to ignore the width of non-conserved regions is a major advantage of the proposed method over other motif finders, as the lengths of the binding sites are usually easier to estimate than the separating distances.

Keywords: biclustering; sequence alignment; motif finding algorithm; structured motif identification; binding site; TF; transcription factor; TFBS; cis-regulatory module discovery; promoter region; motif finder; integrative mining.

Reference to this paper should be made as follows: Gonçalves, J.P., Moreau, Y. and Madeira, S.C. (xxxx) 'AliBiMotif: Integrating alignment and biclustering to unravel Transcription Factor Binding Sites in DNA sequences', *Int. J. Data Mining and Bioinformatics*, Vol. x, No. x, pp.xxx–xxx.

Copyright © 2012 Inderscience Enterprises Ltd.

Biographical notes: Joana P. Gonçalves received the Diploma in Computer Science and Engineering from the University of Beira Interior (UBI), Portugal, in 2007. She is currently a PhD student at IST, Technical University of Lisbon, Portugal, and a Doctoral Researcher at the Knowledge Discovery and Bioinformatics group (KDBIO) of INESC-ID in Lisbon, Portugal, visiting the Bioinformatics Research group (BIOI) at the Katholieke Universiteit Leuven, Belgium. Her PhD work focuses on the development of integrative mining strategies to unravel functional or regulatory gene modules. She is also the developer of BiGGEsTS, a software for biclustering analysis of time series expression data, and Polar Mapper, an application for integrated analysis of protein-protein interaction networks and mRNA data. Her research interests include discrete algorithms, integrative data mining and applications to biological and medical problems.

Yves Moreau received the BSc Degree in Electrical Engineering from the Faculté Polytechnique de Mons, Belgium, in 1992, the MSc Degree in Applied Mathematics from the Brown University, Providence, RI, USA in 1994 and the PhD Degree in Electrical Engineering from the Katholieke Universiteit Leuven, Belgium, in 1998. He is a Full Professor at the Katholieke Universiteit Leuven and head of the Bioinformatics Research group at the Department of Electrical Engineering of the same university. He is also a Principal Investigator of the SymBioSys Center for Computational Systems Biology and of the BioMaGNet Interuniversity Attraction Pole, a collaborative network of belgian universities on Computational Systems Biology. His research interests fall in the fields of bioinformatics and computational biology, including the development and application of computational methods in systems biomedicine and top down systems biology.

Sara C. Madeira received the Diploma in Computer Science from the University of Beira Interior, Portugal, in 2000, and the MSc Degree in Computer Science and Engineening from IST, Technical University of Lisbon, Portugal, in 2002. In December 2008, she received the PhD Degree in Computer Science and Engineering also from IST. Her PhD Thesis in the area of computational biology is entitled "Efficient Biclustering Algorithms for Time Series Gene Expression Data Analysis". She is currently a Senior Researcher at the Knowledge Discovery and Bioinformatics group of INESC-ID in Lisbon, Portugal, and an Auxiliar Professor at the Department of Computer Science and Engineering of IST. Her research interests include algorithm design, machine learning, data mining and their application in integrative bioinformatics and biomedical problems.

1 Introduction

Transcription Factors (TFs) are key elements of regulatory mechanisms, binding to specific sites in the DNA of protein coding genes to enhance or inhibit their transcription into mRNA. Transcription Factor Binding Sites (TFBSs) are short stretches with 5–25 nucleotides long usually located within non-coding parts of

DNA in the so-called promoter regions. These can be modelled as structured motifs, that is, sets of conserved sequences (motifs) separated by non-conserved regions of unspecified length. It is known that genes regulated by common TFs share identical binding sites. In this context, similar promoter regions can provide important insights into functional relatedness of genes and their corresponding regulatory mechanisms. Identifying structured motifs is a challenging problem, since one does not know whether a given set of sequences is regulated by the same TFs or not, and prior information on the composition of the motif is usually unavailable.

Methods for identifying structured motifs in DNA sequences follow either a probabilistic or combinatorial approach. Most probabilistic strategies rely on iterative Expectation-Maximisation (EM) or Gibbs sampling (Bailey and Elkan, 1994; Buhler and Tompa, 2002; Lawrence et al., 1993; Schug and Overton, 1997; Thakurta and Stormo, 2001), where convergence to a global maximum is not guaranteed. Moreover, they are known to be sensitive to noise in data. The majority of probabilistic methods has been restricted to the identification of simple motifs, although a few approaches support the search for structured motifs (Bailey and Elkan, 1994; Thakurta and Stormo, 2001).

Combinatorial methods typically enumerate all possible motifs and search for their occurrences in DNA sequences using heuristics to reduce the exponential search space (Carvalho et al., 2006; Eskin and Pevzner, 2002; Marsan and Sagot, 2000; Mendes et al., 2006; Zhang and Zaki, 2006). However, they are not able to discriminate relevant motifs from potentially numerous random motifs matching the generated models. Furthermore, they require a large number of parameters. This is a major limitation when little is known about the configuration of the underlying motifs and imposes severe restrictions on the motifs that can be found.

In this context, we propose AliBiMotif, an algorithm to identify structured motifs composed of conserved regions satisfying a minimum length. AliBiMotif further disregards the width of non-conserved regions separating the binding sites. This feature presents an advantage over other motif finders, as the length of conserved regions is usually easier to estimate. Following the approach of Wang et al. (2007), coupling sequence alignment and biclustering to obtain local multiple sequence alignment of RNA sequences in a divide-and-conquer approach, AliBiMotif identifies TFBSs in two steps: identification of conserved regions spanning subsets of sequences, and grouping of sequences with similar motif structures. First, pairwise sequence alignment is used to reveal locally conserved regions in the sequences. A greedy clustering approach is then applied to agglomerate regions exhibiting a large degree of similarity into larger sets. Each cluster is called a candidate block and defines a conserved region spanning multiple sequences. Second, structured motifs are identified using a biclustering algorithm to group sets of sequences with similar ordered sets of candidate blocks.

This paper is outlined as follows. Section 2 describes related work on methods for structured motif identification. Section 3 presents AliBiMotif, motivated with examples. It further includes algorithmic details and devises its complexity analysis. Finally, Section 4 outlines conclusions and future work.

2 Related work

MEME (Bailey and Elkan, 1994) and Co-Bind (Thakurta and Stormo, 2001) are two examples of probabilistic motif finders. MEME (Bailey and Elkan, 1994) uses

EM to fit successive mixture models to the set of sequences in order to identify the several components of the structured motifs. It requires the width of the motifs as input. Co-Bind (Thakurta and Stormo, 2001) models structures of two motifs using Position Weight Matrices (PWMs) and finds PWMs maximising the joint likelihood of pairwise occurrences. Since a limited number of Gibbs sampling steps is performed to select motifs, many patterns are disregarded. Convergence to an optimal PWM is also not guaranteed.

Regarding the combinatorial approaches for structured motif identification, we highlight SMILE (Marsan and Sagot, 2000), MITRA-dyad (Eskin and Pevzner, 2002), RISO (Carvalho et al., 2006), EXMOTIF (Zhang and Zaki, 2006) and MUSA (Mendes et al., 2006). SMILE (Marsan and Sagot, 2000) relies on a traversal with backtracking of a generalised suffix tree to spell all occurrences of simple motifs for a structured motif with gaps. The search for each motif is guided by a virtual lexicographic trie containing all possible motifs in a range of lengths. Its time complexity is exponential in the number of gaps (Carvalho et al., 2006). MITRA-dyad (Eskin and Pevzner, 2002) reduces the problem of searching for pairs of motifs to the one of finding a simple motif. It combines SMILE's tree-like structure and search, and the Winnower's graph construction for pruning the search space. MITRA relies on a mismatch tree. Although it will typically examine a smaller search space, analysing a graph per node increases traversal time. RISO (Carvalho et al., 2006) is also based on SMILE and introduces box-links to store the information needed to jump from each motif to another and skip the corresponding nodes in the tree. A factor tree is used. Combined, these enhancements produce a significant reduction in both time and space. EXMOTIF (Zhang and Zaki, 2006) extracts all frequent occurrences of a structured motif model with a given quorum. Sequences are represented by sorted lists of symbol occurrence positions. The positional join operation efficiently builds a list of occurrences when concatenating two sequences. Occurrences of simple/structured motifs are obtained by positional joins on the list of symbols/list of occurrences of simple motifs. MUSA (Mendes et al., 2006) generates all motif models of a given length. A matrix of co-occurrences is then constructed to depict a tolerant score of the most common configuration in the sequences. Biclustering is then applied to group models and identify structured motifs. Identification of all interesting correlations is not guaranteed but weak motifs, only revealed by exhaustive enumeration, can be found.

3 AliBiMotif method

AliBiMotif is an algorithm to identify and report structured motifs contained in a set of DNA sequences. It is inspired on BlockMSA (Wang et al., 2007), an algorithm combining sequence alignment and biclustering in a divide-and-conquer approach to produce local Multiple Sequence Alignment (MSA). We note that both local MSA and structured motif finding aim at identifying locally conserved regions in sequences. Nevertheless, the problems are distinct as the expected outcome of a local MSA is an alignment of the sequences while the identification of structured motifs should report only the conserved regions of the structured motifs and the subsets of sequences in which they appear. This observation is relevant in the sense

that it allows one to disregard the non-conserved regions in-between when searching for the components of a structured motif.

In what follows, we motivate AliBiMotif and describe its algorithmic details. We first introduce basic definitions to support the explanation of our approach.

Fragment: For a sequence S_k of length $|S_k|$, a fragment is a contiguous subsequence, $S_k[i..j]$ $(1 \leq i < j \leq |S_k|)$, starting at position i and ending at j.

n-block: An n-block $b = \{f_1, f_2, \ldots, f_n\}$ is an alignment region which is highly conserved in a set of n sequences, S_{sub}. It thus contains n equal-length fragments, one from each of the n sequences in S_{sub}. In this case, $|S_{sub}| = n$.

n-block similarity score: Given an n-block b, its similarity score is the sum of the alignment scores of all $\binom{n}{2}$ pairwise combinations of fragments f_i, f_j $(1 \leq i < j \leq n)$ from b: $Score_{block}(b) = \sum_{1 \leq i < j \leq n} Score(f_i, f_j)$, where $Score(f_i, f_j)$ is the alignment score between f_i and f_j.

Structured motif: A structured motif is an ordered set of non-overlapping n-blocks occurring exactly or approximately in a subset of n sequences.

Given the aforementioned definitions, the problem of identifying structured motifs translates into that of finding maximal ordered sets of blocks composed of fragments occurring approximately in subsets of sequences. AliBiMotif addresses this problem in two steps, described in detail in Sections 3.1 and 3.2. First, locally conserved regions within subsets of sequences (candidate blocks) are identified using an approach adapted from BlockMSA (Wang et al., 2007). Then, subsets of sequences exhibiting a similar structure of candidate blocks are grouped and reported together with the corresponding structured motifs using AliBiMotif biclustering proposed in this work. Throughout this section, we appropriately refine the concept of structured motif in the context of the biclustering approach.

3.1 Construction of candidate blocks

AliBiMotif starts by identifying regions exhibiting a high degree of conservation across subsets of the input sequences. Formally, the problem is defined as follows. Consider a set of sequences, $S = \{S_1, S_2, \ldots, S_N\}$ a threshold k corresponding to the minimum fragment length and a 2-block similarity score threshold t. As in BlockMSA, our approach uses an external sequence alignment program to generate a library of pairwise alignments for all combinations of sequence pairs. A window of length k is slid through each pairwise alignment, delimiting consecutive overlapping k-length sub-regions composed of two k-length fragments from the pair of sequences. BlockMSA keeps only the 2-blocks which hold two properties: being ungapped sub-regions and having a pairwise similarity score greater than threshold t. AliBiMotif allows gaps (insertions or deletions) to be included in the conserved region by imposing only threshold t on the alignment score, which consequently controls the number of gaps. Pairwise blocks are then used to construct candidate blocks, potentially larger and containing fragments from more

than two sequences. Seed fragments are chosen from the closest sequences. Each seed fragment constitutes an initial cluster. A greedy clustering-based approach is then applied to iteratively add more fragments from distinct sequences to each cluster. A fragment is added to a cluster if it shows the highest local conservation consistency with the fragments already in the cluster (maximum sum of pairwise alignment scores) (Wang et al., 2007).

3.2 Mining of structured motifs using AliBiMotif biclustering

A biclustering method is used to mine the structured motifs from the sequences. The differences between AliBiMotif biclustering and the biclustering approach in BlockMSA (Wang et al., 2007) lie in two key steps: the mapping of sequences and candidate blocks to the biclustering formulation, and the biclustering algorithm (see Figure 1).

BlockMSA defines a binary matrix, where rows and columns represent candidate blocks and sequences, respectively. Each element is either 1 or 0 depending on whether the block is in the sequence or not (Figure 1(b) and (e)). BiMax (Prelič et al., 2006) is then applied to identify biclusters in the matrix. It uses a divide-and-conquer approach to extract biclusters composed of 1s not entirely contained in any other bicluster, that is, inclusion-maximal biclusters (Figure 1(b) and (e)).

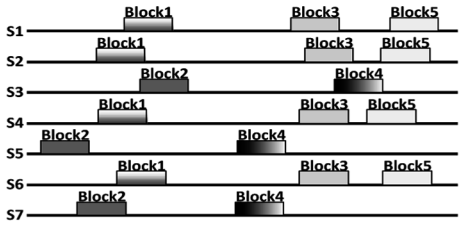
AliBiMotif rewrites each sequence as the ordered set of candidate blocks it contains (Figure 1(c) and (f)). A biclustering algorithm based on e-CCC-Biclustering (Madeira and Oliveira, 2009) is then used to identify ordered structures of blocks occurring exactly or approximately in the sequences (Figure 1(c) and (f)).

Figure 1 aims to present the discrepancies between the results generated by BlockMSA and AliBiMotif mapping and biclustering approaches when both perfect and approximately conserved structured motifs are considered. Figure 1(a) and (d) show the candidate blocks identified in two different sets of sequences, where Figure 1(a) contains the original example used to motivate BlockMSA (Wang et al., 2007) and Figure 1(d) is an adaptation of the former to include inconsistencies in the block occurrences, giving rise to imperfect structured motifs. In the case of BlockMSA, this results in biclusters not entirely composed of 1s, while for AliBiMotif the consequence is that the rewritten sequences exhibit insertions, deletions and substitutions in their ordered lists of blocks.

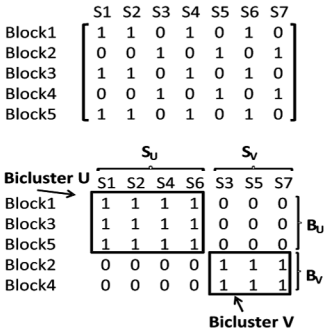
Changes to the original example are highlighted in Figure 1(d), where Block3 was inserted in S5 and deleted from S6 and Block2 was replaced by Block1 in S7. Figure 1(b) and (c) show that both BlockMSA and AliBiMotif biclustering approaches are able to retrieve the same biclusters when no inconsistencies are introduced. Figure 1(e) and (f) highlight the 'imperfect' biclusters corresponding to the previous ones found by BlockMSA and AliBiMotif biclustering, when inconsistencies are considered.

In Figure 1(e), it is clear that BlockMSA is no longer able to recover the same biclusters, excluding both sequences S6 and S7, while AliBiMotif still is (see Figure 1(f)). Repercutions of the inconsistencies introduced in the adapted example (Figure 1(d)) are highlighted in both the mappings (top figures in Figure 1(e) and (f)) and the results of the biclustering approaches (bottom figures in Figure 1(e) and (f)) of BlockMSA and AliBiMotif. Both algorithms identify additional

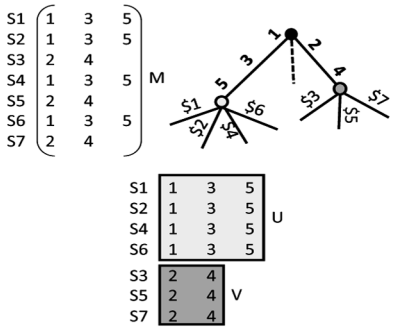
Figure 1 Candidate blocks identified in (a) the example in (Wang et al., 2007), and (d) an adaptation to include inconsistencies in the block occurrences. (b), (e): BlockMSA mapping of blocks to matrix (top) and biclusters found by BiMax (bottom) for (a) and (d). (c), (f): AliBiMotif sequence rewriting (top) and biclusters found by AliBiMotif biclustering (bottom) for (a) and (d).



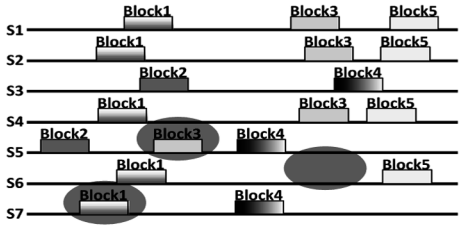
(a) Perfect biclusters: sequences and blocks.



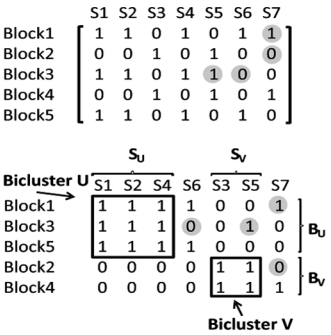
(b) Perfect biclusters: BlockMSA.



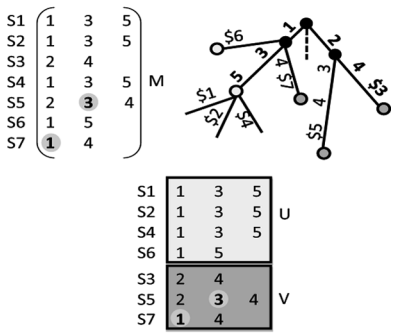
(c) Perfect biclusters: AliBiMotif.



(d) Imperfect biclusters: sequences and blocks.



(e) Imperfect biclusters: BlockMSA.

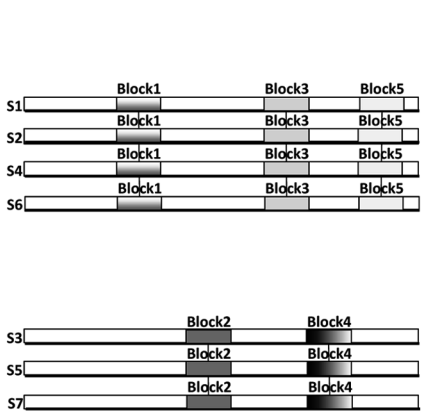


(f) Imperfect biclusters: AliBiMotif.

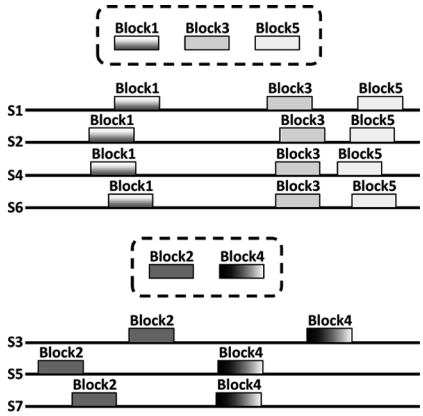
biclusters, including {{S6,S7}, {Block1}} in BlockMSA and {{S1,S2,S4,S6,S7}, {Block1,Block3}} in AliBiMotif, which are not represented.

As BlockMSA and AliBiMotif address different problems, they also produce distinct outputs. Figure 2 presents the output of both methods for the examples in Figure 1, where: Figure 2(a) and (b) show the sequence alignment obtained by BlockMSA and the structured motifs unravelled by AliBiMotif for the original example in Figure 1(a), respectively; and Figure 2(c) and (d) show the sequence alignment obtained by BlockMSA and the structured motifs identified by AliBiMotif for the adapted example in Figure 1(d), considering only the biclusters represented in Figure 1(e) and (f), respectively.

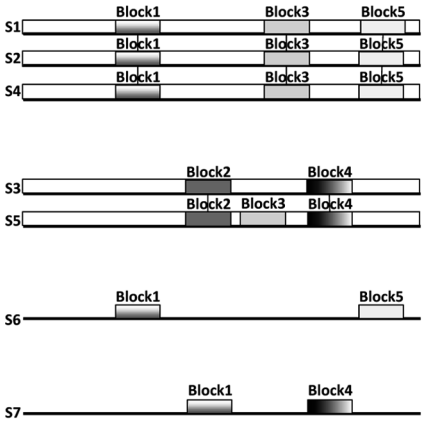
Figure 2 Output of (a) BlockMSA and (b) AliBiMotif for the example in (Wang et al., 2007) depicted in Figure 1(a). Output for an adaptation to consider inconsistencies in the block structure (Figure 1(d)), leading to biclusters not entirely composed of 1s and imperfect alignments in the case of (c) BlockMSA, and approximately conserved structured motifs in the case of (d) AliBiMotif. Changes to the example (Figures 1(a) vs 1(d)): Block3 inserted in S5, deleted from S6; Block2 replaced by Block1 in S7.



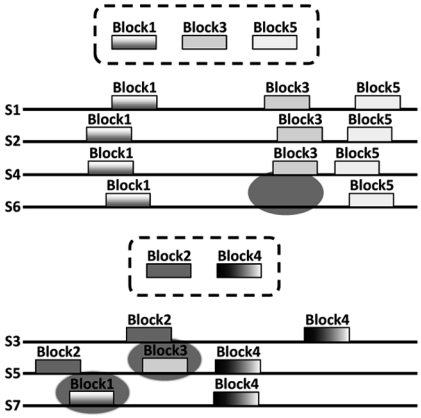
(a) Perfect biclusters: BlockMSA.



(b) Perfect biclusters: AliBiMotif.



(c) Imperfect biclusters: BlockMSA.



(d) Imperfect biclusters: AliBiMotif.

In the case of perfectly conserved structured motifs, it is clear from Figure 2(c) that BlockMSA devises perfect alignments of the sequences in two disjoint subsets of sequences, namely {S1,S2,S4,S6} and {S3,S5,S7}. In Figure 2(d) one can see that the output of AliBiMotif is also a pair of perfectly conserved structured motifs composed by the ordered set of blocks {Block1,Block3,Block5} spanning sequences {S1,S2,S4,S6} and another ordered set of blocks {Block2,Block4} occurring in sequences {S3,S5,S7}.

Regarding the example with inconsistent or approximately conserved block structures, Figure 2(c) shows that, for the biclusters considered in Figure 1(e), BlockMSA generates two alignments comprising the disjoint sets of sequences {S1,S2,S4} and {S3,S5}, from which sequences S6 and S7 were left out, respectively. The output of AliBiMotif for the same example, considering the retrieved biclusters presented in Figure 1(f) and pictured in Figure 2(d), shows that both structured motifs unravelled in the case of perfectly conserved block structures and described previously in Figure 2(b) could be recovered, besides the introduced inconsistencies. Note, however, that the occurrences of these structured motifs in the sequences are no longer exact, as it is also highlighted in Figure 2(d).

Outlined the goals and strategies of both approaches, we now proceed with a detailed description of AliBiMotif, our proposed method for the identification of approximately conserved structured motifs in DNA sequences.

3.2.1 Mapping of sequences and blocks to the biclustering formulation

AliBiMotif defines the following mapping from the results of candidate block construction to the biclustering formulation: each sequence is rewritten as the ordered set of the candidate blocks it contains, disregarding the non-conserved regions in-between. In this case, the alphabet of the new sequences holds the identifiers of all candidate blocks unravelled in the previous step (see Figure 1(c)). Given the rewritten sequences, the problem of searching for structured motifs translates into that of grouping sequences with approximate block structures, that is, identifying e-motifs (see definition below) and their corresponding occurrences.

Our mapping leads to three important properties. First, the non-conserved regions are excluded from further analysis prior to the application of the biclustering method, which is more efficient. Moreover, it enables to completely disregard the distance between the components of the structured motif, often hard to estimate. Second, we take advantage of both the block ordering granted by the sequence alignment and the fact that a block appears in each sequence at most once to formulate the structured motif finding problem as an efficient biclustering approach restricted to the search of contiguous subsequences pertaining a maximum number of e errors, e-motifs, together with their occurrences in the rewritten sequences. Third, the introduction of an error threshold on the block structure enables to relax the contiguity constraints and further provide a reasonable degree of control on the total length of the structured motif.

e-neighbourhood: The e-neighbourhood of a sequence S_k of length $|S_k|$, is defined as the set of sequences $N_e(S_k)$ needing at most e transforming operations (substitution, insertion, deletion) to obtain S_k. In other words, for each sequence S_i $(1 \leq i \leq |N_e(S_k)|)$ in $N_e(S_k)$, the maximum Hamming distance allowed between S_i and S_k is e, $Hamming(S_k, S_i) \leq e$, where e is an integer such that $e \geq 0$.

e-motif: An e-motif is a sequence m of length $|m|$ occurring within an e-neighbourhood, $N_e(m)$, in a set of sequences at potentially distinct starting positions.

3.2.2 Biclustering algorithm

AliBiMotif biclustering is adapted from e-CCC-Biclustering (Madeira and Oliveira, 2009). The major differences focus on:

- the use of sequences of variable lengths, opposed to a matrix-like structure
- the elimination of a matrix transformation, which was used to restrict the occurrences of a model to the same positions in the sequences.

This biclustering approach identifies and reports all maximal e-motif biclusters (see definitions below) in a set of N sequences rewritten for a set of B candidate blocks. In order to be valid, a given e-motif model m of length $|m|$ must have length of at least q_m symbols and occur exactly or approximately with a maximum of e errors in at least q_s distinct sequences. Formally: given a set of N sequences, S, and three integers $e \geq 0$, $2 \leq q_s \leq N$ and $2 \leq q_m \leq B$, where e is the maximum number of errors allowed per e-motif occurrence, q_s the quorum on the number of sequences and q_m the mininum length for a valid model, AliBiMotif biclustering finds all e-motif models m with length of at least q_m appearing in at least q_s distinct sequences in S.

e-motif bicluster: An e-motif bicluster is an e-motif m, a subset of sequences $s = \{s_1, s_2, \ldots, s_k\}$ from a set S ($|s| \leq |S|$) and a set of initial and final positions $p = \{(i_1, f_1), (i_2, f_2), \ldots, (i_k, f_k)\}$, such that each pair of positions, (i_x, f_x) $(1 \leq x \leq k)$, identifies an e-neighbourhood occurrence of e-motif m in sequence s_x.

Sequence-maximal e-motif bicluster: An e-motif bicluster is sequence-maximal if it cannot be added more sequences while maintaining the property referred in the definition of an e-motif.

Left/right-maximal e-motif bicluster: An e-motif bicluster is left/right-maximal if its e-motif cannot be extended by adding a symbol to its beginning/end without losing sequences.

Maximal e-motif bicluster: An e-motif bicluster is maximal if it is sequence-maximal and left- and right-maximal.

Node-occurrence: A node-occurrence of a model m is a triple (v, v_e, p), where v is a node in T, v_e is the number of operations needed to transform m into the string-label of v and $p \geq 0$ is a position in T such that: (i) If $p = 0$ we are at node v; (ii) If $p > 0$ we are in $E(v)$, the edge from $parent_v$ to v, in a point p between two symbols in $label(E(v))$ such that $1 \leq p \leq |label(E(v))|$.

AliBiMotif biclustering starts by building a generalised suffix tree T for the set of sequences S. After further preprocessing, namely to compute the number of leaves

$L(v)$ in the subtree rooted at every node v, T is used to spell all valid e-motif models based on two properties:

- all prefixes of a valid model are also valid models
- when $e = 0$, spelling a model leads to one node v in T such that $L(v) \geq q_s$.

When $e > 0$, spelling a model leads to nodes $v_1, ..., v_k$ in T for which $\sum_{j=1}^{k} L(v_j) \geq q_s$. Since occurrences of a valid model are nodes in T, they are called node-occurrences (see definition above). AliBiMotif identifies valid models by extending them while traversing T and reporting their corresponding node-occurrences. Consider a node v in T, together with: its parent node, $parent_v$; the edge between nodes $parent_v$ and v, $E(v)$, and its corresponding label, $label(E(v))$, and length, $|label(E(v))|$. Three lemmas support the extension of a model m to a new model m' obtained by concatenating m with a symbol α ($\alpha \in \Sigma$):

Lemma 3.12: *$(v, v_e, 0)$ is a node-occurrence of model $m' = m\alpha$, if and only if: $(v, v_e - 1, 0)$ is a node-occurrence of m (**Deletion**);*

*$(parent_v, v_e, 0)$ is a node-occurrence of m and $label(E(v)) = \alpha$ or $(parent_v, v_e, |label(E(v))| - 1)$ is a node-occurrence of m and $label(E(v))[|label(E(v))|]) = \alpha$ (**Match**);*

*$(parent_v, v_e - 1, 0)$ is a node-occurrence of m and $label(E(v)) = \beta \neq \alpha$ or $(v, v_e - 1, |label(E(v))| - 1)$ is a node-occurrence of m and $label(E(v))[|label(E(v))|]) = \beta \neq \alpha$ (**Substitution**);*

*$(parent_v, v_e - 1, 0)$ is a node-occurrence of m and $|label(E(v))| = 1$ or $(v, v_e - 1, |label(E(v))| - 1)$ is a node-occurrence of m and $|label(E(v))| > 1$ (**Insertion**).*

Lemma 3.13: *$(v, v_e, 1)$ is a node-occurrence of a model $m' = m\alpha$, if and only if:*

*$(v, v_e - 1, 1)$ is a node-occurrence of m (**Deletion**);*

*$(parent_v, v_e, 0)$ is a node-occurrence of m and $label(E(v))[1] = \alpha$ (**Match**);*

*$(parent_v, v_e - 1, 0)$ is a node-occurrence of m and $label(E(v))[1] = \beta \neq \alpha$ (**Substitution**);*

*$(parent_v, v_e - 1, 0)$ is a node-occurrence of m and $|label(E(v))| > 1$ (**Insertion**).*

Lemma 3.14: *(v, v_e, p), where $2 \leq p < |label(E(v))|$, is a node-occurrence of a model $m' = m\alpha$, if and only if:*

*$(v, v_e - 1, p)$ is a node-occurrence of m (**Deletion**);*

*$(v, v_e, p - 1)$ is a node-occurrence of m and $label(E(v))[p] = \alpha$ (**Match**);*

*$(v, v_e - 1, p - 1)$ is a node-occurrence of m and $label(E(v))[p] = \beta \neq \alpha$ (**Substitution**);*

*$(v, v_e - 1, p - 1)$ is a node-occurrence of m (**Insertion**).*

Biclustering algorithmic details: AliBiMotif biclustering (Algorithm 1) identifies and reports all maximal e-motif biclusters with at least q_m candidate blocks (minimum e-motif length) and appearing in at least q_s sequences in four steps:

Step 1: Compute all models corresponding to sequence-maximal and right-maximal e-motif biclusters using a generalised suffix tree built for the set of rewritten sequences (Procedure `computeRightMaximal-Biclusters`).

Step 2: Delete models identifying non left-maximal e-motif biclusters from the sequence-maximal and right-maximal e-motif biclusters computed in Step 1 (Procedure `deleteNonLeftMaximalBiclusters`).

Step 3: Delete models representing the same e-motif biclusters from the models corresponding to maximal e-motif biclusters computed in the previous steps (Procedure `deleteRepeatedBiclusters`).

Step 4: Report the structured motif and corresponding occurrences for all maximal e-motif biclusters (Procedure `reportMaximalBiclusters`).

Algorithm 1: AliBiMotif biclustering

Input : $\{S_1, ..., S_N\}$, Σ, e, q_s, q_m
Output: Maximal e-motif biclusters.

```
1 modelsOcc ← {}
2 computeRightMaximalBiclusters(Σ, e, q_s, q_m, {S_1, ..., S_N}, modelsOcc)
3 deleteNonLeftMaximalBiclusters(modelsOcc)
4 if e > 0 then
5 |  deleteRepeatedBiclusters(modelsOcc)
6 reportMaximalBiclusters(modelsOcc)
```

Biclustering Step 1. Computing sequence-right-maximal e-motif biclusters

Procedure `computeRightMaximalBiclusters` computes all valid models m together with their node-occurrences Occ_m corresponding to sequence-maximal and right-maximal e-motif biclusters. The following tasks are performed:

- Build a generalised suffix tree T for the sequences of candidate blocks. Use `addNumberOfLeaves` to calculate the number of leaves in the subtree of every node v. Use `addColorArray` to set the bits corresponding to the sequences present in the leaves of the subtree rooted at every node v.
- Define the node-occurrences of a model m as triples (v, v_e, p), where p is used to specify whether we are at node v ($p = 0$) or in an edge $E(v)$ between nodes v and $parent_v$ ($p > 0$). Node-occurrences are stored in Occ_m.
- Identify all valid models satisfying the quorum constraints and identifying sequence-maximal and right-maximal e-motif biclusters using `spellModels`.

Results are stored in a list, $modelsOcc$. This list contains triples $(m, seqOcc_m, numberOfSeqOcc_m)$, where m is the model, $seqOcc_m$ is an array of bits signaling the distinct sequences in the node-occurrences of m, Occ_m, and

$numberOfSeqOcc_m$ is the number of bits set to 1 in $seqOcc_m$, expressing the number of sequences where model m occurs.

Procedure computeRightMaximalBiclusters

Input : Σ, e, q_s, q_m
Output: $modelsOcc$, a list of $(m, Occ_m, seqOcc_m, numberOfSeqOcc_m)$

```
1  T ← constructGeneralizedSuffixTree({S_1, ..., S_N})
2  addNumberOfLeaves(T)
3  m ← ""
4  length_m ← 0
5  parent_m ← ""
6  numberOfSeqOcc_{parent_m} ← 0
7  Occ_m ← {(root(T), 0, 0)}
8  Ext_m ← {}
9  modelsOcc ← {}
10 if e = 0 then
11 |  foreach edge E(v) leaving node root(T) to node v do
12 |  |  if label(E(v))[1] is not a string terminator then
13 |  |  |  addSymbol(label(E(v))[1], Ext_m)
14 else
15 |  addColorArray(T)
16 |  foreach symbol α in Σ do
17 |  |  addSymbol(α, Ext_m)
18 spellModels(Σ, e, q_s, q_m, modelsOcc, T, m, length_m, Occ_m, Ext_m,
   parent_m, numberOfSeqOcc_{parent_m})
```

Procedure `spellModels` generates all possible models and identifies and stores their node-occurrences by traversing the generalised suffix tree. Upon completion of a given model m, it then recursively checks every model $m\alpha$ obtained after extending m using each of the remaining valid symbols α from the alphabet Σ in the list Ext_m. Procedure `modelHasQuorum` is used to check whether a model m satisfies the quorum constraints or not. Only valid models m corresponding to right-maximal e-motif bicluster are kept. Procedure `checkRightMaximality` is used to remove from $modelsOcc$ valid models m that can be extended with α and for which the resulting model, $m\alpha$ is also a valid model and has a set of node-occurrences, $Occ_{m\alpha}$, containing as many sequences as the set of node-occurrences of its parent model m, Occ_m. In this case, m does not identify a right-maximal e-motif bicluster, since its e-motif can be extended to the right with α without losing sequences. The sequences where m occurs and their number are obtained using procedures `computeSequences` and `numberOfSequences`, respectively. Col_m holds the bit vector representing the sequences where model m occurs.

Extensions of model m through the addition of a symbol α are performed by `"extendModelFromNode"` and `"extendModelFromBranch"`, whether they follow from a node v or from a position in the branch descending from the parent of v to itself.

Variables maS and miS store the maximum and minimum numbers of sequences in the node-occurrences of the model, respectively.

```
Procedure spellModels
  Input: Σ, e, q_s, q_m, modelsOcc, T, m, length_m, Occ_m, Ext_m, parent_m,
         numberOfSeqOcc_parent_m
1 seqOcc_m ← computeSequences(Occ_m)
2 numberOfSeqOcc_m ← numberOfSequences(seqOcc_m)
3 if length_m > 0 then
4 |   keepModel(q_s, q_m, m, length_m, Occ_m, numberOfSeqOcc_m, modelsOcc)
5 |   checkRightMaximality(modelsOcc, numberOfSeqOcc_m, parent_m,
  |   numberOfSeqOcc_parent_m)
6 if length_m < maximumModelLength then
7 |   foreach symbol α in Ext_m do
8 |   |   maS ← 0
9 |   |   miS ← ∞
10|   |   if α is not a string terminator then
11|   |   |   Col_mα ← {}
12|   |   |   if e > 0 then
13|   |   |   |   for i from 1 to N do
14|   |   |   |   |   Col_mα[i] ← 0
15|   |   |   Ext_mα ← {}
16|   |   |   Occ_mα ← {}
17|   |   |   foreach node-occ (v, v_e, p) in Occ_m do
18|   |   |   |   removeNodeOccurrence((v, v_e, p), Occ_m)
19|   |   |   |   if p = 0 then
20|   |   |   |   |   extendModelFromNode(T, (v, v_e, p), α, Occ_m, Occ_mα,
  |   |   |   |   |   Col_mα, Ext_mα, maS, miS, 0)
21|   |   |   |   else
22|   |   |   |   |   extendModelFromBranch(T, (v, v_e, p), α, Occ_m, Occ_mα,
  |   |   |   |   |   Col_mα, Ext_mα, maS, miS, 0)
23|   |   |   if modelHasQuorum(maS, miS, Col_mα, q_s) then
24|   |   |   |   spellModels(Σ, e, q_s, q_m, modelsOcc, T, mα, length_m + 1,
  |   |   |   |   Occ_mα, Ext_mα, m, numberOfSeqOcc_m)
```

```
Procedure extendModelFromNode
  Input: T, (v, v_e, p), α, Occ_m, Occ_mα, Col_mα, Ext_mα, maS, miS, l
1 if l = 0 and v_e < e then
2 |   extendModel(T, α, Occ_m, Occ_mα, (v, v_e + 1, p), Col_mα, Ext_mα, maS,
  |   miS, l, false)
3 if v is an internal node then
4 |   foreach child of node v, child_v do
5 |   |   if label(E(child_v))[1] is not a string terminator then
6 |   |   |   p_child_v ← 1
7 |   |   else
8 |   |   |   p_child_v ← 0
9 |   |   tryExtension(T, (v, v_e, p), child_v, p_child_v, β, α, Occ_m, Occ_mα, Col_mα,
  |   |   Ext_mα, maS, miS, l)
```

Two additional procedures complete the set of operations needed to spell valid right-maximal models: `tryExtension` and `extendModel`. Procedure `tryExtension` tries to extend a given model m with a symbol α, having β as

Procedure extendModelFromBranch

Input: T, (v, v_e, p), α, Occ_m, $Occ_{m\alpha}$, $Col_{m\alpha}$, $Ext_{m\alpha}$, maS, miS, l

if $l = 0$ ***and*** $v_e < e$ **then**
  extendModel(T, α, Occ_m, $Occ_{m\alpha}$, $(v, v_e + 1, p)$, $Col_{m\alpha}$, $Ext_{m\alpha}$, maS, miS, l, false)
if $|label(E(v))| > p$ ***and*** $label(E(v))[p+1]$ *is not a string terminator* **then**
  $p_{new} \leftarrow p + 1$
else
  $p_{new} \leftarrow 0$
tryExtension(T, (v, v_e, p), v, p_{new}, β, α, Occ_m, $Occ_{m\alpha}$, $Col_{m\alpha}$, $Ext_{m\alpha}$, maS, miS, l)

Procedure tryExtension

Input: T, (v, v_e, p), s, p_s, β, α, Occ_m, $Occ_{m\alpha}$, $Col_{m\alpha}$, $Ext_{m\alpha}$, maS, miS, l

if Occ_m *is not empty* **then**
  $(x, x_e, p_x) \leftarrow getNextNodeOccurrence(Occ_m)$
  if $x = s$ *and* $p_x = p_s$ **then**
    removeNodeOccurrence((x, x_e, p_x), Occ_m)
    $s_e \leftarrow$ findMinimumError(v_e, $v_e + 1$, $v_e + 1$, $x_e + 1$, β, α)
    if $s_e \leq e$ **then**
      extendModel(T, α, Occ_m, $Occ_{m\alpha}$, (s, s_e, p_s), $Col_{m\alpha}$, $Ext_{m\alpha}$, maS, miS, l, true)
    else
      removeChildrenNodeOccs(Occ_m, s, p_s)
    return
$s_e \leftarrow$ findMinimumError(v_e, $v_e + 1$, $v_e + 1$, β, α)
extendModel(T, α, Occ_m, $Occ_{m\alpha}$, (s, s_e, p_s), $Col_{m\alpha}$, $Ext_{m\alpha}$, maS, miS, l, false)

the actual symbol occurrence in the tree. `tryExtension` is also given the node-occurrence from which the model has to be extended, (v, v_{err}, p), and the position it will potentially reach in the tree, as the pair s, p_s. It considers four distinct extension operations: match, substitution, deletion and insertion. These only affect the error calculation procedure, `findMinimumError`. Procedure `removeChildrenNodeOccs` is used by `tryExtension` to recursively remove all node-occurrences from Occ_m which identify a position in the tree below the one given by the pair s, p_s. Extension is finally accomplished by procedure `extendModel`. Essentially, this procedure updates the relevant information for subsequent model extensions. It also calls either `extendFromNode` or `extendFromBranch` procedure when further extensions need to be considered.

Biclustering Step 2. Deleting models of non left-maximal e-motif biclusters

Step 2 of AliBiMotif biclustering approach uses Procedure `deleteNonLeftMaximalBiclusters` adapted from e-CCC-Biclustering (Madeira and Oliveira, 2009) with small changes to remove from the models stored in the list $modelsOcc$ those that do not correspond to left-maximal e-motif biclusters. Models are removed from $modelsOcc$ by first building a trie with the reverse patterns of all (right-maximal) models m and storing the number of sequences in $numberOfSeqOcc_m$ for every corresponding node. A non left-maximal node is defined as any node in the trie

```
Procedure extendModel
  Input: T, α, Occ_m, Occ_mα, Col_mα, (v, v_e, p), Ext_mα, maS, miS, l
         doRecursion
1  addNodeOccurrence((v, v_e, p), Col_mα)
2  updateMaxSeqMinSeq(maS, miS, v)
3  if e > 0 then
4  |  updateColors(Col_mα, v)
5  if v_e = e then
6  |  if p > 0 then
7  |  |  if label(E(v))[p + 1] is not a string terminator then
8  |  |  |  addSymbol(label(E(v))[p + 1], Ext_mα)
9  |  |  |  if doRecursion = true then
10 |  |  |  |  extendModelFromBranch(T, (v, v_e, p), α, Occ_m, Occ_mα,
   |  |  |  |  Col_mα, Ext_mα, maS, miS, l + 1)
11 |  else
12 |  |  if v is an internal node then
13 |  |  |  foreach child s of node v do
14 |  |  |  |  if label(E(s))[1] is not a string terminator then
15 |  |  |  |  |  addSymbol(label(E(s))[1], Ext_mα)
16 |  |  if doRecursion = true then
17 |  |  |  extendModelFromNode(T, (v, v_e, p), α, Occ_m, Occ_mα, Col_mα,
   |  |  |  Ext_mα, maS, miS, l + 1)
18 else
19 |  addAllSymbols(Σ, Ext_mα)
20 |  if doRecursion = true then
21 |  |  if p > 0 then
22 |  |  |  extendModelFromBranch(T, (v, v_e, p), α, Occ_m, Occ_mα, Col_mα,
   |  |  |  Ext_mα, maS, miS, l + 1)
23 |  |  else
24 |  |  |  extendModelFromNode(T, (v, v_e, p), α, Occ_m, Occ_mα, Col_mα,
   |  |  |  Ext_mα, maS, miS, l + 1)
```

that has at least one child with as many sequences in its node-occurrences as itself. Non left-maximal nodes are marked doing a depth-first traversal of the trie and computing, for each node, the maximum sequence count among all its children. Models whose corresponding node in the trie is marked as non left-maximal are finally removed from $modelsOcc$.

Biclustering Step 3. Deleting models representing the same e-motif biclusters

When approximate occurrences of a model are allowed, as in the case of AliBiMotif biclustering algorithm, distinct valid models may identify the same e-motif bicluster if their e-motifs belong to the same e-neighbourhood. Following the approach of e-CCC-Biclustering (Madeira and Oliveira, 2009), procedure `deleteRepeated-Biclusters` uses a hash table to remove all models that, although maximal, correspond to repeated biclusters. From all valid models sharing the same node-occurrences in $modelsOcc$, that is, occurring in the same subsets of sequences with

the same sets of initial and final positions (same blocks), a single representative model m is kept.

Biclustering Step 4. Reporting structured motifs

Reporting the structured motifs involves performing the following operations for each valid model: merging the overlapping blocks and reporting the resulting e-motifs and corresponding biclusters. The goal of the merging step is to extend the overlapping blocks resulting from the candidate block construction process in order to obtain a set of disjoint blocks. It is performed both in the components of the e-motif and in the occurrences of these in the sequences. After extension the e-motif becomes a structured motif with disjoint components. Each component is reported together with the initial and final positions in the sequences where it occurs.

3.3 Complexity analysis of AliBiMotif

Construction of candidate blocks (see Section 3.1) takes $O(N^2a^2) + O(Nfa)$ time, where N is the number of sequences, f is the number of seed fragments and a is the average sequence length in an unaligned region (Wang et al., 2007).

Regarding the biclustering approach for mining the structured motifs (see Section 3.2), the major procedures exhibit the following time complexities for a set of N sequences obtained using the aforementioned mapping (see Section 3.2.1) with total length L, after B candidate blocks have been identified by the candidate block construction step (B is the size of the alphabet for the rewritten sequences) and considering e maximum errors on the individual components of the e-motifs.

Valid models corresponding to right-maximal e-motif biclusters are computed using `computeRightMaximalBiclusters`, underlying $O(NLB^e)$ operations. This result is bounded by the complexity of procedure `spellModels`, used to spell all right-maximal valid models and identify the corresponding node-occurrences in $O(NLB^e)$, since the remaining operations are less costly. Both the construction of the generalised suffix tree and the computation of the number of leaves $L(v)$ in the subtree rooted at each node v are $O(L)$ using Ukkonen's algorithm (Gusfield, 1997; Ukkonen, 1995) and a depth-first traversal of the tree, respectively. When $e > 0$, adding the colour array to all nodes in T takes $O(NL)$ time and initialising Ext_m takes $O(B)$.

Non left-maximal e-motif bicluster models are removed by procedure `deleteNonLeftMaximalBiclusters` in $O(lLB^e)$ time, where l is the length of the longest sequence, as the number of right-maximal models in $modelsOcc$ after applying `computeRightMaximalBiclusters` is $O(LB^e)$ and the size of each model is bounded by $O(l)$. Note that, as each candidate block can occur in a sequence at most once, the length of the longest sequence is bounded by the number of identified candidate blocks, $l \leq B$.

Procedure `deleteRepeatedBiclusters` takes $O(NlLB^e)$ to remove the maximal models representing the same e-motif biclusters, since computing the hash key based on the sequences and blocks for each of the $O(LB^e)$ e-motif bicluster models is $O(Nl)$. The quantity Nl is always greater or equal than the total length of the sequences L, $Nl \geq L$.

Overlapping blocks in the maximal e-motif bicluster models are merged in $O(NlLB^e)$. Reporting the resulting structured motifs takes $O(NlLB^e)$ time, as

the number of models is bounded by $O(LB^e)$ and the numbers of distinct sequences and blocks in each model are $O(N)$ and $O(l)$, respectively. Procedure `reportMaximalBiclusters` then takes $O(NlLB^e)$ and the overall complexity of the biclustering approach is thus $O(NlLB^e)$.

4 Conclusions

We propose AliBiMotif, a method to unravel structured motifs in multiple DNA sequences. This approach to motif finding efficiently identifies all maximal structured motifs satisfying the following constraints: ordered sets of at least q_m potentially overlapping conserved regions occurring approximately in at least q_s sequences with a maximum of e errors, where each conserved region has length of at least k and exhibits a mininum degree of conservation t across the spanned sequences. Locally conserved regions are extracted using pairwise sequence alignment and then agglomerated into larger sets, called candidate blocks, using a greedy clustering method (Wang et al., 2007). The original sequences are rewritten as the ordered sets of their underlying blocks. Structured motifs are subsequently identified and reported using a biclustering approach inspired on e-CCC-Biclustering (Madeira and Oliveira, 2009).

Construction of candidate blocks takes $O(N^2a^2) + O(Nfa)$ time, where N is the number of sequences, f is the number of seed fragments and a is the average sequence length in an unaligned region (Wang et al., 2007). The overall time complexity of the biclustering step is $O(NlLB^e)$, where l is the length of the longest mapped sequence, L is the sum of the lengths of the mapped sequences and B is the number of identified candidate blocks, corresponding to the alphabet of the mapped sequences. In this case, $l \leq B$ and $Nl \geq L$.

AliBiMotif presents a number of advantages relative to existing motif finders. First, it requires a fairly limited number of parameters when compared with alternative approaches. Second, the aforementioned thresholds enable a reasonable relaxation of the constraints imposed on the configuration of the structured motifs. In particular, the minimum degree of conservation allows for deletions, insertions and substitutions to be considered in the case of each component, nevertheless restricting the number of gaps. Moreover, although the size of the candidate blocks must be prespecified, the length of each component of the structured motif is maximised by considering overlapping blocks later merged into larger conserved regions. Furthermore, the quorums allow for a straightforward filtering of uninteresting structured motifs, namely the ones occurring in a reduced number of sequences. In addition, the maximum number of errors allows one to control the number of insertions, deletions and substitutions that can be considered in the ordered sequences of blocks composing the structured motif, consequently limiting the total size of the unravelled structured motifs. Third, our mapping from the original sequences and candidate blocks to the structured motif mining formulation prevents the biclustering algorithm from processing the non-conserved regions, which further enables the distance between the components of the structured motif to be disregarded. This is of major relevance, since the length of the non-conserved regions is often hard to estimate, particularly when no prior knowledge is available.

Our approach mines potentially overlapping structured motifs. Such property is advantageous in the sense that the algorithm is able to unravel distinct TFBSs shared by a given sequence with several subsets of additional sequences, supporting the reasoning that a particular sequence can be bound to by different TFs to enable or repress its transcription depending on the biological processes in progress.

Currently, AliBiMotif uses the candidate block construction step outlined in Wang et al. (2007). However, it remains unclear if global alignment is the best strategy to successfully identify locally conserved regions. In fact, this is the main motivation for the existence of local alignment methods. Moreover, the greedy clustering approach used for candidate block construction does not lead to an optimal solution. In this context, it would be worth to consider replacing the combination of sequence alignment and greedy clustering by a biclustering strategy applied to the original sequences. A combinatorial biclustering method would guarantee all locally conserved regions to be unravelled. Moreover, it would maximise the length of each conserved region, avoiding the generation of an excessive and unnecessary number of overlapping candidate blocks. Nevertheless, such algorithm should be able to properly deal with both repetitions of symbols of the alphabet (nucleotides) and of the same motif in distinct positions in the sequences, in order to maintain the order of the identified conserved regions.

We do not present experimental results, as the paper focus is algorithmic. Envisioned future work includes the application of the method to unravel structured motifs in real data, a practical comparison with state of the art structured motif finders, and a publicly available implementation of AliBiMotif.

Acknowledgement

This work was partially supported by FCT (INESC-ID multiannual funding) through the PIDDAC Program funds and project ARN – Algorithms for the Identification of Genetic Regulatory Networks, PTDC/EIA/67722/2006. JPG is the recipient of a doctoral grant supported by FCT (SFRH/BD/36586/2007).

References

Bailey, T. and Elkan, C. (1994) 'Fitting a mixture model by expectation maximization to discover motifs in biopolymers', *Proceedings of the Second International Conference on Intelligent Systems for Molecular Biology*, AAAI Press, Stanford, California, USA, pp.28–36.

Buhler, J. and Tompa, M. (2002) 'Finding motifs using random projections', *Journal of Computational Biology*, Vol. 9, No. 2, pp.225–242.

Carvalho, A., Freitas, A., Oliveira, A. and Sagot, M-F. (2006) 'An efficient algorithm for the identification of structured motifs in DNA promoter sequences', *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, Vol. 3, No. 2, pp.126–140.

Eskin, E. and Pevzner, P. (2002) 'Finding composite regulatory patterns in DNA sequences', *Bioinformatics*, Vol. 18, No. Sup I, pp.S354–S363.

Gusfield, D. (1997) *Algorithms on Strings, Trees, and Sequences*, Computer Science and Computational Biology Series, Cambridge University Press, New York, USA.

Lawrence, C.E., Altschul, S.F., Boguski, M.S., Liu, J.S., Neuwald, A.F. and Wootton, J.C. (1993) 'Detecting subtle sequence signals: a Gibbs sampling strategy for multiple alignment', *Science*, Vol. 262, No. 5131, pp.208–214.

Madeira, S.C. and Oliveira, A.L. (2009) 'An efficient biclustering algorithm for finding genes with similar patterns in time-series expression data', *Algorithms for Molecular Biology*, Vol. 4, No. 8.

Marsan, L. and Sagot, M-F. (2000) 'Extracting structured motifs using a suffix tree – algorithms and application to promoter consensus identification', *Journal of Computational Biology*, Vol. 7, pp.345–354.

Mendes, N., Casimiro, A., Santos, P., Sá-Correia, I., Oliveira, A. and Freitas, A. (2006) 'MUSA: a parameter free algorithm for the identification of biologically significant motifs', *Bioinformatics*, Vol. 22, No. 24, pp.2996–3002.

Prelič, A., Bleuler, S., Zimmermann, P., Wille, A., Bühlmann, P., Gruissem, W., Hennig, L., Thiele, L. and Zitzler, E. (2006) 'A systematic comparison and evaluation of biclustering methods for gene expression data', *Bioinformatics*, Vol. 22, No. 9, pp.1122–1129.

Schug, J. and Overton, G. (1997) 'Modeling transcription factor binding sites with Gibbs sampling and minimum description length encoding', *Proceedings of the Fifth International Conference on Intelligent Systems for Molecular Biology*, AAAI Press, Halkidiki, Greece, pp.268–271.

Thakurta, D. and Stormo, G. (2001) 'Identifying target sites for cooperatively binding factors', *Bioinformatics*, Vol. 17, No. 7, pp.608–621.

Ukkonen, E. (1995) 'On-line construction of suffix trees', *Algorithmica*, Vol. 14, No. 3, pp.249–260.

Wang, S., Gutell, R. and Miranker, P. (2007) 'Bielustering as a method for RNA local multiple sequence alignment', *Bioinformatics*, Vol. 23, No. 24, pp.3289–3296.

Zhang, Y. and Zaki, M. (2006) 'EXMOTIF: efficient structured motif extraction', *Algorithms for Molecular Biology*, Vol. 1, No. 21.