Module identification using biological constraints

E. De Maria[1], M. Zantoni[1], A. Dovier[1], and A. Policriti[1,2]

[1] Dipartimento di Matematica e Informatica, Università di Udine
[2] Istituto di Genomica Applicata, Udine.

Abstract. One of the main issues of computational biology consists in the identification of short strings (motifs) that occur approximately in a set of longer strings. A challenging problem is the one of extracting sets of motifs (modules) that satisfy some constraints on the distance between composing motifs and appear in a significant portion of the string input data set. In this work, we encoded such a problem in CLP(FD) and we tested our program on different data sets, obtaining significant results.

Introduction

After the first major effort in sequencing, the Human Genome Project, many other genomes have been sequenced and the next big challenge is to build reliable maps the functional elements. The essence of this challenge is finding specific patterns in a vast amount of information, and it is therefore natural for the fields of biology and information technology to cooperate in creating solutions to meet it. The discovery/identification of short strings occurring approximately in a set of longer strings/sequences is a classic in today's computational biology. In our particular case we refer to these short strings as *motifs*. The field of motif discovery has seen, in recent years, a shift of attention from single motifs to sets of motifs that cooperate in regulating the expression of a gene. This has resulted in a new and more challenging objective. In biological terms, instead of finding single *Transcription Factor Binding Sites* (TFBS, genomic motifs responsible for the binding of Transcription Factors to Promoters and other regulative elements) we now want to tackle the problem of finding (constrained) collections of TFBSs modeled as composite motifs also known as *modules*.

In this paper we focus our attention on the problem of finding one or more modules common to (a significant portion of) a given set of strings, imposing and managing distance constraints on the collection of simple motifs generated inspecting the input.

1 Biological Background

An important concept behind this work is gene regulation. Regulation is the amount or timing of introduction of a functional product of a gene. The product can be anything from mRNA to a protein or even a modified protein. In this

text the term is used as a descriptor of the affinity to transcribe a given gene. An up-regulated gene has higher likelihood of being transcribed and is transcribed more often.

A Transcription Factor is a description of proteins that attach themselves to binding sites in a gene's promoter region and, thereby, regulate the transcription of the gene. For this reason the discovery of such functional elements in the non-coding regions of DNA can be very valuable. The regulation of a gene is much more complex than this description might suggest. Orientation and folding of the DNA are significant, among many other factors. Still the search and determination of transcription factor binding sites are two of the most important functional elements in any genome, for understanding regulation. TFBSs are usually short, around 5-15 base pairs (bp), and are frequently degenerate sequence motifs. As a result of this, potential binding sites can occur frequently by chance in large genomes of higher eukaryotes. This makes the functional elements even harder to find.

Predicting promoter elements or extracting their consensus sequence are important steps towards the global comprehension of the mechanisms undergoing genes co-regulation. In order to achieve this goal, it is necessary to take into consideration aspects like the correct combination and the precise spatial organization of the regulatory sites, as demonstrated in recent papers [6, 11]. The order and relative distances between the binding sites, thus, can no longer be considered negligible constraints, and whatever the method used to extract a consensus sequence is, the prediction of precise promoters structure cannot be considered completed unless more biological knowledge is used during the prediction [1, 5, 8, 9]. Transcription Factor (TF) molecules interact with each other and bind to DNA to establish a gene transcription signal. The number of different TFs in a module, the number of TFBS, the spatial constraints, the order of TFBS and relative strands of TFBS differ for different regulatory pathways.

It is possible to extend the concept of motifs searching from the single binding site approach to a broader, module-based approach, to identify groups of at most 3-4 different TFBS that are conserved in different co-regulated genes and that maintain a constant overall distance with respect to one another. More complex modules composed by several motifs are difficult to localize, as they might be distributed in a large portions of the DNA.

It is important to underline that the module search/determination is a follow-up of the motif search activity. As a matter of fact, the goodness of the obtained results is tightly linked to the accuracy of the motifs discovery model used.

In literature, the issue of motif discovery has been well studied [10, 2].

2 Finding a common substring

Finding a consensus sequence representing the best approximation of all the similar results that have been obtained by a search, is crucial in order to recover useful information from the examined (biological) sequences. The problem can be formulated as follows: find a substring or a "similar" subsequence that is

common to many of the strings in the set. We use the Hamming distance d_H to define the concept of "similarity" among substrings, even though our results can be easily adapted to a different notion of distance.

The problems we are interested in include the *Closest String Problem* (CStrP) and the *Closest Substring Problem* (CSStrP), with or without a threshold.

Initially, guided by the needs of genomic research, statistical approaches were used to give solutions for the CStrP. The problem had been previously studied because of its connection with the area of coding theory, where it was proved to be NP-hard [4]. The CSStrP models the more general situation where the strings that must be compared do not have the same length, and one wants to find just parts of the string that are similar. The CSStrP, being a generalization of the CStrP, can be easily shown to be NP-hard. In terms of parameterized complexity, the main results for the CSStrP is that it cannot be solved in polynomial time, even when the distance parameter is fixed [3]. This is expressed, in terms of parameterized complexity theory, by showing that the CSStrP is in the class W[1]-hard.

As said before, it is important to underline that this task produces the input for the module identification problem. As a matter of fact, the goodness of the obtained results in the following is linked to the accuracy of the motifs discovery model used.

The notion of module has been well formalized in [7] with the definition of *structured motif* and the notion of module introduced below is a natural adaptation to the case of discovery (as opposed to search).

The approach described in this paper is based on results obtained using the algorithm ScanPro [13, 14, 15], which in turn was presented using a constraint programming approach [12].

3 Finding sets of motifs

Let $\mathcal{F} = \{s_1, \ldots, s_\alpha\}$ be e a set of α strings, each one of length β or less, over an alphabet Σ (i.e. nucleotides, aminoacids,...). Let $K = \{1 \ldots \alpha\}$ and $H = \{1 \ldots \beta\}$. Let $\mathcal{MT} \subseteq \Sigma^\star$ be a set of finite strings (*motifs*) such that each $\mu \in \mathcal{MT}$ is a substring that occurs in one or more strings of $\mathcal{F}$. We define the *motif alphabet* Γ to be a set isomorphic to $\mathcal{MT}$.

Definition 1. *Let* $\mathfrak{M} : \Gamma \to \mathscr{P}(K \times H)$*, the* instance function*, be such that given a motif* $\mu \in \Gamma$*,* $\mathfrak{M}(\mu) = \{(k_1, h_1), \ldots, (k_t, h_t)\}$ *is a set of pairs of integer numbers representing the indices of elements of* $\mathcal{F}$ *where* μ *occurs (even with repetitions) and the relative positions.*

We denote by $(\mathfrak{M}(\mu))_1 = \{k_1, \ldots, k_t\}$ and $(\mathfrak{M}(\mu))_2 = \{h_1, \ldots, h_t\}$ the *projections* of $\mathfrak{M}(\mu)$, and by μ^{k_i,h_i} the occurrence of the motif μ in string k_i starting at position h_i (that is, $s_{k_i}[h_i \ldots |\mu| - 1] = \mu$).

A *module* is defined as an ordered sequence of characters of Γ that orderly occurs in a given input sequence and a pair of integers representing the minimum and maximum distance between two adjacent motifs in the considered structured motif. Let $\mathbb{N}^\perp = \mathbb{N} \cup \{\perp\}$.

Definition 2 (Module). *A* module ϕ *is a triple*

$$\left(\langle \mu_j \rangle_{j \in \langle 1,\ldots,J \rangle}, d_{min}, d_{max}\right) \in (\Gamma^\star \times \mathbb{N}^\perp \times \mathbb{N}^\perp),$$

such that for some $k \in ((\mathfrak{M}(\mu_j))_1 \cap (\mathfrak{M}(\mu_{j+1}))_1)$ *and all* $j < J$*, there exist* $(k,h) \in \mathfrak{M}(\mu_j)$ *and* $(k,h') \in \mathfrak{M}(\mu_{j+1})$*, for which*

$$d_{min} \leq |h' - h| \leq d_{max}.$$

Formalization of the problem

Our goal is to find all modules that occur in at least q different sequences of $\mathcal{F}$ and satisfy constraint on composing motifs and their relative distances. Each module occurrence $(\langle \mu_1 \mu_2 \ldots \mu_J \rangle, d_{min}, d_{max})$ is a subsequence built with the strings $\mu_1, \mu_2, \ldots, \mu_J$ (in this order) and satisfying $d \leq d_{min}$ and $D \geq d_{max}$, where d (resp. D) is the minimum (resp. maximum) distance between (starting points of) μ_j and μ_{j+1}.

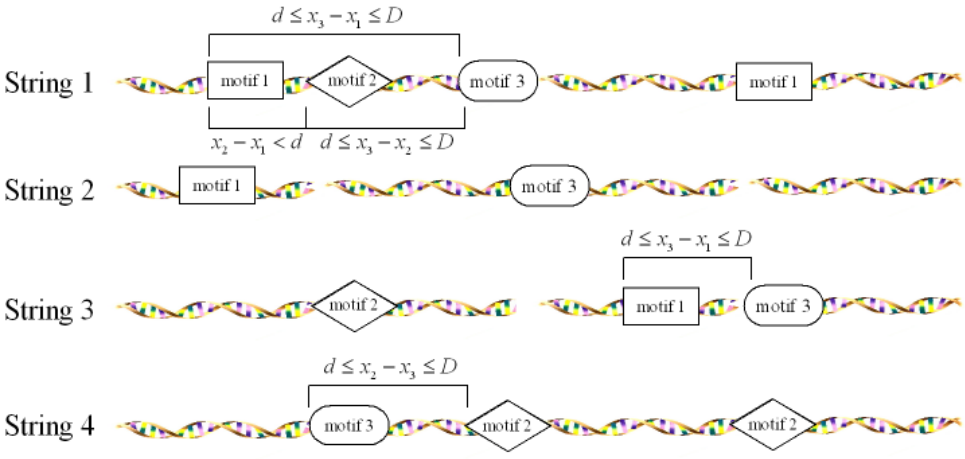

Fig. 1. For $q = 1$ (no constraints on the number of strings) there are 3 modules of length 2:[motif1, motif3], [motif2,motif3] and [motif3,motif2]. For $q = 2$ there is 1 module of length 2: [motif1, motif3]. For $q = 3$ there are no modules. [motif1, motif2, motif3] is not a module of length 3 because in String1 the distance between motif1 and motif2 is lower than d.

The complexity of the problem is dominated by $|\Gamma|^J$. If J is a constant, then the complexity of the decision test and of finding all the solutions is polynomial. Otherwise, if J is part of the input, $|\Gamma|^J$ grows exponentially on the inpout. In this case, finding all the solutions is inherently exponential. We have to figure out whether the associated decisional problem is NP complete or not.

4 Encoding in CLP(FD)

Given the alphabet Γ of motifs, a set $\mathcal{F} = \{s_1, \ldots s_k\}$ of strings, and a list holding the occurrences of each motif in each string, our aim is to find out all the possible modules of length t with occurrences above a given threshold q, i.e., modules with at least t Γ-characters that appear in a number of strings greater than or equal to q. Moreover, we require modules to satisfy minumum/maximum constraints on relative distances between composing motifs. Whenever we find a module, we want to count its occurrences in each one of the strings where it is present and to keep track of the positions where it occurs. We decided to encode this problem using Constraint Logic Programming over Finite Domains (in particular, SICStus 4 with clpfd).

The main predicate is `mod_search(+M, +Q, +Lb, +Lb)`, where `M` is the number of motifs present in the string set $\mathcal{F}$, `Q` is the lower bound on the number of strings where each module must appear, and `Lb` (resp. `Ub`) is the lower (resp. upper) bound on the distance between subsequent motifs in a module. `mod_search` retrieves from a file a list $L = \{l_1 \ldots l_k\}$ of lists containing the occurrences of each motif in each string. For each $1 \leq i \leq k$, list $l_i = \{\ell_{i1} \ldots \ell_{iM}\}$ is a list of lists containing the occurrences of each motif in string s_i. For each $1 \leq j \leq M$, list ℓ_{ij} contains the occurrences of the j-th motif in string s_i. As an example, list $L =$ [$[[1, 38], [24], [55, 70]]$, $[[12], [], [1, 25, 47]]$] gives the information that there are 2 strings and 3 motifs.
In the first string

- the first motif occurs in positions 1 and 38;
- the second motif occurs in position 24;
- the third motif occurs in positions 55 and 70.

In the second string

- the first motif occurs in position 12;
- the second motif does not occur;
- the third motif occurs in positions 1, 25, and 47.

Predicate `mod_search` definition mainly exploits two predicates, `callconstrain` and `allsolutions`. The first one looks for the presence of each structured motif of length 2 in the different strings of $\mathcal{F}$; the second one distinguishes between the structured motifs that appear in at least `Q` strings (modules) and the other ones. For each module (structured motif of the former type), it counts its occurrences in every string where it appears and it keeps trace of the positions where it occurs.

As far as predicate `callconstrain` is concerned, to perform the search of a single structured motif in a single string it takes advantage of predicate `onestring (+Lb, +Ub, +S, +[A,B], -V_AB , -M_AB, -N_AB)`, where `Lb` and `Ub` are the input lower and upper bounds, `S` is a list of lists, which contains the occurrences of each motif in a given string, `[A,B]` is a structured motif of length 2 made by motif `A` and motif `B`, `V_AB` is a boolean variable and `M_AB`, `N_AB` are

the positions where the structured motif occurs. Variable V_AB is set to 1 if in the given string there exists a pair of motif occurrence positions M_AB and N_AB such that Lb $\leq$ N_AB - M_AB $\leq$ Ub; otherwise V_AB is set to 0 and M_AB and N_AB are set to -1.

The definition of predicate onestring/7 is the following:

```
onestring(Lb,Ub,S,[A,B],V_AB , M_AB,N_AB):-
            nth1(A,S,L1), % L1 is the occurrences list of A in S
            nth1(B,S,L2), % L2 is the occurrences list of B in S
            list_to_fdset(L1,D1),
            list_to_fdset(L2,D2),
            M_AB in_set D1, % M_AB is the position of A in S
            N_AB in_set D2, % N_AB is the position of B in S
            N_AB-M_AB #>= Lb, N_AB-M_AB #=< Ub, !, V_AB #= 1.          (*)


onestring(_,_,_,[_,_],0 , -1,-1).
```

Since in the library clpfd of SICStus Prolog the domains of variables are internally represented as FD set terms, we use operation list_to_fdset to turn the list of occurrence positions of each motif in each string into a FD set. Then we take advantage of operation in_set to state the belonging of the occurrence position of each motif of the structured motif in each string in the proper FD set.

As far as predicate allsolutions is concerned, its inputs are the threshold Q and, for each structured motif, a list of boolean variables Bool and a list of M_AB and N_AB values (the length of Bool equals $|\mathcal{F}| = k$ because we associate a boolean variable to each string). Given a structured motif, if it appears in at least Q strings, that is, sum(Bools,#>=,Q), then we count its occurrences in each string where it is present; otherwise we avoid it. To count such occurrences and to keep trace of them, we use a standard mechanism based on assert and retract.
The source code is available at http://www.dimi.uniud.it/demaria/modules.html.

5 Results

We carried out several tests on an AMD Opteron 2.2 GHz Linux machine. First of all, we tested our program on a data set consisting in 26 strings of length 500 containing 23 motifs of length 6. Following biologists suggestions, we constrained the distance between subsequent motifs in a module to belong to the interval [10,90] and we looked for modules of length 2 appearing in at least q strings, with $q \geq 1$. The results of the test are presented in Figure 2. As the threshold q increases, the number of modules satisfying the constraint decreases. The minimum q such that there are not modules which satisfy the constraint is 15. As far as execution time is concerned, it decreases when q increases, that is, the number

of modules decreases. In fact, the less modules are, the less module occurrences to count are and the less `assert` and `retract` operations are needed.

Quorum	Modules	Time(ms)
1	385	410
2	275	350
3	198	300
4	143	270
5	117	240
6	94	250
7	60	190
8	39	190
9	24	190
10	12	180
11	8	150
12	3	170
13	1	180
14	1	170
15	0	150

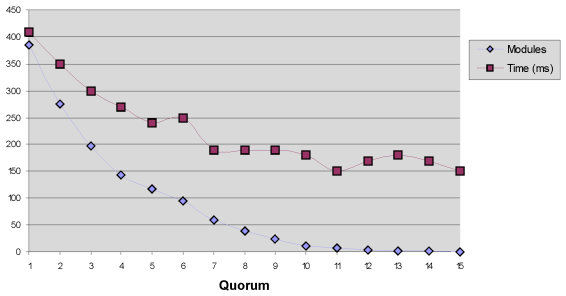

Fig. 2. Modules search on 26 strings of length 500. Decreasing of module number and execution time according to the growth of the quorum.

The results of such tests were analyzed by biologists who extrapolated relevant conclusions. In fact one of the input motifs, namely `GCAGNG`, was classified by biologists as an unknown one. Surprisingly our test showed that such a motif, a part from being abundant, is the first component of a module (`[GCAGNG,GCTGNG]`) that appears very often (in 11 strings). Such a result has a biological relevance because it means that the unknown motif appears very often in combination with other known motifs.

Another experiment consisted in testing our program on 7 data sets, each one made by 20 strings of length 500, 1000, 1500, 2000, 2500, 3000, and 3500 respectively and containing 15, 21, 27, 33, 39, 42, and 50 motifs respectively. In order to compare execution times, we launched our program on each data set with $q = 12$ and, as in the previous experiment, we constrained the distance between subsequent motifs in a module to belong to the interval [10,90]. The results are present in Figure 3. Time increases quickly because at each step we increase not only the strings length (and consequently the number of occurrences of different motifs in each string) but also the number of motifs.

At last, we tested our program on 10 data sets consisting respectively of 10, 20, 30, 40, 50, 60, 70, 80, 90 and 100 strings of length 500. Each data set contains 15 motifs. We launched our program with q equal to the number of strings divided by 2. The results are present in Figure 4.

String length	Motif number	Time(ms)
500	15	110
1000	21	650
1500	27	1960
2000	33	5100
2500	39	15800
3000	42	29170
3500	50	51450

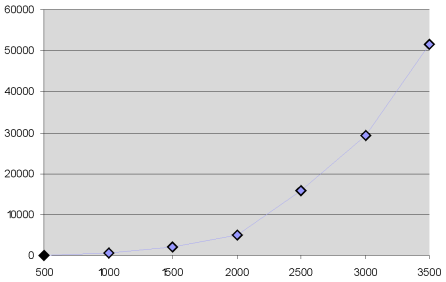

Fig. 3. Modules search on data sets of 20 strings with increasing length and motif number. Growth of execution time according to string length and motif number.

String number	Time(ms)
10	110
20	130
30	210
40	300
50	360
60	410
70	500
80	550
90	620
100	710

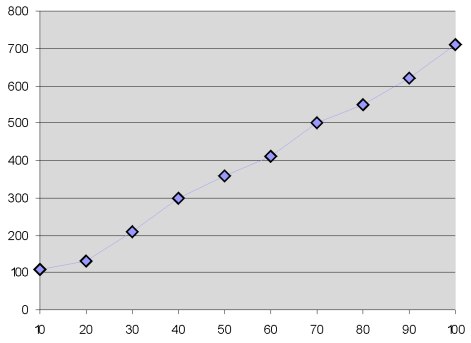

Fig. 4. Modules search on data sets with increasing number of strings. Growth of execution time according to the cardinality of the data set (string number).

6 Conclusions and Future work

Our program for extracting modules can be easily extended in several directions. First of all, it is immediate to generalize it to the research of modules of lengths greater than 2. Modules of length 3 could be thought as a combination of a motif with a module of length 2. Using such an approach, modules of length greater than 2 can be searched recursively. If we were asked to look for a restricted number of modules, we could improve the research by adopting a strategy that consists in considering at first motifs which occur in a greater number of strings, maximizing in such a way the chances to find modules which respect the quorum on the number of strings. We could start considering neighborhoods of radius greater than `Ub` of the occurrence positions of the most frequent motif, then look for occurrences of frequent motifs in these neighborhoods and so on.

Another possible extension concerns the constraints to impose when looking for a module in a set of strings. As an example, it would be reasonable to consider only modules that are sufficiently distant from the the beginning or the end of the string. As another example, biologists are interested in modules whose set of beginning positions in the different strings where they appear is limited by a lower and an upper bound. Such constraints can be very easily added to our program. After substituting the last line of predicate `onestring` (*) by the reified constraint

```
V_AB #<=>(N_AB-M_AB #>= Lb #/\ N_AB-M_AB #=< Ub),
```

one can add in predicate `allsolutions` explained in Section 4 constraints on the 2 lists containing respectively `V_AB` variables (one boolean variable is associated to each string) and `M_AB` and `N_AB` values.

Acknowledgments This work is partially supported by MIUR projects PRIN05-015491 and FIRB03-RBNE03B8KK.

References

[1] G.M. Church, A.M. Michelson, M.S. Halfon, and Y. Grad. Computation-based discovery of related transcriptional regulatory modules and motifs using an experimentally validated combinatorial model. *Genome Research*, 12:1019–1028, 2002.

[2] I. Eidhammer, I. Jonassen, S. H. Grindhaug, D. Gilbert, and M. Ratnayake. A constraint based structure description language for biosequences. *Constraints*, 6(2-3):173–200, 2001.

[3] M.R. Fellows, J. Gramm, and R. Niedermeier. On the parameterized intractability of closest substring and related problems. *Lecture Notes in Computer Science*, pages 262–273, 2002.

[4] M. Frances and A. Litman. On covering problems of codes. *Theor. Comput. Syst.*, 30:113–119, 1997.

[5] G. Kreiman. Identification of sparesely distributed clusters of cis-regulatory elements in sets of co-expressed genes. *Nucleic Acid Research*, 32, 2004.

[6] V.J. Makeev, A.P. Lifanov, A.G. Nazina, and D.A. Papatsenko. Distance preferences in the arrangement of binding motifs and hierarchical levels in organization of transcription regulatory information. *Nucleic Acids Research*, 31(20):6016–6026., October 2003.

[7] M. Morgante, A. Policriti, N. Vitacolonna, and A. Zuccolo. Structured motifs search. *Journal of Computational Biology*, 12(8), October 2005.

[8] B.D. Pfeiffer, P. Tomancak, S. Celniker, M. Levine, G.M. Rubin, M.B. Eisen, B.P. Berman, and Y. Nibu. Exploiting transcription factor binding site clustering to indentify cis-regulatory modules involved in pattern formation in the drosophila genome. *PNAS*, 99:757–762, 2001.

[9] R. Sharan, R. Shamir, Y. Shiloh, R. Elkon, and C. Linhart. Genomewide in silico identification of transcriptional regulators controlling the cell cycle in human cells. *Genome Research*, 13:773–780, 2003.

[10] R. Staden. Searching for patterns in protein and nucleic acid sequencies. *Methods in Enzymology*, 183:193–211, 1990.

[11] G. Terai and T. Takagi. Predicting rules on organization of cis-regulatory elements, taking the order of elements into account. *Bioinformatics*, 20(7):1119–1128, May 2004.

[12] M. Zantoni, E. Dalla, A. Policriti, and C. Schneider. Biological constraints for the consensus subsequence problem. In *Workshop on Constraint Based Methods for Bioinformatics (WCB05)*, 2005.

[13] M. Zantoni, E. Dalla, A. Policriti, and C. Schneider. Finding regulatory elements fixing error layouts. In *International Symposium on Computational Biology & Bioinformatics (ISBB)*, 2006.

[14] M. Zantoni, E. Dalla, A. Policriti, and C. Schneider. Motif discovery fixing mismatch positions. In *Communications to SIMAI Conferences, ISSN 1827-9015*, May 2006.

[15] M. Zantoni, E. Dalla, A. Policriti, and C. Schneider. TFBS discovery fixing layouts. Extended Abstract - RECOMB 2006, April 2006.