An Exact Algorithm to Identify Motifs in Orthologous Sequences from Multiple Species *

Mathieu Blanchette, Benno Schwikowski, and Martin Tompa
Department of Computer Science and Engineering
Box 352350
University of Washington
Seattle, WA 98195-2350 U.S.A.
206-543-9263
fax: 206-543-8331
{blanchem,benno,tompa}@cs.washington.edu

Abstract

The identification of sequence motifs is a fundamental method for suggesting good candidates for biologically functional regions such as promoters, splice sites, binding sites, etc. We investigate the following approach to identifying motifs: given a collection of orthologous sequences from multiple species related by a known phylogenetic tree, search for motifs that are well conserved (according to a parsimony measure) in the species. We present an exact algorithm for solving this problem. We then discuss experimental results on finding promoters of the *rbcS* gene for a family of 10 plants, on finding promoters of the *adh* gene for 12 *Drosophila* species, and on finding promoters of several chloroplast encoded genes.

Keywords: sequence analysis, motif, promoter, orthologous, phylogeny, algorithm.

1. Using motifs to find functional regions

One of the fundamental techniques in biological sequence analysis is the identification of sequence motifs as a means of suggesting good candidates for biologically functional regions such as promoters, splice sites, binding sites, etc. For short repeated sites, the common way this is done is to assemble a collection of sequences from a single genome believed to contain the site, and search in these sequences for a pattern that occurs in a statistically significant overabundance. To find promoters, for example, one would identify genes believed to be coregulated, and then look for patterns in their regulatory regions. (See, for example, Brāzma *et al.* (1998), Chu *et al.* (1998), Fraenkel *et al.* (1995), Galas *et al.* (1985), Hertz and Stormo (1999), Roth *et al.* (1998), Spellman *et al.* (1998), Staden (1989), Tavazoie *et al.* (1999), Sinha and Tompa (2000), and van Helden *et al.* (1998).) From a computational point of view, the methods used to find the patterns fall into four main categories. Most exact methods are enumerative (van Helden *et al.* (1998), Sinha and Tompa (2000)). Heuristic methods include Gibbs sampling (Lawrence *et al.* (1993)), expectation maximization (Bailey and Elkan (1995), Lawrence and Reilly (1990)), and greedy search (Hertz and Stormo (1999)).

In his 1999 ISMB invited address, Kerlavage (1999) suggested an orthogonal approach to finding motifs: assemble a collection of orthologous regions from multiple related species, and search for motifs that are well conserved in most or all of the species. This approach has the advantage of being able to identify functional regions that occur only once per genome, for instance a promoter that is peculiar to one gene.

Kerlavage's idea is not new, though, and has been used succesfully to find promoters for pancreatic β cell type-specific genes (Magnuson & Shelton 1989) and for glucokinase genes (Magnuson & Thomas 1993). However, to our knowledge, no systematic method has been developed for this kind of task. With the large number of genome projects in progress and planned, Kerlavage's suggestion is compelling and timely, and an algorithmic solution is necessary. In this paper, we investigate the computational aspects of this approach and apply the technique to various set of orthologous sequences.

From the computational viewpoint, the problem is as follows. We are given orthologous sequences $s_1, s_2, \ldots, s_n$, one from each of n related species. We are looking for sequences $t_1, t_2, \ldots, t_n$, where t_i is a substring of s_i for all $1 \leq i \leq n$, such that $t_1, t_2, \ldots, t_n$ have an unusually high measure of mutual sequence similarity.

What constitutes "unusually high mutual se-

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

quence similarity"? One difficulty is a familiar one: if all n sequences are weighted equally, then a cluster of closely related species will carry undue weight in determining the motif. For this reason, traditional methods for finding overrepresented patterns may not be appropriate. To cope with this, we assume we are also given a good phylogenetic tree $T = (V, E)$ with the n species at its leaves, and measure mutual sequence similarity by parsimony. Many multiple alignment algorithms such as CLUSTALW (Higgins, Thompson, & Gibson 1996) explictly or implicitly use a phylogenetic tree to avoid the problems of overrepresentation.

In more detail, suppose the leaves of T are numbered $1, 2, \ldots, n$, and the internal nodes $n+1, n+2, \ldots, |V|$. The *substring parsimony problem* is to find substrings $t_1, t_2, \ldots, t_n$ of $s_1, s_2, \ldots, s_n$, respectively, and strings $t_{n+1}, t_{n+2}, \ldots, t_{|V|}$ that minimize

$$P(T) = \sum_{\{u,v\} \in E} d(t_u, t_v), \qquad (1)$$

where $d(t, t')$ is the distance between strings t and t'. Throughout most of the paper we will assume that each substring t_i has the same user-specified length k, and that $d(t, t')$ is the Hamming distance. This is not an unreasonable metric for promoters, whose instances usually differ from each other by substitutions rather than insertions and deletions. In Section 2.3, we discuss the extension to other metrics such as edit distance.

Global multiple alignment such as computed by CLUSTALW (Higgins, Thompson, & Gibson 1996) cannot generally be used to exhibit these short conserved regions, because the signal sought is overwhelmed by the noise in the input sequences, and because the position of the signal may vary widely among the species. A typical scenario arises in identifying promoters, in which the input might be the upstream sequences of n orthologous genes. These sequences might each be 1000bp long, and the promoter(s) only 10bp long.

Even for ungapped motifs as described above, the substring parsimony problem is *NP*-hard, so we cannot expect an efficient algorithm that finds the optimal solution. The proof of *NP*-hardness is nearly identical to one of Akutsu (1998). His problem was not based on phylogeny, but was instead to find n ungapped, equal length substrings that maximize the relative entropy (information content) score.

In Section 2 we present a novel algorithm to solve the substring parsimony problem *exactly*. Given sequences $s_1, s_2, \ldots, s_n$, tree T, and an integer k, the algorithm identifies a length k substring of each input sequence and a length k sequence at each internal node of T that achieve the global minimum of the parsimony measure $P(T)$ given in Equation (1). The running time of this algorithm is $O(nk(l + 4^k))$, where $l = \frac{1}{n}\sum_{i=1}^{n} |s_i|$. We also describe a simple pruning idea that decreases this time in most cases.

Although the running time is exponential in k, the algorithm is practical for $k \leq 13$, which is sufficient for finding short conserved motifs such as promoters, splice sites, and binding sites. The fact that the running time is linear in the input size nl (when k is constant) is important in practice, because the regulatory regions and number of species can be quite large in some applications.

In Section 3 we discuss our implementation, and describe experimental results on finding promoters of the *rbcS* gene for a family of 10 plants, on finding promoters of the *adh* gene for 12 *Drosophila* species, and on finding promoters of several chloroplast encoded genes. To give an idea of the actual running time, the experiment to find the most conserved 9-mer among 98 chloroplast sequences each of length 600bp took approximately two minutes. We also discuss how we use simulated data to assess the significance of the motifs identified by our implementation.

Our algorithm does not require that the n input sequences be the same length. Because of this, another interesting application of the algorithm is to determine if a known functional site of a few species S has orthologs in related species. To do so, instead of labeling the leaves corresponding to S with their entire input sequences s_i, label them with just their known functional site. Running the algorithm of Section 2 will then locate parsimonious matches in the remaining input sequences.

2. Algorithm

To solve the substring parsimony problem optimally, we adapt the method of Sankoff and Rousseau (1975). Their method was formulated for general metric spaces, and assumed that the leaves correspond to fixed elements of the metric space. In a simple form, it was first described by Fitch (1971). In this section we show how the method can be adapted to the case of k-substrings (i.e., substrings of length k), and extended to include the choice of the substrings at the leaves.

The most straightforward adaptation runs in time $O(nk(l + 4^{2k}))$, which is prohibitive for the values of k we have in mind. In Section 2.1, we show how to improve this running time dramatically.

As in the original method by Sankoff and Rousseau, we assume that the tree is rooted at an arbitrary internal node r. Let $\Sigma = \{\mathtt{A}, \mathtt{C}, \mathtt{G}, \mathtt{T}\}$. For

each node $v \in V$ and each sequence $t \in \Sigma^k$, our algorithm computes and stores the *subtree score* $d_v^*(t)$. $d_v^*(t)$ is the minimal possible parsimony score in the subtree under v, under the restriction that v is labeled with t. The core idea of the method is that subtree scores can be computed recursively, proceeding from the leaves of the tree to its root. For each leaf v one defines $d_v^*(t) := 0$ if t is a k-substring of s_v, and $d_v^*(t) := \infty$ otherwise. Then, for an internal node v with children $C(v)$ and any sequence $t \in \Sigma^k$,

$$d_v^*(t) = \sum_{w \in C(v)} \min_{t' \in \Sigma^k} \left(d_w^*(t') + d(t', t) \right).$$

The best possible score for the whole tree corresponds to the best subtree score at the root node r, namely $\min_{t \in \Sigma^k} d_r^*(t)$.

After an optimal sequence t_r has been found in this way, optimal choices for sequences t_w for all other nodes w can again be found recursively by moving from the root node towards the leaves. In particular, if a sequence t_v has been determined for the parent node v of a node w, then t' is an optimal choice for t_w if and only if $d_w^*(t') + d(t', t_v)$ is minimal.

Figure 1 gives the details of this algorithm. Notice that lines 6 and 8 contain choices that can be resolved in several ways to yield all possible optimal solutions.

Concerning the running time of the algorithm, observe that the Hamming distance $d(t, t')$ between two sequences t and t' in Σ^k can be computed in time $O(k)$, and T has at most $2n-2$ edges. Therefore, lines 3–5 can be executed in time $O(nk \cdot 4^{2k})$, and the running time of the whole algorithm is $O(nk(l + 4^{2k}))$.

2.1. Running time reduction from $O(k \cdot 4^{2k})$ to $O(k \cdot 4^k)$ per edge

A significant reduction in running time can be achieved by replacing the computation of $d_v^*(t)$ in lines 3–5 of Figure 1 with the following. For each child w of the current node v, a table T_w with an entry $T_w(t)$ for each $t \in \Sigma^k$ is generated. Let $c_w(t)$ be $\min_{t' \in \Sigma^k} \left(d_w^*(t') + d(t', t) \right)$. $c_w(t)$ is the contribution of w to $d_v^*(t)$. $T_w(t)$ will eventually contain $c_w(t)$. The value of $d_v^*(t)$ will then be obtained as $\sum_{w \in C(v)} T_w(t)$.

Each table T_w is filled in a series of rounds. Before the first round, each sequence $t' \in \Sigma^k$ is added to an initially empty set S_i with $i = d_w^*(t')$. The first round begins with an empty table T_w and the nonempty set S_i with the minimal index i. After that, the value of i is increased by one in each round. An entry for t is only stored in T_w if no previous entry for t was stored. In round i, such new entries are generated in two ways. First, any entry $T_w(t') = i-1$ generates new entries $T_w(t) = i$ for all t with $d(t, t') = 1$. Second, for each $t \in S_i$ an entry $T_w(t) = i$ is generated. The process ends when T_w is filled.

To see why, in round i, the entry i is stored for exactly those t with $c_w(t) = i$, assume that this holds for all rounds up to some fixed round i_0. Consider any t with $c_w(t) = i_0$. At the beginning of round i_0, $T_w(t)$ is empty. According to the definition of $c_w(t)$ there exists a sequence t' with $d_w^*(t') + d(t', t) = i_0$. If $t' = t$, then t is a member of S_{i_0}, and an entry i_0 for t is generated. Otherwise, there exists a sequence t'' with $c_w(t'') = i_0 - 1$ that differs from t by a single substitution. According to our assumption, T_w contains the entry $i_0 - 1$ for t'' from the previous round, and therefore the entry i_0 for t is generated and stored in T_w.

The time needed by this process is dominated by the generation of $O(k)$ neighbors for each of the 4^k entries in T_w. Therefore, generating a single table T_w can be accomplished in time $O(k \cdot 4^k)$. Since T_w is generated once for every node w of T, except the root node, the overall complexity of the modified algorithm is $O(nk(l + 4^k))$. This presents a substantial reduction in running time, compared to the straightforward algorithm of Figure 1.

2.2. Further running time reduction by a lower bound

In practice, further improvement in the running time can be achieved using a lower bound. Observe that, at any time during the execution of the algorithm, subtree scores have been computed for a set of disjoint subtrees of T. Call the set of root nodes of these subtrees R, and assume that a final solution with a score of b is obtained by choosing a sequence t_w for node $w \in R$. Then the score b of the final solution satisfies

$$b \geq d_w^*(t_w) + \sum_{u \in R - w} \min_{t \in \Sigma^k} d_u^*(t),$$

since any complete solution contains at least the subtree score of $d_w^*(t_w)$ at w, and a score contribution of at least $\min_{t \in \Sigma^k} d_u^*(t)$ from the subtree rooted at each $u \in R - w$.

In the subsequent step of the algorithm, when the subtree scores for a new internal node v are computed, it follows that, if the sequence t_w is chosen for a child w of v,

$$c_w(t_w) \leq d_w^*(t_w) \leq b - \sum_{u \in R - w} \min_{t \in \Sigma^k} d_u^*(t).$$

Thus, sequences t_w with $c_w(t_w) > b(w) := b - \sum_{u \in R-w} \min_{t \in \Sigma^k} d_u^*(t)$ do not participate in the

Algorithm Substring parsimony (input: $s_1, \ldots, s_n, k \in \mathbb{N}, T = (V, E, r)$);

1 *for* each leaf v of T *do*
2 let $d_v^*(t) := 0$ for each k-substring t of s_v, $d_v^*(t) := \infty$ otherwise;
3 *for* each internal node v of T, from the leaves toward the root r, *do*
4 *for* each sequence $t \in \Sigma^k$ *do*
5 compute $d_v^*(t) = \sum_{w \in C(v)} \min_{t' \in \Sigma^k} \left(d_w^*(t') + d(t', t)\right)$;
6 select a $t_r \in \Sigma^k$ such that $d_r^*(t_r)$ is minimal;
7 *for* each child w of a node v, from the root r toward the leaves *do*
8 choose t_w such that $d_w^*(t_w) + d(t_w, t_v)$ is minimal;

Figure 1: Straightforward algorithm for the substring parsimony problem

computation of a solution with a score of b or less, and can therefore be ignored if only such solutions are of interest. In the context of the algorithm from Section 2.1, this means that the computation of the table T_w can be stopped after round $i = b(w)$, and an empty entry in a table $T_w(t)$ for one or more $w \in C(v)$ leads to an empty entry in the table that holds $d_v^*(t)$.

For our practical experiments, described in Section 3, it has proven effective to iteratively run this improved algorithm with values of $b = 0, 1, \ldots$, until a solution is found. When no solution with a score of b or less exists, the number of entries in the tables T_w is typically very small. The algorithm then quickly gets to the point where an empty table T_v is generated for an internal node v, at which point we know that no solution with score at most b exists. The tables T_w are then cleared, b incremented, and the process restarted.

2.3. Extension to general finite metric spaces

Our extension to the algorithm of Sankoff and Rousseau and the running time reductions work in any finite metric space and can thus be used for other appropriate measures d of evolutionary distance.

The most common case is that one considers at most Δ possible evolutionary operations from each point in a space of M possible points. (In our case, $\Delta = 3k$ and $M = 4^k$.) Typically d then measures the minimum (weighted) number of evolutionary operations between two given points. Using a priority queue, the $O(n)$ tables T_w can then be generated in time $O(n \cdot \Delta \cdot M \log M)$, and the total running time is $O\big(n \cdot \Delta \cdot (l + M \log M)\big)$.

Interesting cases that can be tackled using this extension include the space of bounded-length sequences, with a set of evolutionary operations that includes point mutations, insertions and deletions, and reversals. Notice that the running time reduction technique of Section 2.1 completely circumvents the frequent evaluation of the distance function d in line 5 of the main loop in the straightforward algorithm. This is important because in many cases the computation of d itself is a hard problem (e.g., the unsigned reversal distance (Caprara 1997)).

3. Experimental Results

Currently, there are relatively few genes for which the upstream sequence is known for a number of species, but it will certainly increase rapidly in the near future. The algorithm described in Section 2 was used to find the most conserved regions in upstream sequences of selected sets of orthologous genes (Genbank 2000). The algorithm succeeded in identifying many of the known promoters in the given sequences. It also identified conserved regions that are not yet known to be functional. We first report the results obtained on three different data sets, and then describe in Section 3.5 how we assess the significance of the motifs found. This measure of significance is used to determine which promoters are reported in our results.

3.1. Plant's *rbcS*

The ribulose-1,5-bisphosphate carboxylase small subunit gene (*rbcS*) is a gene present in most plants. It has been used as a model for studying the expression of light-regulated genes. A large number of promoters are known to regulate *rbcS* (Arguello-Astorga & Herrera-Estrella 1998). These promoters were identified experimentally, and are usually located within 500bp upstream of the start codon. Their sizes vary from 6 to 12 nucleotides. Three of these promoters are known to be present in most dicotyledons: the TATA-box, the I-box, and the G-box. The TATA-box and the I-box are actually present in most land plants. When presented with upstream sequences from four species of dicotelyledons, the algorithm found three motifs

Group of species	Conserved regions	# mut.	Comment
Dicotyledons (4)	TATATATAG	0	TATA-box
	AGATAAGAT	0	I-box
	CACGTGGCA	1	G-box
Land plants (10)	GATAAGAT	5	I-box
	TATATATA	6	TATA-box

Table 1: Significantly well conserved sequences in upstream region of *rbcS* in plants. Dicotyledons include *M. crystallinum*, *N. plumbaginifolia*, *G. hirsutum* and *P. sativum*. Land plants further include *L. laricina*, *L. gibba*, *Z. mays*, *B. napus*, *P. vittata* and *T. aestivum*. The three sequences found in dicots correspond to the only three promoters known to be common to the four species, and similarly for the ten land plants.

that were significantly well conserved, exactly those three mentioned above (see Table 1). Similarly, on a set of ten upstream sequences from a wide variety of land plants (5 dicotyledons, 3 monocotyledons, one conifer and one fern) only two regions appeared significantly well conserved: the TATA-box and the I-box. In both cases, the phylogenetic tree used by the algorithm was the one built from the taxonomic classification of the organisms.

3.2. *Drosophila*'s alcohol dehydrogenase

The alcohol-dehydrogenase gene (*adh*) and its upstream region have been sequenced for a large number of fruit flies (*Drosophila*). The phylogeny relating these species is very well studied (Flybase 2000). Experiments by Moses *et al.* (1990) have identified promoter sequences present in both *Drosophila melanogaster* and *Drosophila orena*, within the 1200bp upstream of the start codon of *adh* (see Table 2). It is not known whether these promoters are also present in other species of *Drosophila*. We ran the algorithm on the 1200bp upstream sequences of 12 different species of *Drosophila*. Table 2 shows the regions found to be significantly well conserved, as well as regions known to have regulating functions in both *D. melanogaster* and *D. orena*. The method produced 15 contiguous conserved regions. Among these, 10 have more than 50% of their nucleotides in annotated promoters. Overall, 51% of predicted conserved nucleotides belong to known promoters. Notice that this success rate is much higher than what would have been obtained by randomly choosing "conserved regions" in the upstream sequence, which would produce a success rate of 31%. Finally, the set of promoters identified by Moses *et al.* (1990) contains many promoters present in *D. melanogaster* but not in *D. orena*, and vice versa. This suggests that the promoters not reported by our algorithm are absent in a number of the other species considered. More generally, we see which promoters are and are not well conserved in all 12 species.

3.3. Chloroplast genome

Chloroplast-encoded genes have been sequenced in a wide variety of plants, often for phylogeny purposes (Klegg 1993). In many cases, the upstream sequences of the genes are also known, and we applied our algorithm on these data sets. Only a small fraction of the genes studied turned out to contain significantly well conserved regions. Those that did are listed in Table 3. The fact that so few genes appear to have promoters can be explained in part by the fact that chloroplast genes are often contained in operons of a large number of genes. This implies that only the first gene of the operon would appear to have promoters, which actually regulate all genes in the operon.

The chloroplast genome is assumed to have prokaryotic origin, and so it is no surprise to find the Shine-Dalgarno sequence (Shine & Dalgarno 1974) to be well conserved in *psaA*. The motifs found in *psbA* and *psbB* are, as far as we know, not documented. The *rbcL* gene has been sequenced in a very large number of species, and an upstream sequence of 600 nucleotides or more is known in 98 species of euphyllophyta. The algorithm run on this dataset produced 5 conserved regions. Three of these are documented by Manen *et al.* (1994). One of these is a known promoter for *rbcL*. Another is the reverse complement of a known promoter of *atpB*, which is the gene upstream of *rbcL* on the chromosome, on the reverse strand. A third region is part of a well known conserved region just upstream of the start codon. It is not known whether the two other motifs found have any function.

3.4. Performance

The algorithm described above has been implemented in C++ and the program is available from the authors. The running time is mainly affected by two factors, k (the size of the motif), and the score of the motif found (since the pruning method of Section 2.2 finds well conserved motifs very fast). The running times needed to obtain the results presented in this section are reported in Table 4. The

```
-1200  attcacaaggtcgaagctcttagcgttctgactcggggtgctacactgcacaaaattaca
-1140  ttatgcattcttcaaatatccctaatataccaaatatgtattaaaaagtgatcatttacc
-1080  gatcgttcgaaagacggtataggcttacaaaaaaattgccaaagtaatttacaaattcat
-1020  aatggtttttcaagtatataatatgtacatcaatattttcttacatgtattttatggga
-960   tgattatgttttaaactacactcaatttTTTCTCAGTGCACTTCCTGGTgttccattttc
-900   tattgggtccgtACCCGGCGTTTGTTTGCAGATCACTTgcttgccTATTATTATAGCAtt
-840   TTACACATTAcaaaattctggacgtcGCTGCTGCAgccgctGTCGACGTCAACTGCACTC
-780   GCCccCACGACAAAACGGTATTTAAGGCGCtgtgcaagtcccagtcgccgattattGTCT
-720   CAGTGCAGTTGTCAGTTGCAGTTcagcagaccggctagcgagtacttgcatctcttCAAA
-660   TTTACCTAATTGATCAAGTaagtgggaaagggacctcattatgcaagtgcgaatagtaag
-600   agatcactatcactaatggtggagcataataaaatcaattgcatgcaatcgaaatgAATG
-540   CAAACCGGCACaagcagtagcaaacctagtaaacaaattaaaatttggaggcctgtgccg
-480   tggcgaatatttgacttggaAAATCACCTGTTGTTTaaccgctaaaaatagGAATTTTAA
-420   cattaagcacccctgttaatcggcgccgtgccttcgttagctatctcaaaagcgcgcgcg
-360   tgcagacgagcagtaattttccaagcatcaggcatagTTGGGCATAAATTATAAACATac
-300   aaactGAATACTAATATAGAaaaaGCTTTGCCGGCAcaaaatccaaacaaaaacaaaga
-240   gagtgccaaaaataaaacaaaaataaaccgtaaaccGAGCAGCGTTGCCGTCGTTGCGgg
-180   ctgtgaagcttacgtgaatagccgaGAGATCGCGTAATGATAGATAAAGAAAGCTCTACG
-120   TAAgcgaagcttctgggggatagatcttccTATAAATAcgggaccgacgcgaactggaaa
-60    cgaacaactaacggagccctcttcccattgaaacagatcgaaagagcctgctaaagcaat
```

Table 2: Upstream region of the *adh* gene of *Drosophila orena*. Known promoters common to *orena* and *melanogaster* are in capital letters. A nucleotide is underlined if it is included in at least one significantly well conserved region of size 10. In this case, a region was significantly well conserved if it contained fewer than 5 mutations. The 12 *Drosophila* species compared are *ambigua, guanche, subobscura, erecta, orena, teissieri, mauritania, sechellia, melanogaster, simulans, funebris* and *mettleri*.

Gene	Species	Conserved regions	# mut.	Literature
psaA	8 Viriplanta + Synechosystis	AGAGGAGGAC	6	Shine-Dalgarno (1974)
psbA	8 Embryophyta	TTGGTTGACA	1	Unknown
		ATAAACCAAG	2	Unknown
psbB	13 Eukaryota	AATAAAGT	5	Unknown
rbcL	98 Euphyllophyta	TTTACATA	7	*atpB* promoter (Manen, Savolainen, & Simon 1994)
		TATACAATA	9	Similar to -10 prokaryotic promoter (Manen, Savolainen, & Simon 1994)
		AATCAAAT	7	Unknown
		GTTGATAAT	9	Unknown
		AATTCTTAATTCAT	9*	Highly conserved leader region (Manen, Savolainen, & Simon 1994)

Table 3: Significantly well conserved regions in upstream sequences (200bp, except for *rbcL*, 600bp) of chloroplast encoded genes. Similar experiments produced no significantly well conserved regions for chloroplast genes *atpA, atpB, atpH, matK, psaI, psaJ, petA, rpoB, rpl23, rps4* and *rps7*. (*) For this sequence, each substring of length 9 occurred with 9 or fewer mutations.

main constraint on the performance is the amount of memory needed, since a table with 4^k entries is associated with each node of the tree. However, only 2 tables need to be in primary memory at the same time: one for the child and one for the current node considered. Hash tables and other more sophisticated data structures could be used to reduce the amount of memory needed, in conjunction with the lower bounds described in Section 2.2.

3.5. Motif significance

Once our algorithm has been run on a set of sequences, we need a way to assess the significance of the motifs it has found. Indeed, there will be "best conserved regions" in any set of sequences, no matter if these regions have actually been protected from mutations by selective pressure. To assess the significance of a best conserved region R from sequences $s_1, ..., s_n$ on tree T, one would like to know what is the probability that sequence evolution *without* selective pressure would have produced sequences whose best conserved region has as good a parsimony score as R. If one knew the distribution of parsimony scores in the case of no selective pressure, one could readily obtain a significance score for a potential motif. The problem is that this distribution depends on a very large number of factors, both known (such as the sequence length l, the motif size k and the tree topology) and unknown (such as the branch lengths of T, the type of mutations along each branch, etc.). For this reason, it will be impossible to calculate this distribution analytically and also very difficult to approximate it empirically.

Let us assume that we had access to sequences $r_1, ..., r_n$ that were produced by a similar evolutionary process as $s_1, ..., s_n$, but where no region of the sequence was subject to any selective pressure. Then, one could compare the parsimony score of the best motif found in $s_1, ..., s_n$ to the one found in $r_1, ..., r_n$. If the former score were significantly better than the latter, we would conclude that some regions of $s_1, ..., s_n$ were subject to selection. The problem is that sequences $r_1, ..., r_n$ are hard to find. One possibility would be to take these sequences from the same organisms as $s_1, ..., s_n$, but in a region known to contain no conservation and known to have evolved from a common ancestor. Introns could be interesting candidates but, even there, it seems difficult to guarantee that no selective pressure restricted mutations.

Thus, we have to settle for a more approximate solution. We generate sequences $r_1, ..., r_n$ so as to mimic the evolution of $s_1, ..., s_n$, but without selective pressure on any locus. Let $div(T, s_1, ..., s_n)$ denote the amount of divergence that occurred during evolution of $s_1, ..., s_n$ on tree T. $div(T, s_1, ..., s_n)$ could be the sum of the branch lengths of a least-squares (Fitch & Margoliash 1967) or a minimum evolution tree (Felsenstein 1988) obtained from pairwise alignment scores of $s_1, ..., s_n$. For our simulations, we used the minimum evolution tree measure. We generate a set G of p sets of sequences $r_1, ..., r_n$ with divergence similar to $s_1, ..., s_n$, using the algorithm in Figure 2.

The distribution of the parsimony score of the most conserved regions is then computed on the sequences from G, using the algorithm of Section 2. This distribution will be our estimate of the score distribution under the "no-selection" hypothesis. The nature of the mutations performed (substitutions or indels) during the simulation process is not specified in the algorithm. We have very little information about what type of mutations happened on each branch on the tree during evolution of $s_1, ..., s_n$. Notice that, because we are looking for gapless conserved regions, indels should produce a much larger increase in the parsimony score of the best conserved region than substitutions. Indeed, simulations showed that, for the same amount of divergence, the most conserved region had a better score in sequences that had been generated using only substitutions than in those generated using both substitutions and indels. Consequently, if we use only substitutions to generate our simulated sequences, the assessment of significance of motifs found in $s_1, ..., s_n$ will be conservative, in the sense that we will not overestimate the significance of a motif. Finally, in an attempt to mimic s_i's evolution as well as possible, the initial ancestral sequence and the mutations performed during the simulation of evolution are chosen so that the resulting sequences $r_1, ..., r_n$ have the same nucleotide frequency as $s_1, ..., s_n$. A motif was judged significant if fewer than 2% of the simulated sequences obtained a better score than $s_1, ..., s_n$ did.

4. Future work

The algorithm presented in Section 2 is very general, in the sense that it could apply to a large number of interesting metrics, for example edit distance. It appears, however, that the main problem when dealing with these new metrics would be the assessment of significance of the motifs found. On the other hand, if one had *a priori* knowledge that the sequences considered *do* contain a conserved region, then the significance issue is less important. There is certainly interesting and useful research to be done in this direction.

Another idea is that, when finding well conserved motifs, one would like to take into account the background distribution of the nucleotides in $s_1, ..., s_n$, so that an `AAAAAA` motif would score better in se-

Gene	n	l	k	#mut.	Time (sec)	Space (Mb)
rbcS	4	500	9	0	0.3	2.1
rbcS	10	500	8	5	2.5	2.9
adh	12	1200	10	2	7.2	27
psaA	8	200	10	6	41	19
psbA	8	200	10	1	1.5	17
psbB	13	200	8	5	3.3	2.9
rbcL	98	600	8	5	28	11
rbcL	98	600	9	6	132	36

Table 4: Running time and memory usage of the algorithm on upstream sequences studied, on a Pentium II 400MHz. The determinant factors are k and the score of the motif found.

Algorithm Validation set generation (input: $s_1, \ldots, s_n, T = (V, E, r)$);

$G := \emptyset$;
while $|G| \neq p$ *do*
    Initialize the root r of T with a random sequence;
    Simulate evolution over T, by performing m mutations per branch,
        where m is chosen uniformly at random between 0 and B,
        for each branch, producing sequences $r_1, ..., r_n$;
    if $|div(T, s_1, ..., s_n) - div(T, r_1, ..., r_n)| < \alpha$, add $(r_1, ..., r_n)$ to G;

Figure 2: Algorithm producing selection-free sequences similar to $s_1, ..., s_n$. B is chosen so that $div(T, r_1, ..., r_n)$ is close to $div(T, s_1, ..., s_n)$, on average. To assess motif significance, we use $p = 100$ and $\alpha = 0.02 * div(T, s_1, ..., s_n)$.

quences where A is rare than in those where A is very common. It is not clear how one should amalgamate the mutation score with the background distribution score. Nonetheless, such a method would probably have eliminated some of the predictions the algorithm made in *Drosophila*'s *adh* upstream sequences (which are A-rich), as at least three of the possibly spurious regions consisted mostly of A's.

Another attractive idea would be to have an algorithm that would find conserved motifs present in *most*, but maybe not all, of the sequences, thus allowing some phyla to lose the promoter. Again, the modifications required to the algorithm would be relatively minor.

Finally, in the case where we are looking for a set of promoters regulating a gene, one could make use of the knowledge that not only are promoters conserved in their sequences, but also the order in which they appear in the upstream sequence is likely to be conserved. This could allow us to reject spurious conserved regions that appear in different orderings in different sequences.

From an application point of view, there is nothing that limits our algorithm to find only promoters. Indeed, it might be applied successfully for finding interesting regions in RNA structures or introns. It could also be used to find active sites or secondary structure elements in proteins, although the factor of 4^k in the running time would be replaced by 20^k.

Acknowledgments

We thank Amir Ben-Dor, Jeremy Buhler, Joe Felsenstein, and Rimli Sengupta for helpful discussions and encouragement.

References

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

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

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

Brāzma, A.; Jonassen, I.; Vilo, J.; and Ukkonen, E. 1998. Predicting gene regulatory elements *in silico* on a genomic scale. *Genome Research* 15:1202–1215.

Caprara, A. 1997. Sorting by reversals is difficult. In *RECOMB97: Proceedings of the First An-*

nual International Conference on Computational Molecular Biology, 75–83.

Chu, S.; DeRisi, J.; Eisen, M.; Mulholland, J.; Botstein, D.; Brown, P. O.; and Herskowitz, I. 1998. The transcriptional program of sporulation in budding yeast. *Science* 282:699–705.

Felsenstein, J. 1988. Phylogenies from molecular sequences: inference and reliability. *Annual Review of Genetics* 22:521–565.

Fitch, W. M., and Margoliash, E. 1967. Construction of phylogenetic trees. *Science* 155:279–284.

Fitch, W. M. 1971. Toward defining the course of evolution: Minimum change for a specified tree topology. *Systematic Zoology* 20:406–416.

Flybase. `http://flybase.bio.indiana.edu`

Fraenkel, Y. M.; Mandel, Y.; Friedberg, D.; and Margalit, H. 1995. Identification of common motifs in unaligned DNA sequences: application to *Escherichia coli* Lrp regulon. *Computer Applications in the Biosciences* 11(4):379–387.

Galas, D. J.; Eggert, M.; and Waterman, M. S. 1985. Rigorous pattern-recognition methods for DNA sequences: Analysis of promoter sequences from Escherichia coli. *Journal of Molecular Biology* 186(1):117–128.

Genbank. `http://www.ncbi.nlm.nih.gov/Genbank`

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

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

Kerlavage, A. R. 1999. Computational genomics: Biological discovery in complete genomes. In *Proceedings of the Seventh International Conference on Intelligent Systems for Molecular Biology*, xii. Heidelberg, Germany: AAAI Press.

Klegg, M. T. 1993. Chloroplast gene sequences and the study of plant evolution. *Proceedings of the National Academy of Science USA* 90(2):363–367.

Lawrence, C. E., and Reilly, A. A. 1990. An expectation maximization (EM) algorithm for the identification and characterization of common sites in unaligned biopolymer sequences. *Proteins: Structure, Function, and Genetics* 7:41–51.

Lawrence, C. E.; Altschul, S. F.; Boguski, M. S.; Liu, J. S.; Neuwald, A. F.; and Wootton, J. C. 1993. Detecting subtle sequence signals: a Gibbs sampling strategy for multiple alignment. *Science* 262:208–214.

Magnuson, M., and Shelton, K. 1989. An alternate promoter in the glucokinase gene is active in the pancreatic beta cell. *Journal of Biological Chemistry* 264(27):15936–15942.

Magnuson, M., and Thomas, J. 1993. Evolutionary conservation of elements in the upstream glucokinase promoter. *Biochem. Soc. Trans.* 21(1):160–163.

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

Moses, K.; Heberlein, U.; and Ashburner, M. 1990. The *adh* gene promoters of *Drosophila melanogaster* and *Drosophila orena* are functionally conserved and share features of sequence structure and nuclease-protected sites. *Molecular and Cellular Biology* 10(2):539–548.

Roth, F. P.; Hughes, J. D.; Estep, P. W.; and Church, G. M. 1998. Finding DNA regulatory motifs within unaligned noncoding sequences clustered by whole-genome mRNA quantitation. *Nature Biotechnology* 16:939–945.

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

Shine, J., and Dalgarno, L. 1974. The 3′-terminal sequence of *E. coli* 16S ribosomal RNA: Complementarity to nonsense triplets and ribosome binding sites. *Proceedings of the National Academy of Science USA* 71:1342–1346.

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

Spellman, P. T.; Sherlock, G.; Zhang, M. Q.; Iyer, V. R.; Anders, K.; Eisen, M. B.; Brown, P. O.; Botstein, D.; and Futcher, B. 1998. Comprehensive identification of cell cycle-regulated genes of the yeast *Saccharomyces cerevisiae* by microarray hybridization. *Molecular Biology of the Cell* 9:3273–3297.

Staden, R. 1989. Methods for discovering novel motifs in nucleic acid sequences. *Computer Applications in the Biosciences* 5(4):293–298.

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

van Helden, J.; André, B.; and Collado-Vides, J. 1998. Extracting regulatory sites from the upstream region of yeast genes by computational analysis of oligonucleotide frequencies. *Journal of Molecular Biology* 281(5):827–842.