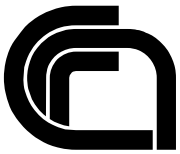

Consiglio Nazionale delle Ricerche

Composite Motif Finder: a combinatorial tool for finding cis-regulatory modules

M. Leoncini, M. Montangero, M. Pellegrini, K. Tillan

IIT TR-08/2012

Technical report

Luglio 2012

Istituto di Informatica e Telematica

Composite Motif Finder: a combinatorial tool for finding cis-regulatory modules

Mauro Leoncini
Dipartimento di Ingegneria dell'Informazione
Univ. di Modena e Reggio Emilia
Modena, Italy
leoncini@unimore.it

Manuela Montangero
Dipartimento di Ingegneria dell'Informazione
Univ. di Modena e Reggio Emilia
Modena, Italy
manuela.montangero@unimore.it

Marco Pellegrini
IIT-CNR
Via Moruzzi 1,
56124 Pisa, Italy
marco.pellegrini@iit.cnr.it

Karina Panucia Tillan
Dipartimento di Secienze e Metodi dell'Ingegneria
Univ. di Modena e Reggio Emilia
Modena, Italy
karina.panuciatillan@unimore.it

ABSTRACT

Controlling the differential expression of many thousands different genes at any given time is a fundamental task of metazoan organisms and this complex orchestration is controlled by the so-called *regulatory genome* encoding complex regulatory networks. *Cis-Regulatory Modules* (*CRMs*) are fundamental units of such networks. Intuitively CRMs are small stretches of DNA where several Transcription Factors bind so to perform in a cooperative manner a specific regulation task for nearby genes. The *in silico* prediction of CRMs is still an open problem, notwithstanding continuous progress and activity in the last two decades. In this paper we describe a new efficient combinatorial approach to the problem of detecting CRMs in promoter sequences, given in input a database of Transcription Factor Binding Sites encoded as Position Weight Matrices. Testing with benchmark data from TRANSFAC we attain significant better average performance against nine state-of-the-art competing methods over 12 data sets, even at high level of "noise" in the data sets.

Keywords

Cis-Regulatory Module prediction, evaluation benchmark, combinatorial algorithm.

1. INTRODUCTION

Transcription Factors (*TF*) are proteins that bind to short specific stretches of DNA (called *TFBS - Transcription Factor Binding Sites*) in the proximity of genes and participate in regulating the expression of those genes [4]. The "language" of gene regulation is a complex one since a single TF regulates multiple genes, and a gene is usually regulated over time by a cohort of cooperating TFs. This complex network of interactions is still far from being completely uncovered and understood even for well studied model species. Groups of TFs that concur in regulating the expression of groups of genes form functional elements of such complex network, and, also, are likely to have TFSB in the proximity of the regulated genes. This phenomenon of *clustering* of TFBS is used by exploratory "in silico" tools in order to predict the location and composition of *Cis-Regulatory Modules* (*CRMs*). A CRM is a stretch of DNA, usually of length ranging from a few tens to a hundred base pairs (bp), where a number of cooperating TFs can bind and regulate the expression of nearby genes [5]. A gene is usually associated to several CRMs [5].

TFBSs are often described by means of *Position Weight Matrices* (*PWMs*) (see Section 2 for a quick recap). Several hundreds of validated PWMs for identifying TFBS are available in databases such as TRANSFAC [32] and JASPAR [21]. Observe that, although these PWM have been subject to validation in some form, the highly degenerate nature of the TFBS implies that, when scanning sequences for PWM mathches, false positive non-functional matches are quite likely. Thus additional information and criteria are needed to filter them out.

Due to the importance of elucidating regulatory networks over the last two decades more than a hundred methods have been proposed for the prediction of single functional TFBS [18, 23, 9, 33], while several dozens for predicting functional CRMs [29]. However, Wasserman and Sandelin [31] observe that a great majority of computationally predicted TFBS arise from pure chance and are non-functional. One line of attack to overcome this difficulty is to use "unreliable" identification of single TFBS in order to detect, more robustly, putative CRMs, which in turn can increase the chance that

the constituent TFBS are indeed functional. A second line of attack (often used in conjunction with the first one) is to carefully select the sequences to be searched so that these are more likely to contain similar functional CRMs.

A survey of van Loo and Marynen [29] classifies the CRM prediction tools into three large families depending on the data set fed to the searching procedure.

CRM scanners. These methods scan a set of sequences (or a whole genome) trying to find CRMs based on a strict predefined model (i.e. matching a specified set of TFBSs defined by PWMs given as part of the input). They make use of libraries of known motifs and associated PWMs and of the propensity of TFBSs to clustering into CRMs.

CRM builders. These methods look for similar CRMs (without a pre-specified pattern) in a set of co-regulated or co-expressed genes. The matching PWMs can be derived from a library or can be generated internally by the tool, based on statistical over-representation of some sub-strings. The identified CRMs can be found in the regulatory regions of some or all of the given co-regulated genes.

CRM genome screeners. These methods do not make any assumption about the specific set of TFs working together, nor on the over-representation of some CRM over any other in the input sequence. They are the most general tools and are usually applied to analyze long sequences (even whole genomes) searching for clusters of TFBS, without predefined models.

Progress in the in-silico CRM prediction is hampered by lack of a common benchmark inclusive of test data and performance measures. To date the most comprehensive attempt in this direction for the case of single motif detection is due to Tompa and co-authors [27], while for the case of composite motifs (which are an abstraction of CRMs), Klepper et al. [14] propose the first benchmarking framework. In [14] test data extracted from TRANSFAC (and TRANSCompel [16]) are provided which are mostly from mouse, rat and homo sapiens. The test sets are made of sequence sets to be scanned and sets of PWMs for the TFBSs to be sought. Such lists of PWM also include, as "noise", a number of "decoy" matrices.

A thorough discussion of issues relative to benchmarking for TFBS and CRM finding tools, including alternative data sets and alternative approaches can be found in [15].

In this paper we present *Composite Motif Finder* (*CMF*) a new tool for cis-regulatory modules finding that adopts a two stage approach: first looks for candidate single TFBSs in the given sequences, and then use them to form modules by using mainly combinatoric (rather than statistic) arguments. CMF borrow some ideas from a previous tool we developed for a slightly different, but related problem [6]. Using the data set and the published data in [14, 22] we can readily compare CMF's performance against eight state of the art methods listed in [14] and another one presented in [22], showing that our tool is highly competitive with the others.

A direct comparison is possible also with other tools (*e.g.*, CORECLUST [17] and CREME [24]) and their experimental framework, that we plan to perform in the near future. However we feel that the initial hints obtained within the framework [14] give comforting evidence of the merits of the approach embodied in CMF.

The detection of TFBS and CRMs is a complex challenging problem (witness the ample spectrum of approaches) which is far from having a satisfactory definitive solution [30], thus there is ample scope for improvements both from the modelling point of view and from the algorithmic one. *Composite Motif Finder* introduces several new key ideas within a combinatorial overall approach, which, on one hand, have been shown empirically to be valid on challenging benchmarks, and on the other hand may prove useful in developing future more advanced solutions.

The rest of the paper is organized as follows: Section 2 introduces preliminary notions and definitions, Section 3 describes the algorithm adopted by CMF and, finally, Section 4 reports experimental results.

2. PRELIMINARY NOTIONS

In this Section we briefly define or recall the fundamental notions used in the rest of the paper.

Let $\mathcal{D} = \{A, C, G, T\}$ be the alphabet representing the 4 DNA base pairs (bp). A short word $w \in \mathcal{D}^*$ is called an *oligonucleotide*, or simply *oligo*. Typically $|w| \leq 20$. Let $\mathcal{S}$ be a set of DNA fragments, e.g., sequences of base pairs from the promoter regions of some genes. We say that w *occurs* in $S \in \mathcal{S}$ if and only if w is a substring of S.

From a computational point of view, a *DNA motif* (or simply *motif*) is a representation of a set of oligos, that are meant to describe possible Transcription Factor (TF) binding loci. The representation can be made according to one of a number of models presented in the literature. Here we adopt the well-known Position Weight Matrices (PWMs).

A PWM $M = (m_{b,j})$, $b \in \mathcal{D}, j = 1, \ldots, k$, is a $4 \times k$ real matrix. The generic element $m_{b,j}$ gives a score for nucleotide b being found at position j in the subset of length-k oligos that M is intended to represent. Scores are typically computed from frequency values. But how can we associate oligos to PWMs? Different answers have been given to this question in the literature (see, for instance, [12, 3, 20]). Here we adopt perhaps the simplest one.

Consider a word $w = w_1 w_2 \ldots w_k$ over $\mathcal{D}^k$, and define the *score* of w, according to M, simply as the sum of the scores of each nucleotides: $S_M(w) = \sum_{j=1}^{k} m_{w_j,j}$. The maximum possible score given by M to any word in $\mathcal{D}^k$ is clearly $S_M = \sum_{j=1}^{k} \max_{b \in D} m_{b,j}$. Then we say the M represents word w iff $\frac{S_M(w)}{S_M} \geq \tau$, for some threshold value $\tau \in (0, 1]$. In the following, we will identify motifs with their matrix representation.

A motif M has a *match* (or *occurrence*) in $S \in \mathcal{S}$ if and only if there is a substring of S that is represented by M. We borrow some terminology from [22] and call *discretization* the process of determining the matches of a motif in a set

of DNA sequences.

A *motif class* is a set of motifs. Ideally, in CMF all the motifs in a class describe potential binding sites for a single TF. For this reason, we often freely speak of TFs to refer to motif classes. A *match of a motif class* in a DNA sequence S, or simply a *TF match*, is a match of any of the motifs in that class. Note that motif classes have the ability to represent oligos of different lengths.

In CMF a *combinatorial group* (or just *group*) is a collection of not necessarily distinct TFs that have close-by matches in a sufficiently large fraction of the input sequences. The minimum fraction allowed for a set of TFs to be considered a combinatorial group is termed *quorum*. The *width* or *span* of a group match in a sequence S is the "distance" (measured in bps) between the first bps of first and last TF match of the group in S. In set-theoretic terms, groups are multisets. We consider the "customary" definitions of intersection and symmetric difference over multisets, which we recall using the following two simple examples:

$$\begin{aligned} \{x,x,y,y,y,z\} \cap \{x,y,y,w\} &= \{x,y,y\} \\ \{x,x,y,y,y,z\} \setminus \{x,y,y,w\} &= \{x,y,z\} \end{aligned}$$

Finally, a *Cis-Regulatory Module* (or simply *CRM*) is a set of close-by TF matches in some input sequence. CRMs represent CMF's *best guess* for functional TF binding regions. Note that no quorum constraint is imposed to CRMs; in fact, as groups of TF matches, CRMs are clearly unique objects. As we shall see in Section 3, CMF builds CRMs by "extending" group matchings.

3. ALGORITHM

CMF main operation mode is CMRs discovery in a set $\mathcal{S} = \{S_1, \ldots, S_N\}$ of DNA sequences, using a collection of PWMs. CMF is also able to run a number of third-party motif discovery tools, the output of which can then be used either to directly find potential CRMs or to "synthesize" PWMs, to be later used under the main operation mode.

PWMs can be passed to CMF in either a single or multiple files. In the latter case, CMF assumes that each file contains PWMs for only one given TF. Actually, when the input set is prepared using matrices taken from an annotated repository (e.g., the TRANSFAC database [16]), assuming the knowledge of the corresponding TFs is not an artificial scenario.

In this Section we briefly describe the four steps that implement CMF's main operation mode on input a single PWM file, leaving all the details to the full paper:

1. *PWM clustering*, to organize the matrices in equivalence classes;
2. *Discretization*, to detect potential PWM matches in the input sequences;
3. *Module finding*, which is the crucial CRMs detection step;
4. *Group and module filtering*, to filter candidates and pick, under the *Zero Or One Per Sequence* (*ZOOPS*) model the presumably best CRM in each sequence.

3.1 PWM clustering

By default, CMF assumes that the PWMs in the input file correspond to different TFs. Note, however, that the actual number (and identities) of TFs that bind to the input sequences is an information that may be available from the experimental protocols upstream data processing. As the number of PWMs may be much larger than the number of TFs[1], if the user explicitly provides the latter information, CMF performs a preliminary clustering step forming groups of PWMs that are assumed to correspond to different TFs.

The similarities among matrices, needed by clustering, are computed by the RSAT's utility `compare-matrices` [26]. CMF then builds a weighted adjacency graph whose nodes are the matrices and edges all the pairs (M_1, M_2) such that the similarity between matrices M_1 and M_2 is above a default threshold (set in `compare-matrices`).

CMF has two options for the actual clustering method. The first one is the well-known single linkage, which essentially reduces to a variation of Kruskal's algorithm for the construction of a maximum cost spanning forest. More precisely, let N_m be the number of matrices and let N_F be the number of TFs; then CMF performs at most $\min\{|\mathrm{E}|, N_m - N_F\}$ steps of Kruskal's algorithm, where E is the set of edges in the similarity graph. The clusters returned are the graphs induced by the vertices in distinct trees of the forest.

The second clustering option available in CMF is single linkage followed by a dense core computation performed on each subgraph produced by single linkage. Here again CMF uses external software, namely the *Pseudo-clique enumerator* (*PCE*) [28], that we chose since it is sufficiently light for the graph sizes we reasonably expect to handle.

If the PWM file essentially includes only "good" matrices corresponding to possibly different TFs, then even a simple clustering algorithm like single-linkage may be able to correctly separate them into the "right" groups. On the other hand, if the good matrices are mixed with noisy ones, then PCE refinement is likely to give better results. The first state of affairs may apply, for instance, when the input PWM file is the result of a judicious compilation from some annotated database. On the other hand, the second circumstance is likely to occur if CMF is used downstream motif discovery software tools.

3.2 Discretization

Even with the most accurate PWM description of a motif, the problem of determining the "true" motif matches in the input sequences is all but a trivial task. Whatever the algorithm adopted, there is always the problem of setting some thresholds to separate matches from non-matches, a choice that may have a dramatic impact on the tool's performance. We have already presented (Section 2) the discretization criterion adopted in CMF; what is still missing is just the description of the CMF's *threshold management*.

In general, low thresholds improve sensitivity while high thresholds may improve the rate of positive predicted val-

[1]This may happen if many matrices are loaded from an annotated database (possibly all of them), or when they are produced by running third-party tools.

ues (PPVs). As a preliminary observation, we note that the inherent combinatorics of CRMs may help to improve the PPV statistics (see next Section). A reasonable strategy is then to moderately privilege sensitivity during discretization, with the hope of increasing precision when detecting CRMs. As a second notice, we recall that for the purpose of predicting CRMs the important notion is that of TF match, rather than motif match. For TFs with many matrices, low thresholds may incur in a very high number of TF matches, with possibly high sensitivity but "very" low PPV. On the other hand, high thresholds may make it hard for a poorly represented class to emerge.

In light of the above argument, we formulate the following general and simple criterion: assign TFs (motif classes) with many/few matrices a high/low threshold. We are aware that the mathematical meaning of the previous statement may in turn be the subject of a debate. We did our best to keep the user away from such questions that regard (so called) *nuisance parameters* [10]. In CMF, the thresholds (as well as other values that may affect program's behavior) are kept in a configuration file; while the default values may clearly be changed, the ones used in the experiments described in Section 4 proved to give good results across a comprehensive benchmark. In particular, for what concerns the discretization thresholds, CMF starts from the default value set in the TAMO package (on which CMF in part builds), namely $\tau = 0.7$. However, this is quite strong and deemed appropriate only for very popular motif classes. Hence CMF reduces it to a minimum default value of 0.5 for poorly represented TFs (one or two PWMs). These assignments are completely transparent to the typical user.

As a final remark, we observe that lowering the thresholds may give the additional benefit of not filtering out low-score matches[2]. We have more on this point, though, and we will come back to it in Section 3.4.

3.3 Module finding

The previous two steps result in a set of TFs (motif classes) and a set of TFs matches, which are the "input" to the Module finding step. This is in turn divided into two main subprocesses: (a) finding combinatorial groups, and (b) extending groups to CRMs.

(a) To determine the combinatorial groups, CMF uses two internal parameters that are progressively relaxed until each TF is possibly included in at least one group. These parameters are the maximum allowed width W of CRMs and the minimum quorum q for combinatorial groups: $W \in \{W_1, W_2, \ldots, W_r\}$, with $W_1 < W_2 < \ldots < W_r$, and $q \in \{q_1, \ldots, q_s\}$, with $1 \geq q_1 > q_2 > \ldots > q_s > 0$. The values W_i and q_j are set in the CMF configuration files.

Let $N = |\mathcal{S}|$. At each iteration (i.e., for given values of W and q), CMF starts computing the maximal groups that respect the width constraint in each sequence: $M_1, \ldots, M_N$, where $M_i = \{m_1^{(i)}, \ldots, m_{n_i}^{(i)}\}$; each $m_k^{(i)}$ is in turn a pair $\langle m, v\rangle$, where $m \in \mathcal{R}^*$ is the group and $v = 1$ is the (initial) quorum. Note that, while the groups are multisets, each M_i is instead a set and thus does not include duplicate elements. Starting from $G = M_1$, CMF then computes the *Pairwise Intersection* set (PI) between G and all the other M_j:

$$\begin{aligned} G &= M_1 \\ G &\leftarrow PI(G, M_j), j = 2, \ldots N \end{aligned} \quad (1)$$

Pairwise intersection means that each element of G is intersected with each element of M_j. Note, again, that $PI(G, M_j)$ is a set, hence the possible duplicates are removed. The intersection I of $\langle m_1, v\rangle \in G$ and $\langle m_2, 1\rangle \in M_j$ is defined as follows using operations over multisets.

$$I(\langle m_1, v\rangle, \langle m_2, 1\rangle) = (\quad \langle m_1 \cap m_2, v+1\rangle, \; \langle m_1 \setminus m_2, v\rangle, \; \langle m_2 \setminus m_1, v\rangle \quad)$$

After intersection, groups that are not maximal with respect to at least one between TFs composition and quorum are immediately discarded. Also discarded are those maximal groups that can not possibly satisfy the minimum quorum constraint. At the end of the iteration, CMF checks whether each TFs is represented in at least one group that satisfy the quorum constraint. In the affirmative case the subprocess stops. Otherwise, another iteration begins with either width or minimum quorum value (alternately) relaxed. If the termination condition is not satisfied (not even with the weakest parameters), then some TFs will not be represented in the computed groups.

(b) Given a group G with quorum v, CMF first retrieves its actual matches in v sequences. Then tries to extend each match by possibly including other TFs matches that satisfy the width constraints. This is done independently for each sequence. All these extended group matches form the CRMs that CMF gives in output under the ANR (Any NumbeR) model.

We will say more about the complexity of the module finding step in Section 3.5. Here we observe, however, that pairwise intersections are performed in a highly efficient way as string comparison operations. In fact, before the group finding process starts, the n_F TFs produced by the clustering step are mapped to an (arbitrary) n_F-letter alphabet $\mathcal{R}$, and the maximal groups kept as character-sorted strings. For instance, if $\mathcal{R} = \{A, B, C, \ldots\}$, then we might have groups like, e.g., $AAABB$, ACC, and so on.

3.4 Group and module filtering

Under the ZOOPS model, groups and modules are further filtered, in order to pick the "best" instance in those sequences where there is more than one possible answer. We have three filtering phases.

The first filter can be termed "group filtering" as it aims at eliminating those group matches that are not "strong enough". As outlined in Section 3.2, a weak TF match may be allowed, due to possibly low thresholds for poorly represented TFs. At the group level, though, we impose stronger requirements, with the aims of both improving PPVs and possibly limiting the number of TFs matches, which highly affects the cost of the module finding step (see Section 3.5).

[2]Sometimes referred to as *weak signals* in the literature.

The maximum threshold τ specified in the CMF configuration file (which defaults to the value 0.7, as in TAMO) is here used to discard those group matches that exhibit less than $\tau \times 100$ percent of conserved bps over all TF matches of the group. A bp is conserved simply if it's the one that scores the highest in its PWM column.

The second filter works as follows. After discarding weak group matches, we perform a clustering of the remaining ones with respect to match widths and return the strongest match among the ones in the most popular cluster. Notice that there is some literature on the so called *structured motifs* (see, e.g., [19]). These are CRMs in which the order of the TF matches and (to some extent) the inner spacings are fixed. We do not use any of these pieces of information, but simply note that, if the spacings are fixed, then the width of the CRM is fixed (or exhibits small oscillations). This argument give some support to our choice of width clustering.

The third filter, of a statistic nature, uses p-values. CMF accept an input parameter that gives the maximum p-value acceptable for combinatorial groups, and which defaults to 0.01. The p-value of groups in CMS is computed exactly as specified in [22] for the modules.

3.5 Computational cost

The cost of the bare CMF algorithm is dominated by the module finding step or, more precisely, by the combinatorial group finding subprocess. This is easily seen to be exponential in the length of the longest group g (regarded as a string over $\mathcal{R}$) in any of the initial sets M_i's, simply because g may have an exponential number of maximal subgroups that satisfy also the quorum constraint. In turn, the length of g may be of the order of module width and hence of sequence length. This is clearly a hardly achievable worst-case. At the other extreme, there is the situation where we only have two TFs and look for sites where both TFs bind (as for the TRANSCompel dataset of Section 4). In this case the cost of the subprocess is linear in the number of sequences.

When combinatorial group finding is fast, which is often the case in practice, the computational bottlenecks move to other parts of the code, i.e., outside of the software module that implements the core CMF algorithm. In particular, the computation of PWM pairwise similarities takes quadratic time in the number of PWMs, which can be pretty high in a number of scenarios. To give some figures, on input the AP1-Ets dataset (see Section 4.1), which is the largest one of the TRANSCompel benchmark, CMF took 2m07s time in case of the noise_0 set of matrices[3]. The running time increased to full 33m in case of any of the noise_75 sets of PWMs, the vast majority of which (around 29m of total 31m increase) due to pairwise similarity computations. On the other hand, on muscle and liver datasets (that have few matrices) CMF took less than 10s. Note that, currently, similarities are computed by third-party software written in Perl. A substantial (practical) improvement would be obtained by simply recoding this module in a efficient compiled language (say, C++).

[3]The experiments were performed on a quite old i386 machine with 1GB of RAM running GNU/Linux with Kernel 2.6.32-5-686.

4. EXPERIMENTS

In this section we present the results obtained from a number of experiments performed on the benchmark datasets presented in [14] (see Section 4.1). In the following, we refer to [14] as to the *assessment paper*.

We compare CMF against the eight tools considered in the assessment paper (CisModule [34], Cister [7], Cluster-Buster (CB) [8], Composite Module Analyst (CMA) [13], MCAST [2], ModuleSearcher (MS) [1], MSCAN [11] and Stubb [25]). We also consider another tool, named COMPO, developed by the same research group that performed the assessment [22]. The results presented here show that CMF is competitive with existing discovery tools.

4.1 Datasets

We use both the TRANSCompel as well as the liver and muscle datasets presented in [14], which we briefly recall here. Together, they form a comprehensive benchmark that, to some extent, gives a non biased view of the relative merits of the various prediction tools.

The TRANSCompel benchmark includes ten datasets corresponding to as many modules, each consisting of two binding sites for different TFs from the following set: *AP1, Ets, NFAT, NFκB, CEBP, Ebox, AML, IRF, HMGIY, PU1*, and *Sp1*. Any dataset is named after the two TFs that contribute to the corresponding modules: AP1-Ets, AP1-NFAT, AP1-NFκB, CEBP-NFκB, Ebox-Ets, Ets-NFκB, NFκB-HMGIY, PU1-IRF, and Sp1-Ets. In [14], all the matrices corresponding to a same TF were grouped to form an "equivalence set", and treated as they were one. These matrices form what is called, in the assessment paper, the *noise_0 benchmark*.

To simulate conditions in which input data are fuzzier, the authors "generated" noisy PWMs (picking them at random from TRANSFAC), that were added to the noise_0 benchmark to give rise to the so-called *noise_N benchmarks*, for $N \in \{50, 75, 90, 95, 99\}$. For a given TF, noise_$N$ means a set containing as many as N% noisy matrices and (100-N)% "good" ones. However, noise_99 has a special meaning, namely that the set includes all the matrices in the TRANSFAC database. In the experiments performed here, we only consider noises $0, 50$ and 75. For each level of noise (except noise_0, of course), the assessment paper includes the ten different datasets (i.e., AP1-Ets, AP1-NFAT, etc), in which the good matrices are mixed with different noisy ones.

Two additional benchmarks are discussed in [14], each consisting of a single dataset, namely *liver* and *muscle*. Liver includes modules with up to nine binding sites from four different TFs, while muscle comes with up to eight sites from five TFs.

Each dataset has its own collection of (not necessarily distinct) input sequences, where occurrences of the ranging from a minimum of four (for the Ebox-Ets dataset) to a maximum of twenty-four (for the muscle dataset).

We obtained the statistics for the tools evaluated in [14] from `http://tare.medisin.ntnu.no/composite/composite.php`. Regarding COMPO, in [22] the authors provide statistics for liver and muscle only. However, they make available all

the *predictions* made by COMPO on input TRANSCompel data at `http://tare.medisin.ntnu.no/compo/`. We used such data to compute the statistics given here for COMPO.

4.2 Scoring predictions

Due to limited space, prediction accuracy is measured here by means of statistics at the *nucleotide-level*, leaving the analysis at *motif level* for the full paper. In particular, we compare CMF against all the other nine tools using the *correlation coefficient* (CC). We also compare CMF and COMPO (the best performing tool among CMF's competitors) using other popular statistics, namely: *Sensitivity* (Sn), *Positive Predicted Values* (PPV), *Performance Coefficient* (PC), and *Average Site Performance* (ASP). For a definition the reader may consult the assessment paper or [27].

Average results over different datasets are computed in two different ways (as in [14]). The *Combined score* is obtained by summing up the values of true/false positive/negative predictions over the union of the sequences corresponding to the datasets being combined. The *overall* score is simply the arithmetic mean of the single datasets' scores. For the noisy TRANSCompel benchmarks, combined scores are computed by averaging, over the ten noisy sets, the combined scores obtained on each set. The overall score is instead a double average: first the average, over the 10 noisy sets, of the measures obtained for single datasets, and then the arithmetic mean over the ten datasets.

4.3 Experiments and Results

In all the experiments, CMF was run with fixed configuration file, in particular with $W = \{100, 150, 200\}$, $q = \{80, 50, 20\}$ and minimum p-value=0.01 (see Section 3.3). However, for the TRANSCompel benchmark no filtering on p-value was specified (i.e., p-value=1), as we looked only for combinations of two different TFs (in case of liver and muscle there are instead hundreds of possible groups).

The first set of experiments involved only CMF over the TRANSCompel benchmark. For each dataset we performed two runs, one with matrices already separated by TF (i.e., giving CMF two PWM input files), and one with mixed matrices, only letting CMF know that the TFs involved are just two. Table 1 shows the results obtained.

By comparing the two columns, we gain some knowledge about CMF's ability to correctly infer the "true motif classes" (or TFs) from a set of mixed, but "good" matrices. In eight of the ten datasets, the output produced by CMS is exactly the same, implying that CMF recovered the two different TFs. On input the AP1-NFAT dataset CMF failed in both cases (confirming once more the hardness of this dataset [14]). Hence, only for one dataset (NFκB-HMGIY) CMF really failed because of poor TF separation. Poor separation also occurred on input the noise_50 and noise_75 datasets, if using the single linkage algorithm (results not shown). However, with the PCE refinement, the results improved again to a competitive level (Tables 3 and 4).

Tables 2 to 4 shows the results obtained by CMF compared to the best results from the the nine competitors for the three TRANSCompel benchmarks. In these experiments, all the PWMs were submitted to CMF in a unique file; in fact, this is certainly the way PWMs are provided to COMPO, and hence, for fair comparisons, we did not take advantage of CMF's ability to exploit the knowledge of TF memberships.

Dataset/CC	CMF Two PWM files	CMF One PWM file
AP1-Ets	0.428	0.428
AP1-NFAT	0.052	0.035
AP1-NFκB	0.665	0.665
CEBP-NFκB	0.736	0.736
Ebox-Ets	0.550	0.550
Ets-AML	0.767	0.767
IRF-NFκB	0.890	0.890
NFκB-HMGIY	0.533	0.075
PU1-IRF	0.804	0.804
Sp1-Ets	0.509	0.509
Combined CC	0.556	0.524
Overall CC	0.593	0.546

Table 1: CC values for noise_0 TRANSCompel data, when feeding CMF with two or one input files.

Dataset	CC	
	CMF	Best competitor
AP1-Ets	**0.428**	0.299 (MS)
AP1-NFAT	0.035	**0.151 (CMA)**
AP1-NFκB	**0.665**	0.585 (COMPO)
CEBP-NFκB	**0.736**	0.717 (CB)
Ebox-Ets	**0.550**	0.548 (COMPO)
Ets-AML	**0.767**	0.424 (COMPO)
IRF-NFκB	0.890	**0.912 (MSCAN)**
NFκN-HMGIY	0.075	**0.404 (MS)**
PU1-IRF	**0.804**	0.435 (MS)
Sp1-Ets	**0.509**	0.202 (Cister)
Combined CC	**0.524**	0.408 (COMPO)
Overall CC	**0.546**	0.387 (COMPO)

Table 2: CC results for noise_0 TRANSCompel data. Bold face figures highlight the best result among CMF and competitors MS = ModuleSearcher, CB = Cluster-Buster.

The results suggest that CMF is indeed competitive with other state of the art tools. On the vast majority of the datasets as well as in the cumulative statistics, CMF outperformed all the other tools. Among the competitors, it seems to us that the overall good behavior of COMPO deserves consideration.

In Table 5 we compare CMF and COMPO on a wider sets of statistics. Here a sort of complementary behavior among the two tools is worth noticing, as the noise level increases, with respect to the Sn and PPV statistics. In fact, while CMF's sensitivity exhibits a negative trend, this does not seem the case for COMPO. On the other hand, CMF seems better at keeping answer precision at an acceptable rate.

CMF behaved very well also on input the liver and muscle datasets, giving the best results on muscle and the third best

Dataset	CC	
	CMF	Best competitor
AP1-Ets	**0.427**	0.277 (MS)
AP1-NFAT	0.005	**0.133 (CMA)**
AP1-NFκB	0.375	**0.546 (COMPO)**
CEBP-NFκB	**0.724**	0.698 (COMPO)
Ebox-Ets	**0.550**	0.548 (COMPO)
Ets-AML	**0.641**	0.425 (COMPO)
IRF-NFκB	**0.890**	0.850 (MS)
NFκN-HMGIY	0.054	**0.404 (MS)**
PU1-IRF	**0.631**	0.433 (MS)
Sp1-Ets	**0.439**	0.155 (CMA)
Combined CC	**0.502**	0.392 (COMPO)
Overall CC	**0.474**	0.386 (COMPO)

Table 3: CC values, averaged over the ten noise sets, for noise_50 TRANSCompel data. CMA = Composite Module Analyst

Dataset	CC	
	CMF	Best competitor
AP1-Ets	**0.332**	0.277 (MS)
AP1-NFAT	0.039	**0.120 (CMA)**
AP1-NFκB	0.452	**0.546 (CMA)**
CEBP-NFκB	**0.724**	0.698 (COMPO)
Ebox-Ets	0.477	**0.488 (COMPO)**
Ets-AML	0.383	**0.394 (COMPO)**
IRF-NFκB	**0.851**	0.850 (MS)
NFκN-HMGIY	0.092	**0.404 (MS)**
PU1-IRF	**0.366**	0.296 (MS)
Sp1-Ets	**0.131**	0.120 (CMA)
Combined CC	**0.410**	0.404 (COMPO)
Overall CC	**0.362**	0.351 (COMPO)

Table 4: CC values, averaged over the ten noise sets, for noise_75 TRANSCompel data.

result on liver, as shown in Table 6. On these datasets, however, the observed performance differences between CMF and the other tools (with the exception of CisModules) are less significant.

5. CONCLUSIONS

In this paper we have presented CMF, a novel tool for Cis-Regulatory Module detection whose algorithmic core is based on purely combinatorial ideas. Using well-known benchmark data, our software proved to be highly competitive against nine state-of-the-art other tools.

Giving the good results obtained, we are encouraged to carry further activities on CMF, including the ones listed below.

- We plan to make further comparisons, including other tools as well as other experimental frameworks (e.g., the already mentioned CORECLUST [17] and CREME [24], to name a few), not only with the goal of better estimating CMF's value, but also with the aim at understanding its limitations, e.g., why it fails on input specific datasets, and how to possibly overcome them.

Dataset	Tool	PPV	Sn	PC	ASP	CC
TRANSCompel Noise 0	CMF	**0.50**	**0.60**	**0.37**	**0.55**	**0.52**
	COMPO	0.40	0.47	0.28	0.44	0.41
TRANSCompel Noise 50	CMF	**0.52**	**0.53**	**0.35**	**0.52**	**0.50**
	COMPO	0.37	0.48	0.26	0.42	0.39
TRANSCompel Noise 75	CMF	**0.44**	0.43	**0.28**	**0.43**	**0.41**
	COMPO	0.32	**0.45**	0.23	0.39	0.35

Table 5: A wider statistic comparison between CMF and COMPO

Tool	Dataset	
	Liver	Muscle
CMF	0.524	**0.563**
COMPO	0.560	0.470
Cluster-Buster	**0.588**	0.411
Cister	0.306	0.356
MSCAN	0.510	0.498
ModuleSearcher	0.425	0.463
MCAST	0.504	0.296
Stubb	0.476	0.243
CMA	0.358	0.462
CisModule	-0.005	0.289

Table 6: CC results for the Liver and Muscle datasets

- We plan to perform experiments also under such scenarios where the PWMs are produced (directly on indirectly) by third-party motif finding programs. Clearly, whether or not good results can be achieved here will largely depend on the quality of the external tools performance. However, our hope here is to exploit CMF's ability to filter out false positives (see Table 5) to achieve at least good PPV.

- Internally, we want to explore different similarity measures between PWMs and possibly different clustering algorithms, since in some cases poor predictions are caused by bad TF detection (as in case of the NFκB-HMGIY dataset, see Table 1).

6. ACKNOWLEDGMENTS

The present work is partially supported by the Flagship project *InterOmics* (PB.P05) which is funded and supported by the Italian MIUR and CNR organizations, and by the joint IIT-IFC Laboratory for Integrative System Medicine (LISM).

7. REFERENCES

[1] S. Aerts, P. Van Loo, G. Thijs, Y. Moreau, and B. De Moor. Computational detection of cis -regulatory modules. *Bioinf.*, 19(suppl 2):ii5–ii14, 2003.

[2] T. L. Bailey and W. S. Noble. Searching for statistically significant regulatory modules. *Bioinformatics*, 19(suppl 2):ii16–ii25, 2003.

[3] Q. K. Chen, G. Z. Hertz, and G. D. Stormo. Matrix search 1.0: a computer program that scans dna sequences for transcriptional elements using a

database of weight matrices. *Comp. appl. in the biosciences : CABIOS*, 11(5):563–566, 1995.

[4] E. H. Davidson. *The Regulatory Genome: Gene Regulatory Networks In Development And Evolution*. Academic Press, 1 edition, 2006.

[5] S. B.-T. de Leon and E. H. Davidson. Gene regulation: Gene control network in development. *Ann. Rev. of Biophysics and Biomolec. Struct.*, 36(1):191–212, 2007.

[6] M. Federico, P. Valente, M. Leoncini, M. Montangero, and R. Cavicchioli. An efficient algorithm for planted structured motif extraction. In *Proceedings of the 1st ACM workshop on Breaking frontiers of computational biology*, CompBio '09, pages 1–6, New York, NY, USA, 2009. ACM.

[7] M. C. Frith, U. Hansen, and Z. Weng. Detection of cis -element clusters in higher eukaryotic dna. *Bioinf.*, 17(10):878–889, 2001.

[8] M. C. Frith, M. C. Li, and Z. Weng. Cluster-Buster: Finding dense clusters of motifs in DNA sequences. *Nucl. Acids Res*, 31(13):3666–8, 2003.

[9] M. Häußler and J. Nicolas. Motif Discovery on Promotor Sequences. Research Report RR-5714, INRIA, 2005.

[10] J. Hu, B. Li, and D. Kihara. Limitations and potentials of current motif discovery algorithms. *Nucleic Acids Res*, 33:4899–4913, 2005.

[11] . Johansson, W. Alkema, W. W. Wasserman, and J. Lagergren. Identification of functional clusters of transcription factor binding motifs in genome sequences: the mscan algorithm. *Bioinf.*, 19(suppl 1):i169–i176, 2003.

[12] A. Kel, E. Göß ling, I. Reuter, E. Cheremushkin, O. Kel-Margoulis, and E. Wingender. Matchtm: a tool for searching transcription factor binding sites in dna sequences. *Nucleic Acids Research*, 31(13):3576–3579, 2003.

[13] A. Kel, T. Konovalova, T. Waleev, E. Cheremushkin, O. Kel-Margoulis, and E. Wingender. Composite module analyst: a fitness-based tool for identification of transcription factor binding site combinations. *Bioinf.*, 22(10):1190–1197, 2006.

[14] K. Klepper, G. Sandve, O. Abul, J. Johansen, and F. Drabløs. Assessment of composite motif discovery methods. *BMC Bioinformatics*, 9(1):123, 2008.

[15] K. Klepper, G. K. Sandve, M. B. Rye, K. H. Bolstad, and F. Drabløs. Benchmarking of methods for motif discovery in dna. In L. Elnitski, H. Piontkivska, and L. R. Welch, editors, *Advances In Genomic Seq. Anal. And Pattern Disc.*, chapter 6, pages 135–158. World Scientific Publishing Co, 2011.

[16] V. Matys, O. V. Kel-Margoulis, E. Fricke, I. Liebich, S. Land, A. Barre-Dirrie, I. Reuter, D. Chekmenev, M. Krull, K. Hornischer, N. Voss, P. Stegmaier, B. Lewicki-Potapov, H. Saxel, A. E. Kel, and E. Wingender. TRANSFAC and its module TRANSCompel: transcriptional gene regulation in eukaryotes. *Nucl. acids Res.*, 34:D108–D110, Jan. 2006.

[17] A. A. Nikulova, A. V. Favorov, R. A. Sutormin, V. J. Makeev, and A. A. Mironov. Coreclust: identification of the conserved crm grammar together with prediction of gene regulation. *Nucl. Acids Res.*, 2012. doi: 10.1093/nar/gks235.

[18] G. Pavesi, G. Mauri, and G. Pesole. In silico representation and discovery of transcription factor binding sites. *Briefings in Bioinf.*, 5(3):217–236, 2004.

[19] N. Pisanti, A. M. Carvalho, L. Marsan, and M.-F. Sagot. Risotto: Fast extraction of motifs with mismatches. In J. R. Correa, A. Hevia, and M. A. Kiwi, editors, *LATIN*, volume 3887 of *Lecture Notes in Computer Science*, pages 757–768. Springer, 2006.

[20] D. S. Prestridge. Signal scan: a computer program that scans dna sequences for eukaryotic transcriptional elements. *Comp. appl. in the biosc.: CABIOS*, 7(2):203–206, 1991.

[21] A. Sandelin, W. Alkema, P. G. Engström, W. W. Wasserman, and B. Lenhard. Jaspar: an open-access database for eukaryotic transcription factor binding profiles. *Nucl. Acids Res.*, 32:91–94, 2004.

[22] G. Sandve, O. Abul, and F. Drablos. Compo: composite motif discovery using discrete models. *BMC Bioinf.*, 9(1):527, 2008.

[23] G. K. Sandve and F. Drabløs. A survey of motif discovery methods in an integrated framework. *Biol. direct*, 1(1):11+, Apr. 2006.

[24] R. Sharan, I. Ovcharenko, A. Ben-Hur, and R. M. Karp. Creme: a framework for identifying cis-regulatory modules in human-mouse conserved segments. *Bioinf.*, 19(suppl 1):i283–i291, 2003.

[25] S. Sinha, E. van Nimwegen, and E. D. Siggia. A probabilistic method to detect regulatory modules. *Bioinf.*, 19(suppl 1):i292–i301, 2003.

[26] M. Thomas-Chollier, O. Sand, J. V. Turatsinze, R. Janky, M. Defrance, E. Vervisch, S. Brohee, and J. van Helden. RSAT: regulatory sequence analysis tools. *Nucleic Acids Research*, 36:W119–W127, 2008.

[27] M. Tompa, N. Li, T. L. Bailey, Church, and et al. Assessing computational tools for the discovery of transcription factor binding sites. *Nature Biotechnology*, 23(1):137–144, 2005.

[28] T. Uno. Pce: Pseudo clique enumerator, ver. 1.0, 2006.

[29] P. Van Loo and P. Marynen. Computational methods for the detection of cis-regulatory modules. *Briefings in Bioinf.*, 10(5):509–524, 2009.

[30] T. Vavouri and G. Elgar. Prediction of cis-regulatory elements using binding site matrices - the successes, the failures and the reasons for both. *Curr. Opinion in Genetics & Develop.*, 15(4):395 – 402, 2005.

[31] W. W. Wasserman and A. Sandelin. Applied bioinformatics for the identification of regulatory elements. *Nature Rev. Genet.*, 5(4):276–287, 2004.

[32] E. Wingender, P. Dietze, H. Karas, and R. Knüppel. Transfac: a database on transcription factors and their dna binding sites. *Nucl. Acids Res.*, 24(1):238–241, 1996.

[33] F. Zambelli, G. Pesole, and G. Pavesi. Motif discovery and transcription factor binding sites before and after the next-generation sequencing era. *Brief. in Bioinf.*, 2012.

[34] Q. Zhou and W. H. Wong. Cismodule: De novo discovery of cis-regulatory modules by hierarchical mixture modeling. *Proc. Nat. Ac. of Sciences USA*, 101(33):12114–12119, 2004.