Algorithms for phylogenetic footprinting *

Mathieu Blanchette
Department of Computer Science and Engineering
Box 352350
University of Washington
Seattle, WA 98195-2350 U.S.A.
206-543-5118
fax: 206-543-8331
blanchem@cs.washington.edu

ABSTRACT

Phylogenetic footprinting is a technique that identifies regulatory elements by finding unusually well conserved regions in a set of orthologous non-coding DNA sequences from multiple species. In an earlier paper, we presented an exact algorithm that identifies the most conserved region of a set of sequences. Here, we present a number of algorithmic improvements that produce a 1000 fold speedup over the original algorithm. We also show how prior knowledge can be used to identify weaker motifs, and how to handle data sets in which only an unknown subset of the sequences contain the regulatory element. Each technique is implemented and successfully identifies a large number of known binding sites, as well as several highly conserved but uncharacterized regions.

Keywords: Phylogenetic footprinting, motif finding, regulatory elements, phylogeny, algorithm.

1. INTRODUCTION AND MOTIVATION

A major challenge of current genomics is to understand how gene expression is regulated. An important step towards this understanding is the capability to identify regulatory elements (REs) associated with a given gene. Most of these regulatory elements are relatively short stretches of DNA (5 to 25 nucleotide-long), located in the non-coding sequence surrounding a gene. Most known transcription factor binding sites are located 5' of the coding region, but some are also found in the 3' sequence, and even in introns. In all these cases, REs are located in otherwise non-functional sequences.

* This material is based upon work supported in part by an NSERC fellowship, by the National Science Foundation and DARPA under grant DBI-9601046, and by the National Science Foundation under grant DBI-9974498.

Phylogenetic footprinting [28] is a technique that uses this functional/non-functional sequence dichotomy to identify regulatory elements. Functional sequences tend to evolve much slower than non-functional sequences, as they are subject to selective pressure. It is this difference in mutation rates that phylogenetic footprinting exploits. To identify regulatory elements associated with a given gene, one will consider a set of orthologous non-coding sequences from a group of related species (for example, the non-coding sequence located 5' of the insulin gene in ten different species of vertebrates). If these sequences contain unusually well conserved regions, it is a good conjecture that these regions have some regulatory function. This approach was used with success to identify REs in various genes: *TNF-α* [21], *CFTR* [30], *ϵ-globin* [13], and *rbcL* [22], among others. See the excellent review by Duret and Bucher [9] for more details.

This multi-species approach is to be contrasted with a much more common method for identifying regulatory elements, consisting of finding motifs in a set of non-coding sequences associated with *several* related genes from *a single* species. Frequently occurring patterns in these sequences are likely to correspond to binding sites of a common transcription factor. This approach has been used with success by a number of researchers and several algorithms and statistical analyses were developed (see, for example, Hughes *et al.* [18], Hertz and Stormo [15], Tavazoie *et al.* [29], Bailey and Elkan [5], Sinha and Tompa [27]). All these multi-gene approaches have an important inherent limitation: they will only find REs that are common to a number of genes of a given organism. Furthermore, the set of genes considered has to be obtained from experimental data (e.g. expression arrays). On the other hand, phylogenetic footprinting is capable of identifying REs that are specific to a single gene, as long as they are conserved across several species. The obvious downside is that orthologous non-coding sequences from a large number of species are not always available. In that sense, when phylogenetic footprinting was first proposed by Tagle *et al.* [28], it was a little bit ahead of its time. However, the various genome projects are quickly producing sequences from a wide variety of organisms, and the data needed to perform phylogenetic footprinting are becoming more and more available.

To our knowledge, until our recent work [7], phylogenetic footprinting was performed by computing the global multi-

ple alignment of the orthologous sequences, and by identifying highly conserved regions in the alignment. That approach led to significant discoveries, but is quite limited, in the following sense. First, multiple alignment is an NP-hard problem [31], so even if the optimal alignment could identify REs, we might not be able to compute it. More importantly, the upstream sequences considered are quite often highly diverged (except for their regulatory elements). Since regulatory elements are very short compared to the sequences considered (say 10 nucleotides long in a 1000 nucleotide long sequence), it is likely that the noise of diverged non-functional sequences will overcome the short, conserved signal, in such a way that the optimal alignment might not align REs together. Thus, multiple alignment is not really the tool of choice.

A different approach that is likely to yield better results consists in using classical motif finding algorithms like MEME [5], AlignAce [18] or Consensus [15]. To our surprise, it appears that none of these tools have been used for phylogenetic footprinting. Although we observe in this paper that these tools clearly outperform global multiple alignment, they still have an important shortcoming. Whether they optimize a consensus, sum-of-pairs, or information content criterion, none make use of the phylogenetic relationship of the given sequences. This can be problematic, for example in data sets containing a large number of closely related sequences, and a few more distant sequences. If we ignore the phylogeny underlying the data, the group of closely related sequences will have an unduly high weight in the solution.

In [7], we introduced the Substring Parsimony Problem (SPP), which is a formalization of the phylogenetic footprinting idea. We are given a set of orthologous sequences $S_1, ..., S_n$ from n different species, together with the phylogenetic tree T relating these species. The problem is to find a set of substrings $s_1, ..., s_n$ of $S_1, ..., S_n$ respectively, each substring being of size k, such that the parsimony score of $s_1, ..., s_n$ on T is minimized. The substrings $s_1, ..., s_n$ are the most conserved region of the given sequences, and if the conservation is sufficiently high, it is likely that these substrings are regulatory elements. We presented a practical algorithm that solves the SPP *exactly*, in time linear in $n \cdot l$, where n is the total number of sequences and l is their length, but in time exponential in k, the size of the motif sought. The SPP being NP-hard [1], [6], we can't hope to eliminate the exponential behavior of the algorithm. Still, we show in this paper that there is room for much improvement.

The contribution of this paper is twofold. In Section 2, we generalize the SPP by asking to find *all* sets of substrings for which the parsimony score is below a certain threshold d. We describe new algorithmic methods that improve the running time and space requirement by several orders of magnitude over the algorithm presented by Blanchette *et al.* [7]. That allows us to search for motifs of size more than 20, whereas we used to be limited to motifs of size 10. In Section 3, we present a number of extensions to the algorithm. The first one is to allow insertions and deletions in the computation of the parsimony score, which makes the algorithm presented here a full-fledged algorithm for local multiple alignment on a tree. We then show how to use our prior knowledge about sequence evolution to be able to find weak signals that would be lost otherwise.

The more species are available, the better we are able to detect subtle signals. However, in some cases, some REs might have been lost during the evolution of some species. A solution to the SPP consists of one substring per given sequence, which means that if one or more of the species have lost the use of a RE, we would most likely not find it. In Section 3.2 we relax this constraint by allowing sequences not to participate in the solution. This allows us to find REs conserved in only a subset of the input sequences.

With each algorithm and extension, we report motifs found in various sets of orthologous sequences. We show that we are able to identify a large number of binding sites that have been experimentally verified. For example, most of the known binding sites in *β-actin*, *insulin* and *metallothionein* are identified by our algorithm using sequences from various vertebrates. Our various extensions allow us to identify 5 of the 6 known REs of *rbcL*, using sequences from 50 different species of plants. We also identify a number of highly conserved regions upstream of genes for which no REs are known yet. The sequences downstream of several genes also contain many highly conserved regions, most of which have not been characterized yet.

2. THE SUBSTRING PARSIMONY PROBLEM

The Substring Parsimony Problem is simply a formalization of the phylogenetic footprinting idea:

SUBSTRING PARSIMONY PROBLEM (SPP)

Given: a set of orthologous sequences $S_1, ..., S_n$ from n different species, the phylogenetic tree T relating these species, the size k of the motifs to look for, and an integer d.

Problem: find all sets of substrings $s_1, ..., s_n$ of $S_1, ..., S_n$ respectively, each of size k, such that the parsimony score of $s_1, ..., s_n$ on T is at most d.

Recall that the parsimony score of a set of sequences is the minimum total number of substitutions over the tree T needed to explain the observed sequences. It is computed as the minimum, over all possible labelings of the internal nodes with sequences of size k, of the sum of the Hamming distances between the labels of nodes connected by an edge in T. Looking for sets of substrings that achieve a low parsimony score corresponds to searching for highly conserved regions.

2.1 A dynamic programming algorithm

In Blanchette *et al.* [7], we show that the SPP can be solved optimally by a dynamic programming algorithm similar to that proposed by Sankoff and Rousseau [26] for the computation of the parsimony score of a *fixed* set of strings. The algorithm assumes a rooted tree, so we will root our tree arbitrarily at an internal node r (this will not affect the solution). The algorithm then proceeds from the leaves up to the root. At each node u of the tree, we compute a table W_u containing 4^k entries, one for each sequence of size k. For a string s of size k, we define $W_u[s]$ as the best parsimony

score that can be achieved for the subtree rooted at u, if u was to be labeled with s (i.e. if we force the ancestral sequence at u to be s). Let us denote by $C(u)$ the set of children of u, let $d(s,t)$ be the Hamming distance between sequences s and t, and let $\Sigma = \{A, C, G, T\}$. The tables W can be computed recursively:

$$W_u[s] = \begin{cases} 0 & \text{if } u \text{ is a leaf and } s \text{ is a substring of } S_u \\ +\infty & \text{if } u \text{ is a leaf and } s \text{ is not a substring of } S_u \\ \sum_{v \in C(u)} \min_{t \in \Sigma^k}(W_v[t] + d(s,t)) & \text{if } u \text{ is not a leaf} \end{cases}$$

Then, the score of the optimal solution to the SPP is given by $min_{s \in \Sigma^k}(W_r[s])$. From that point, the ancestral sequences $s_{n+1}, ..., s_{|V|}$ and substrings $s_1, ..., s_n$ can be recovered by tracing back the recurrence, from the root down to the leaves, for each entry of W_r with score at most d.

2.2 Improved algorithm

The straight-forward implementation of this recursion runs in time $O(n \cdot k \cdot (4^{2k} + l))$, where l is the average length of the input sequences $S_1, ..., S_n$. The 4^{2k} factor in the complexity of the algorithm makes it impossible to use for most interesting values of k. In Blanchette *et al.* [7], we show how this recursion can be computed in time $O(n \cdot k \cdot (4^k + l))$, which makes it practical for interesting purposes. Here, we review the improved algorithm.

We will need an auxiliary table $X_{(u,v)}$ for each edge (u,v) in the tree, where u is the parent of v. We define $X_{(u,v)}[s]$ as the best parsimony score that can be achieved for the subtree consisting of u and the subtree rooted at v, if u was to be labeled with s. $X_{(u,v)}$ is obtained from W_v: $X_{(u,v)}[s] = min_{t \in \Sigma^k}(W_v[t] + d(s,t))$.

Notice that once we have computed $X_{(u,v)}[s]$ for each string s and each child v of u, we obtain $W_u[s] = \sum_{v \in C(u)} X_{(u,v)}[s]$. In Blanchette *et al.* [7], we show that the table $X_{(u,v)}$ can be computed in $O(k \cdot 4^k)$ instead of $O(k \cdot 4^{2k})$ time. We now review how this can be done. The idea is similar to a breadth first search of the space of sequences of size k. The search will be divided into phases. At each phase, we will consider a set B of sequences called the boundary. At phase p, the boundary B_p will contain exactly the sequences t such that $X_{(u,v)}[t] = p$. Let $R_a = \{s : W_v[s] = a\}$, let $m = min_{s \in \Sigma^k}(W_v[s])$, and let $N(s) = \{t \in \Sigma^k : d(s,t) = 1\}$. We start at phase m and set $B_m = R_m$. To go from phase p to phase $p+1$, we have

$$\begin{aligned} B_{p+1} &= R_{p+1} \cup \{s \in \Sigma^k : \exists t \in B_p \ s.t. \ s \in N(t)\} - \bigcup_{j \leq p} B_j \\ &= R_{p+1} \cup \bigcup_{s \in B_p} N(s) - \bigcup_{j \leq p} B_j \end{aligned} \quad (1)$$

We continue this boundary expansion until the boundary is empty, at which point all sequences will have been visited and all $X_{(u,v)}[s]$ will have been computed. Since each sequence s has exactly $3k$ neighbors and is part of the boundary only once, the computation of $X_{(u,v)}$ is done in $O(k \cdot 4^k)$ time, and the whole algorithm runs in $O(n \cdot k \cdot (4^k + l))$.

2.3 Sibling bounds

We now present new techniques that greatly reduce the computation time by avoiding the computation of useless entries in the W and X tables. First, notice that if the only solutions of interest are those with parsimony score at most d, there is no need to compute any W_u entries that will have scores above d, as $W_u[s] > d$ implies that any global solution in which u is labeled with s will have score above d. That means that computing entries of $X_{(u,v)}$ with score above d is also useless, and so one can stop the boundary expansion after phase d. This simple observation improves the complexity of the algorithm to $O(n \cdot min(l \cdot (3k)^d, k \cdot (4^k + l)))$. It also reduces the space requirement, as W and X tables will now be very sparse, allowing us to efficiently use hash tables to store them. We will refer to this technique as d-bounding.

We now show how this idea can be pushed further to actually perform only $d/2$ unconstrained expansion phases. Remember that the $X_{(u,*)}$ will eventually be added to form W_u. So far, most entries in the $X_{(u,*)}$ tables end up being rejected because they added up to more than d. To avoid that situation, we are going to compute the entries of the $X_{(u,*)}$ tables in parallel: first do phase zero in all tables, then phase one, etc. Then, after phase p, we know that if an entry $X_{(u,v)}[s]$ has not been computed yet, it must have a score of at least $p+1$. Thus, after phase p, we have the following bound:

$$W_u[s] \geq \sum_{v \in C(u)} \begin{cases} X_{(u,v)}[s] & \text{if } X_{(u,v)}[s] \text{ has been computed} \\ p+1 & \text{otherwise} \end{cases}$$

That means that at phase p, an entry s for which $X_{(u,v)}[s]$ is more than d can't have $W_u[s] \leq d$. Unfortunately, it is not true that entries failing this test can be omitted from the boundary expansion at the next phase. Indeed, it is possible that an entry s for which this bound is larger than d could have a neighbor, visited in the next phase, for which the bound is less than or equal to d. Still, one can greatly improve the boundary expansion computation by noticing that at phase p, an entry s of $X_{(u,v)}$ can only lead to entries with score at most d if

$$d \geq p + \max_{\substack{w \in C(u) \\ w \neq v}} \begin{cases} X_{(u,w)}[s] & \text{if } X_{(u,w)}[s] \text{ has been computed} \\ p+1 & \text{otherwise} \end{cases}$$

That means that any entry s for which this bound is not satisfied can be left out of the boundary. For phases $0...d/2-1$, this bound is useless because it is always satisfied. However, for phases $d/2...d$, it greatly reduces the size of the boundary considered. Indeed, at phase $d-i$, only those entries that have score at most i in all the siblings will be considered in the boundary. Thus, the size of the boundary at phase $d-i$ will be bounded by the size of the intersection of the boundaries of the siblings at phase i. Consequently, only $d/2$ unconstrained expansion phases are peformed , and the overall time complexity of the algorithm is reduced to $O(n \cdot min(l \cdot (3k)^{d/2}, k \cdot (4^k + l)))$. We will refer to this technique as sibling-bounding.

2.4 Parent bounds

The improvement above was obtained by computing the $X_{(u,v)}$ tables simultaneously for all children v of u, and use

those to constrain what entries s can lead to solutions with score at most d. We can push this idea one step further by using not only the siblings, but also the parent, to reduce the number of entries computed. To do that, we will need to compute in parallel all $X_{(u,v)}$ tables of all edges (u,v) in the entire tree, phase by phase. Furthermore, each non-terminal edge (v,u), where v is the parent of u, will have a new table associated with it: $Y_{(v,u)}[s]$ = Best parsimony score achievable for the subgraph ($(T-$subtree rooted at $u) \cup u$), if u was to be labeled with s. Recall that the solutions to the SPP are independent from the node at which the tree is rooted. So, when computing entries of $X_{(u,*)}$, we will temporarily re-root the tree at u, so that $T -$ (subtree rooted at u) becomes one more child of u. The table associated with this new subtree is $Y_{(v,u)}$, and this extra child can be used to constrain the computation of $X_{(u,*)}$ as in Section 2.3, and the $X_{(u,*)}$ tables can be used to constrain the computation of $Y_{(v,u)}$. This new constraint does not reduce the asymptotic complexity of the algorithm, but in practice, it tends to reduce roughly by half the total number of entries computed. The downside is that this new method requires to keep in memory all $X_{(u,v)}$ and $Y_{(v,u)}$ tables, whereas previously, the $X_{(u,*)}$ tables could be discarded as soon as W_u was computed. We will call this technique parent-bounding.

2.5 Filtering subtrings

The relatively tight bounds developed in Section 2.3 and 2.4 make the overall complexity of the algorithm $O(n \cdot min(l \cdot (3k)^{d/2}, k \cdot (4^k + l)))$, significantly better than the original complexity of $O(n \cdot k \cdot (4^k + l))$. Notice that the length l of the input sequences, which was totally dominated by the 4^k term in the original algorithm, is now a determining factor in the running time. That is because the number of substrings inserted in the W tables at the leaves (equation 1) directly affects the size of the boundaries explored. That suggests that reducing the number of substrings inserted in W at the leaves would proportionally reduce the whole running time. Indeed, if we could know in advance that a certain substring s has no chance to be part of a solution with score at most d, there would be no need to include it in W at the leaves.

There are several ways one can find out that a substring s can't produce interesting solutions. First, notice that in any solution $s_1, ..., s_n$ of parsimony score at most d, any pair of the chosen substrings must be at Hamming distance at most d. Thus, when considering whether to include substring s in W_u at a leaf u, one can find the best match of s in each of the $n-1$ other sequences. If any of these best matches have score worse than d, there is no need to consider s. This is a relatively weak bound, but for d small, it very effectively reduces the number of substrings considered. The problem of identifying all substrings of size k that have matches with score at most d in each of a given set of sequences is an interesting one by itself. Currently, the best algorithm we know about is based on suffix-trees and runs in time $O(n \cdot d \cdot l^2)$ [14], but it is very likely that better running times could be obtained.

Stronger bounds could also be used for filtering substrings. For example, if one builds an n-partite graph $G = (V, E)$, where V is the set of substrings of size k of the input sequences and $E = \{(s,t) : s$ and t come from different sequences and $d(s,t) \leq d\}$, then the only subtrings that can participate in an optimal solution are those that belong to an n-clique (see Pevzner and Sze [25] for more on this idea). Obviously, this kind of filtering would be very computationally intensive and it would probably take more time than it would save. Still, fast heuristics using these kinds of ideas seem possible. It is unclear how much computational effort should be invested in filtering. In our implementation, only the pairwise constraint was used.

2.6 Insertions and deletions

Until now, we have assumed that the only mutations allowed in the motifs sought were substitutions. However, the algorithms, bounds and filtering presented so far are independent of the type of mutations involved. One only needs to be able to define the neighborhood of a string, to perform the boundary expansion. In general, if each string has N neighbors, the algorithm will run in $O(n \cdot min(l \cdot N^{d/2}, N \cdot (4^k + l)))$. For example, if we allow for insertions and deletions, we get a complexity of $O(n \cdot min(l \cdot (8k+4)^{d/2}, 8k \cdot (4^k + l)))$. However, it doesn't make much sense to look for motifs of size *exactly* k if indels are allowed. Instead, we need to look for motifs of size *at least* k (and at most $k + d$).

2.7 Performance comparison

The algorithm of Sections 2.2 to 2.6 were implemented in C++ in a program called FootPrinter, and the code is available from the author. Table 1 presents the running time and space, as well as the total number of table entries computed, when the methods described so far are applied. The data set used is the $c-myc$ proto-oncogene 5' sequences, from 7 different vertebrates, whose size vary between 450 and 900 nucleotides. The motifs sought were of size 12, with a parsimony score of at most 3 mutations. This data set is quite typical of the data available these days.

Notice that when all bounding and filtering techniques are applied, we obtain a speed-up of a factor of about 1000 over the original algorithm. Moreover, the memory requirement, which is an important constraint, is reduced by a factor of 100. As expected from the asymptotic complexity of the algorithm, the main factor that determines the running time is d, the score of the motifs sought, and, to a lesser extent, k, the size of the motif sought. In practice, one can easily find motifs with $k = 8$ and $d = 8$, or $k = 12$ and $d = 5$, or $k = 20$ and $d = 2$, or $k = 30$ and $d = 1$. Allowing indels causes an increase in running time, in accordance with the asymptotic complexity. Still, the program runs quickly on a desktop machine.

2.8 Results, part I

Although the amount of sequences in the public domain is growing extremely fast, there are still relatively few genes for which the 5' or 3' non-coding region has been sequenced in several species. Many of those sequences can be found in the ACUTS database [10]. We ran our algorithm on all non-coding sequences known in at least 5 species in ACUTS. There is often little known about the regulation of those genes, so we increased our data set with 4 genes for which some binding sites are known and for which the untranslated regions were sequenced in at least 4 species. Table 2 reports highly conserved regions found by our algorithm in those sequences, with $k = 10$. Motifs longer than 10 were obtained

Technique	No indel Time	No indel Space	No indel #Entries	With indels Time	With indels Space	With indels #Entries
No bounds			$160*10^6$			$1*10^9$
d-bounds only	>500s	>1000M	$>20*10^6$	>10000s	>5000M	$>150*10^6$
Sibling bounds only	32s	40M	810130	727s	340M	6999597
Sibling+parent bounds	24s	39M	414207	573s	315M	3642586
All bounds+Filtering	4s(+10s filtering)	8M	60962	80s(+90s)	51M	559305

Table 1: Performance obtained when the various bounding and filtering techniques described in Section 2.3, 2.4, 2.5 are used. Data set: *c-myc* proto-oncogene gene upstream sequences, k=12, d=3, n=10, l varies between 450 and 900 nucleotides. The solution consists of 3 distinct conserved substrings. The program ran on a Pentium III 550 MHz, with a 512M RAM.

by concatenation of solutions that overlapped in all given sequences.

We lack a statistical basis to establish what is an "unusually well conserved region". Clearly, it depends, among other things, on the total divergence (e.g. in million of year (Myrs)) of the given sequences. In most cases, there appears to be a threshold d_0 for which there are a small number of solutions with score at most d_0, but a huge number of regions with score at most $d_0 + 1$. We report all regions with score at most d_0. This empirically determined d_0 is nicely correlated with the total divergence of the species considered, as expected (data not shown).

A large number of the regions identified by our algorithm are known regulatory elements. Indeed, among the genes for which binding sites are known, more than 80% of our predictions are correct. There is usually little known about regulatory elements in the 3' untranslated regions, but, as Duret *et al.* [10] report, these regions contain many highly conserved regions. Interestingly, in two cases, we identified very long conserved regions in the downstream region of the given gene (data not shown). These turned out to be unannotated genes. In that sense, the algorithm could be used as an aid in gene finding.

The rightmost column of Table 2 indicates whether other motif finding techniques could have identified the motif. We tested three of them: CLUSTALW [17](global multiple alignment, followed by visual inspection of the alignment), MEME [5], and Consensus [15], these last two being motif finders based on information content, making no use of the phylogeny underlying the data. The first thing to notice is that CLUSTALW performs very poorly for highly diverged sequences, as expected, for the reasons mentioned in Section 1. The two other motif finders do much better and correctly identify several regions. This was to be expected, as most of these regions are extremely well conserved and thus fairly easy to find. Nonetheless, our algorithm was able to identify known REs that all other methods missed. Moreover, almost all motifs obtaining significant scores from MEME or Consensus were also found be our method. This validates the assumption underlying our approach, that mutations in REs do follow the underlying phylogeny.

3. FINDING WEAKER SIGNALS

3.1 Branch length and ordering constraints

As mentionned in Section 2.8, there appears to be a threshold d_0 where very few solutions have score at most d_0, but very many have score at most $d_0 + 1$, and we can use this threshold to approximate what "unusually well conserved" means. But there still may be interesting regions that have a parsimony score above d_0. How can we extract them from the heap of other uninteresting solutions? A first type of spurious region occurs when an extremely well conserved motif produces a large number of variations. For example, suppose $(s_1, ..., s_n)$ is a motif with zero mutations, but that we allow up to d mutations. Then, any substring t with $d(s_1, t) \leq d$, and there may be several of them, will produce a motif $(t, s_2, ..., s_n)$ with score at most d. But the $(t, s_2, ..., s_n)$ motif scores well only because $(s_1, ..., s_n)$ is so well conserved, and thus we would like to discard t. This turns out to be easy to do by restricting the number of mutations allowed on each branch of the tree, which requires only slight changes in the formulation of the algorithm of Section 2.2.

Another way to extract interesting solutions from uninteresting ones is to use our prior knowledge about what solutions should look like. For example, unless some large scale genome rearrangement occurred, all conserved regions should occur in the same order in all species. We can use this information to extract from the heap of solutions (whose order vary from one species to another) a subset of motifs that occur in the same order in all given sequences. Although this is itself an NP-hard problem (it is in essence a longest common subsequence problem), a simple visual inspection usually suffices to reject motifs whose order is inconsistent.

Table 3 shows motifs found in *rbcL* and *c-myc* proto-oncogene upstream sequences, when at most two mutations per branch were allowed, and when the ordering constraints are applied. Again, many of these motifs correspond to known REs. Moreover, most of these motifs could not have been found by the algorithm of Section 2.2, because their parsimony score was above the d_0 threshold.

3.2 Allowing for missing regulatory elements

Until now, we have assumed that the conserved regions we were looking for were present in all sequences $S_1, ..., S_n$. That is, we have assumed that they had been subject to selective pressure over all branches of the tree T. That means that we were unable to find REs that were absent from some of the given species. There are many examples of data sets where some REs are common to only part of the tree con-

Region	Species	Motif	#Mut	Ref.	Other methods
β-actin 5'	1,5,12,15,16	ccaatcagcg (-934)	1	<1a>	C
	(1200Myrs)	ttgccttttatggc (-915)	0	<1b>	M, C
		ttgccttttatggtaat (-160)	0	<2a>	M, C
		cttcctttgtc (-131)	0		C
β-actin 3'	1,4,6,7,12,14,15,16,17	ttggcatggctt (70)	1		M,C,W
	(1900Myrs)	ttgactcaggat (127)	1		M,C,W
		aaaaactggaac (141)	1		M,C,W
		agtcattccaaat (242)	2		M,C,W
		tgtaaattatgt (374)	2		M,C
		cctgtacactgac (545)	0	<3>	M,C,W
γ-actin 3'	1,5,6,9,10,13	tcatgctagcctc (78)	0		M,C,W
	(1100Myrs)	aaactggaataag (94)	0		M,C,W
		tgtgcagggtat (353)	1		M,C
c-fos oncogene 5'	1,6,12,18	cacaggatgtccatattag...			
	(1200Myrs)	...gacatctgcgtcag (-478)	3	<4a>	C
		ccgcggcccc (-568)	3		
c-fos oncogene,	1,6,12,18,20	agggatatttata (419)	1		M,C,W
first intron	(1400Myrs)				
Hedgehog morphogen 5'	5,10,11,12,19	ccgatgtgttc (-265)	1		M,C
	(1500Myrs)	cagcccctgtct (-225)	3		M,C
dihydrofolate reductase 5'	1,6,7,23	gggcggggcc (-111)	3	<5a>	
	(1300Myrs)	ttcgcgccaaact (-62)	2	<5b>	M
insulin 5'	1,3,5,8	agacccagca (-395)	0		M,C,W
	(400Myrs)	cctcagcccc (-359)	1	<6a>	M,C
		gccatctgcc (-347)	1	<6a>	M,C,W
		ccctaatgggcca (-322)	1	<6b>	M,C,W
metallothionein 5'	5,7,9,21	agctctgcactc (-225)	3	<7a>	M,C
	(1000Myrs)	gtgcgctcggctctgc (-203)	2	<7b>	M,C
		ttgcgcccgg (-125)	3	<7c>	M,C

Table 2: Highly conserved regions found by our algorithm in non-coding regions. The sequence and position of the motifs reported correspond to the first species listed. Motif positions are from the start codon for 5' sequences, and from stop codon for 3' sequences. All substrings of size 10 of the reported motifs contain at most the reported number of mutations.
Other methods (see Section 2.8): M: Meme, C: Consensus, W: ClustalW.
References: <1a> Bound by unknown factor [12], <1b> Bound by *SRF* [12], <2a> subcellular localization [20], <3> Bound by some factor [19], <4a> Binds *SRF*, *YY1* and *SRE-ZBP* [24], <5a> Binds *Sp1* in mouse [23], <5b> Binds *HIP1* in mouse [23], <6a> TRANSFAC database [32], <6b> [8], <7a> *GC-box* [2], <7b> *MREd* [2], <7c> *MREa* [2].
Species: (1) human, (2) gibbon, (3) owl monkey, (4) rabbit, (5) rat, (6) mouse, (7) hamster, (8) Guinea pig, (9) cow, (10) frog, (11) Japanese newt, (12) chicken, (13) goose, (14) gilthead sea bream, (15) grass carp, (16) common carp, (17) medaka fish, (18) electric ray, (19) *Danio rerio*, (20) fugu, (21) trout, (22) goldfish, (23) fruit fly.

Gene	Species	Motif	#Mut.	Ref.	Other methods
rbcL 5'	50 species of magnoliophyta[(1)] (1200Myrs)	taggatttacatatacaaca (-326)	4	<1a>	M
		aatcaaata (-170)	4		
		gttgataa (-131)	3		
		aattcttaattcat (-32)	2	<1b>	M,W
c-myc proto-oncogene 5'	1,2,5,10,12,16,22[(2)] (1500Myrs)	gggcatttaaatttc (-1176)	3		C
		cactggaacttaca (-1804)	3		C
		gcttcgcctc (-1906)	4	<2a>	
		ctcgctgtagtaattcca (-2029)	4	<2b>	C
		gcttggcgggaaaa (-2101)	3	<2c>	

Table 3: Highly conserved regions found by our algorithm, with ordering constraint and at most 2 mutations per branch. The sequence and position of the motifs reported correspond to the first species listed. All substrings of size 8 of the reported motifs contain at most the reported number of mutations.
Species: [(1)] 5 lilliopsida, 6 magnoliales, 11 asteridae, 1 buxus, 27 rosidae. [(2)] Refer to Table 2.
References: <1a> *atpB* promoter (*atpB* is located 5' of *rbcL*) [22], <1b> highly conserved leader region [22], <2a> *HindIII* site [4], <2b> in contact with zinc finger of *CTCF* [11], <2c> binds *E2F* [16].

sidered, and it would be useful to be able to find those as well, without knowing a priori which species contain those elements. In this section, we describe how an algorithm similar to that of Section 2.2 can handle this situation.

For this problem to make sense, we need a way to compare two solutions containing substrings from different subsets of species. For example, a motif containing m mutations in a group of ten highly diverged vertebrates is more interesting than one with the same number of mutations in a group of ten primates. What needs to be considered is the total amount of evolution (measure in million of years, for example) the motif has resisted. Thus, in what follows, we will assume that we are not only given the phylogenetic tree T that relates the given species, but also the lengths of all its branches. We will be looking for motifs that have a small parsimony score, but that span a large part of the tree. We assume that a RE is either present at the root of the tree, or that it is acquired on exactly one branch of the tree (that is, a RE is not acquired independently on two different phyla). The problem we want to solve here is the following:

SUBSTRING PARSIMONY PROBLEM WITH LOSSES
Given: a set of orthologous sequences $S_1, ..., S_n$ from n different species, the phylogenetic tree T relating these species and the length $\lambda(e)$ of each branch e of T, the size k of the motifs to look for, an integer d, and a set of thresholds $\{\delta_0, \delta_1, ..., \delta_d\}$.
Problem: find all sets of k-substrings $\{s_{i_1}, ..., s_{i_\zeta}\}$, $1 < \zeta \leq n$, where s_{i_j} is a substring of S_{i_j}, such that the parsimony score P of $\{s_{i_1}, ..., s_{i_\zeta}\}$ on the tree induced by the leaves $i_1, ..., i_\zeta$ is at most d, and such that the tree induced by the leaves $i_1, ..., i_\zeta$ is of size at least δ_P.

For example, one might be interested in motifs with a parsimony score of 2 that span at least 500 million years of evolution (i.e. $\delta_2 = 500$ Myrs), and in regions with a parsimony score of 3 that span at least 1000 million years of evolution (i.e. $\delta_3 = 1000$ Myrs).

The algorithm that solves this generalized problem is very similar in spirit to that of Section 2.2. At each node u of T, we now have a set of tables $W_{u,a}$, for $a = 0...d$, where $W_{u,a}[s]$ is defined as the maximal size of a leaf-induced subgraph of the subtree rooted at u, containing u, on which there exists a solution with parsimony score a, if u was to be labeled with s. For each edge (u, v) of the tree, we define $X_{(u,v),a}$ similarly and get

$$X_{(u,v),a}[s] = \begin{cases} \max\{z \in Z_{v,a,s}\} + \lambda((u,v)) & \text{if } Z_{v,a,s} \neq \phi \\ -\infty & \text{otherwise} \end{cases}$$

where $Z_{v,a,s} = \{W_{v,b}[t] : b + d(s,t) = a, t \in \Sigma^k\}$, and

$$W_{u,a}[s] = \max_{\substack{1 \leq q \leq |C(u)|, \\ \{c_1, c_2, ..., c_q\} \subseteq C(u), \\ \{b_1, b_2, ..., b_q\} \text{ partition of } a}} \sum_{i=1}^{q} X_{(u,c_i),b_i}[s]$$

The X tables can be computed by an algorithm very similar to that of Section 2.2. The bounds developed in Sections 2.3 and 2.4 also apply with little modification, and filtering the input substrings could also be used, although the modifications required would be non-trivial (in particular, a substring of sequence S_1 that has no good match with sequence S_2 can't be rejected, because S_2 might not participate in the solution). The set of substrings satisfying the given conditions are recovered by tracing back the recursion from any node u, any entry s, and any score a such that $W_{u,a}[s] \geq \delta_a$.

We illustrate the usefulness of this extension on two plant genes, *rbcS* and *rbcL*, in Table 4. These two genes contain several REs that are known to be present in most, but not all, of the species considered here. When comparing the results on *rbcL* with those obtained when all sequences had to contribute a substring (Table 3), one notices that many more predictions correspond to known REs. Indeed 5 of the 6 REs known in tobacco are found in Table 4. In *rbcS*, all three REs known to be present in at least 5 of the 10 input sequences were found, and no other regions appeared significantly well conserved.

Gene	Species	Motif	#Mut	Induced size	Ref.	Other methods
rbcS	10 embryophyta[(1)]	tatatatag (-105)	1	760 Myrs	<1a>	M,C
	(760Myrs)	agataagat (-163)	3	760 Myrs	<1b>	M
		cacgtggca (-303)	3	570 Myrs	<1c>	M,C
rbcL	50 Magnoliophyta[(2)]	taggatttacatatacaaca...				
	(1200Myrs)	...tata (-326)	0	1180 Myrs	<2a>	M
		ttgggttgcgc(-221)	0	1120 Myrs	<2b>	M,W
		atatgaaagagtatacaata...				
		...atgatgtatttgg (-205)	0	1180 Myrs	<2c>	M,W
		attagttgataat (-135)	1	1000 Myrs	<2d>	
		tgtcgagtagaccttgttgt (-59)	0	1160 Myrs	<2e>	M,W,C
		tgtgagaattcttaattcat...				
		...gagttgtagggaggga (-39)	0	1180 Myrs	<2f>	M,W,C

Table 4: Highly conserved regions found in a subset of the input sequences. All substrings of size 9 of the reported motifs contain at most the reported number of mutations on the induced subtree. Induced size: Size of the tree that contains the motif.
Species : [(1)] spinach, tomato, ice plant, tobacco, rape, cotton, pea, wheat, lemna gibba, corn. [(2)] See Table 3. References From [3]: <1a> TATA-box, <1b> I-box , <1c> G-box, absent from rape, wheat and duckweed. From [22]: <2a> *atpB* promoter (*atpB* is located upstream of *rbcL*), absent from *Spermacoceas surgens*. <2b> *rbcL* promoter, similar to -35 promoter, mutated in rice, *cuscuta*, and cotton, absent from *corytophora*. <2c> *rbcL* promoter, similar to -10 promoter, absent from *cuscuta*. <2d> Unknown function, absent from all 3 iris and from *cuscuta*. <2e> 5' part of leader region, absent from corn and *cuscuta*. <2f> 3' part of leader region, absent from *cuscuta*.

4. CONCLUSION AND FUTURE WORK

In this paper, we present an exact algorithm for phylogenetic footprinting. The algorithm is based on our original work [7], but with significant algorithmic improvements that make it amenable to the largest data sets. We show how to use ordering and branch length constraints to extract interesting regions from spurious ones. We then describe how to handle data sets where REs are only present in a subset of the input sequences. All algorithms presented are guaranteed to yield optimal solutions to the variety of problems addressed. Algorithms are implemented and run quickly an a desktop computer. Using them, we have been able to identify a large number of experimentally determined binding sites, as well as a set of highly conserved regions with no function known yet.

Future work should be directed in three main directions. First, it seems possible to improve the bound presented in Section 2.3 and 2.4, as well as the filtering accuracy and time complexity. Second, it will be important to develop a solid statistical basis to be able to assess how "unusually" well conserved a given region really is. That appears to be a difficult task as these statistics would have to take into account a large number of factors such as the tree topology, the length of its branches, the length and background distribution of the input sequences, etc. It would be even harder in the case where REs are allowed to be missing from some sequences. Still, it is likely that the simple measure of total divergence of the species considered could provide a good estimate of what amount of conservation is really unusual. Third, it appears that quite often, REs work by pair (or even triplet), at a relatively fixed distance from each other. This new kind of prior knowledge could easily be included in our algorithm and may allow us to find pairs of REs that were too weakly conserved to be detected by themselves.

From an application point of view, the next years should provide us with a bulk of new data to look at. Interesting tasks include not only RE prediction, but also gene and splice site detection, as well as amino-acid motif finding.

5. ACKNOWLEDGEMENTS

I would like to thank Martin Tompa for the numerous hours he spent helping me with the completion of this project.

6. REFERENCES

[1] T. Akutsu. Hardness results on gapless local multiple sequence alignment. Technical Report 98-MPS-24-2, Information Processing Society of Japan, 1998.

[2] R. D. Andersen, S. J. Taplitz, S. Wong, G. Bristol, B. Larkin, and H. R. Herschman. Metal-dependent binding of a factor in vivo to the metal-responsive elements of the metallothionein 1 gene promoter. *Mol. Cell. Biol.*, 7:3574–3581, 1987.

[3] G. Arguello-Astorga and L. Herrera-Estrella. Evolution of light-regulated plant promoters. *Annu. Rev. Plant Physiol. Plant Mol. Biol.*, 49:525–555, 1998.

[4] H. Ariga, Y. Imamura, and S. M. M. Iguchi-Ariga. DNA replication origin and transcriptional enhancer in c-myc gene share the c-myc protein binding sequences. *EMBO J.*, 8:4273–4279, 1989.

[5] T. L. Bailey and C. Elkan. Unsupervised learning of multiple motifs in biopolymers using expectation maximization. *Machine Learning*, 21(1-2):51–80, Oct. 1995.

[6] M. Blanchette. An exact algorithm to identify motifs in orthologous sequences from multiple species. Technical Report , Qualification project, University of Washington, 2000.

[7] M. Blanchette, B. Schwikowski, and M. Tompa. An exact algorithm to identify motifs in orthologous sequences from multiple species. In *Proceedings of the Eighth International Conference on Intelligent Systems for Molecular Biology*, pages 37–45, La Jolla, USA, Aug. 2000. AAAI Press.

[8] D. S. W. Boam, A. R. Clark, and K. Docherty. Positive and negative regulation of the human insulin gene by multiple trans-acting factors. *J. Biol. Chem.*, 265:8285–8296, 1990.

[9] L. Duret and P. Bucher. Searching for regulatory elements in human noncoding sequences. *Curr. Op. in Struct. Biol.*, 7:399–405, 1997.

[10] L. Duret, F. Dorkeld, and C. Gauthier. Strong conservation of non-coding sequences during vertebrates evolution: potential involvement in post-transcriptional regulation of gene expression. *Nucleic Acids Research*, 21, 10:2315–2322, 1993.

[11] G. N. Filippova, S. Fagerlie, E. M. Klenova, C. Meyers, Y. Dehner, G. Goodwin, P. E. Neiman, S. J. Collins, and V. V. Lobanenkov. An exceptionally conserved transcriptional repressor, ctcf, employs different combinations of zinc fingers to bind diverged promoter sequences of avian and mammalian c-myc oncogenes. *Mol. Cell. Biol.*, 16:2802–2813, 1996.

[12] R. M. Frederickson, M. R. Micheau, A. Iwamoto, and N. G. Miyamoto. 5' flanking and first intron sequences of the human beta-actin gene required for efficient promoter activity. *Nucleic Acids Res.*, 17:253–270, 1989.

[13] D. Gumucio, D. Shelton, W. Bailey, J. Slightom, and M. Goodman. Phylogenetic footprinting reveals unexpected complexity in trans factor binding upstream from the ϵ-globin gene. *Proc. Natl. Acad. Sci. USA*, 90:6018–6022, 1993.

[14] D. Gusfield. *Algorithms on Strings, Trees, and Sequences*. Cambridge University Press, 1997.

[15] G. Z. Hertz and G. D. Stormo. Identifying DNA and protein patterns with statistically significant alignments of multiple sequences. *Bioinformatics*, 15(7/8):563–577, July/August 1999.

[16] S. W. Hiebert, M. Lipp, and N. J. R. E1a-dependent trans-activation of the human myc promoter is mediated by the e2f factor. *Proc. Natl. Acad. Sci. USA*, 86:3594–3598, 1989.

[17] D. G. Higgins, J. D. Thompson, and T. J. Gibson. Using CLUSTAL for multiple sequence alignments. In R. F. Doolittle, editor, *Computer Methods for Macromolecular Sequence Analysis*, volume 266 of *Methods in Enzymology*, pages 383–401. Academic Press, New York, 1996.

[18] J. Hughes, P. Estep, S. Tavazoie, and G. Church. Computational identification of cis-regulatory elements associated with groups of functionally related genes in Saccharomyces cerevisiae. *J. Mol. Biol.*, 296(5):1205–14, 2000.

[19] T. Kawamoto, K. Makino, H. Niwa, H. Sugiyama, S. Kimura, M. Amemura, A. Nakata, and T. Kakunaga. Identification of the human beta-actin enhancer and its binding factor. *Mol. Cell. Biol.*, 8:267–272, 1988.

[20] E. Kislauskis, X. Zhu, and R. Singer. Sequences responsible for intracellular localization of beta-actin messenger RNA also affect cell phenotype. *J. Cell. Biol.*, 127:441–451, 1994.

[21] J. Leung, E. McKenzie, A. Uglialoro, P. Florez-Villanueva, B. Sorkin, E. Yunis, D. Hartl, and A. Goldfeld. Identification of phylogenetic footprints in primate tumor necrosis factor-α promoters. *Proc. Natl. Acad. Sci. USA*, 97, 12:6614–6618, 2000.

[22] J. F. Manen, V. Savolainen, and P. Simon. The *atpB* and *rbcL* promoters in plastid DNAs of a wide dicot range. *Journal of Molecular Evolution*, 38(6):577–582, June 1994.

[23] A. L. Means and P. G. Farnham. Transcription initiation from the dihydrofolate reductase promoter is positioned by hip1 binding at the initiation site. *Mol. Cell. Biol.*, 10:653–661, 1990.

[24] S. Natesan and M. Gilman. Yy1 facilitates the association of serum response factor with the c-fos serum response element. *Mol. Cell. Biol.*, 15:5975–5982, 1995.

[25] P. Pevzner and S. Sze. Combinatorial approaches to finding subtle signals in DNA sequences. In *Proceedings of the Eighth International Conference on Intelligent Systems for Molecular Biology*, pages 269–278. AAAI Press, Aug. 2000.

[26] D. Sankoff and P. Rousseau. Locating the vertices of a Steiner tree in arbitrary metric space. *Mathematical Programming*, 9:240–246, 1975.

[27] S. Sinha and M. Tompa. A statistical method for finding transcription factor binding sites. In *Proceedings of the Eighth International Conference on Intelligent Systems for Molecular Biology*. AAAI Press, Aug. 2000.

[28] D. Tagle, B. Koop, M. Goodman, J. Slightom, D. Hess, and R. Jones. Embryonic ϵ and γ globin genes of a prosimian primate (Galago crassicaudatus) nucleotide and amino acid sequences, developmental regulation and phylogenetic footprints. *J. Mol. Biol.*, 203:439–455, 1988.

[29] S. Tavazoie, J. D. Hughes, M. J. Campbell, R. J. Cho, and G. M. Church. Systematic determination of genetic network architecture. *Nature Genetics*, 22:281–285, July 1999.

[30] S. Vuillaumier, I. Dixmeras, H. Messai., C. Lapoumeroulie, D. Lallemand, J. Gekas, F. Chebab, C. Perret, J. Elion, and E. Denamur. Cross-species characterization of the promoter region of the cystic fibrosis transmembrane conductance regulator gene reveals multiple levels of regulation. *Biochem J.*, 327:652–662, 1997.

[31] L. Wang and T. Jiang. On the complexity of multiple sequence alignment. *Journal of Computational Biology*, 1:337–348, 1994.

[32] E. Wingender, P. Dietze, H. Karas, and R. Knüppel. TRANSFAC: a database on transcription factors and their DNA binding sites. *Nucleic Acids Research*, 24(1):238–241, 1996. `http://transfac.gbf-braunschweig.de/TRANSFAC/`.