# An Efficient Algorithm for Finding Highly Conserved Regulatory Elements among Orthologous DNA Sequences

Shyong Jian Shyu Chun-Kai Yang

*Department of Computer Science and Information Engineering*
*Ming Chuan University*
*5 Teh-Ming Rd., Gwei Shan, Taoyuan, Taiwan 333, R. O. C.*

*sjshyu@mcu.edu.tw* *ckyang@turing.csie.mcu.edu.tw*

***Abstract-*** *In current genome research, the identification of regulatory elements required for the correct expression of genes is an essential topic. FootPrinter* (*Blanchette, 2000*; *Blanchette and Tompa*, *2003*) *is a well established tool for identifying regulatory elements from a set of orthologous non-coding DNA sequences of various species under a given phylogenetic tree. Such a motif/substring finding problem has been defined as the substring parsimony problem by Blanchette* (*2000*)*. We design a new algorithm which adopts the hashing technique and a fast approach to detect the Hamming distance between two substings to resolve this problem. Experimental results show that our approach is more efficient than FootPrinter when the substrings needed are highly conserved.*

**Keywords**: Regulatory element, Hamming distance, Substring parsimony problem, Hashing.

## 1. Introduction

To understand how gene expression is regulated is an essential challenge of current genomics. The first thing to do for such understanding is the ability to identify regulatory elements associated with a given gene. Most of these regulatory elements are relatively short stretches of DNA (5 to 25 nucleotides long), located in the non-coding sequence surrounding a gene (Tompa, 1999; Blanchette *et al*, 2000). Most known regulatory elements are located 5' of the coding region, but some are also found in the 3' sequence, or even in introns. In all these cases, regulatory elements are located in otherwise non-functional sequences.

*Phylogenetic footprinting* was first proposed by Tagle *et al*., (1988), which is a technique that uses such a functional/non-functional sequence dichotomy to identify regulatory elements. The idea underlying phylogenetic footprinting is that selective pressure causes regulatory elements to evolve at a slow rate than the non-functional surrounding sequence. Therefore the best conserved motifs in a collection of homologous regulatory regions are excellent candidates as regulatory elements (Blanchette *et al*, 2002).

This technique of phylogenetic footprinting was further implemented as a tool, named as FootPrinter (Blanchette and Tompa, 2003), It can not only identify many known functional binding sites but also find several highly conserved motifs. It is effective to predict new unknown regulatory elements. The empirical studies by Blanchette, Kwong and Tompa (2003), which evaluate the accuracy of the motif-finding tools including FootPrinter, MEME (Bailey and Elkan, 1995) and Dialign (Morgenstern *et al*., 1998) on synthetic and real biological data, reveal that FootPrinter would be the most accurate tool in identifying motifs in most cases. One of the main reasons for its good performance is that it takes into account the phylogenetic relationships of species.

It is our aim in this paper to improve the execution time needed by FootPrinter, especially when dealing with those highly conserved regulatory elements among orthogonal sequences. The rest of the paper is organized as follows. Section 2 defines the substring parsimony problem formally and describes briefly the essential idea of FootPrinter. Our method which is specifically designed for highly conserved short motifs is presented in Section 3. The experimental results are summarized in Section 4. Section 5 gives the concluding remarks.

## 2. The Substring Parsimony Problem

From the computational viewpoint, the well conserved motifs finding problem is as follows. Given a set of orthologous sequences $S = \{s_1, s_2, ..., s_n\}$ from $n$ related species, we are looking for sequences $t_1, t_2, ..., t_n$, where $t_i$ is a substring with length $k$ for all $1 \le i \le n$ such that $t_1, t_2, ..., t_n$ have an unusual high measure of sequence similarity. To avoid the overrepresentation of the problem, we do not weight the $n$ sequences equally, but instead we assume a good phylogenetic tree with the $n$ species at its leaves is given and we would like to measure the mutual sequence similarity by parsimony.

This problem was formally defined as the *substring parsimony* problem (Blanchette, 2002) which has been shown to be NP-hard (Akutsu, 1998). The substring parsimony problem is defined as follows:

**Given**: a set of orthologous sequences $S = \{s_1, s_2, ..., s_n\}$ from $n$ different species, the phylogenetic tree $T=(V, E)$, $V=S$, relating these species, the length $k$ of the motifs to look for, and an integer $d$.

**Problem**: find all sets of substrings $t_1, t_2, \ldots, t_n$ of $s_1, s_2, ..., s_n$ respectively, each of length $k$, such that the parsimony score of $t_1, t_2, \ldots, t_n$ on $T$ is at most $d$.

Note 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 defined as the minimum, over all possible labelings of the internal nodes with sequences of length $k$, of the sum of the Hamming distance between the labels of the nodes connected an edge in $T$. In more detail, suppose the internal nodes are labeled as $n+1, n+2, ..., |V|$. We would like to find $t_1, t_2, \ldots, t_n$ of $s_1, s_2, ..., s_n$ respectively and $t_{n+1}, t_{n+2}, \ldots, t_{|V|}$ that minimize

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

where $\delta(t, t')$ is the Hamming distance between string $t$ and $t'$, i.e., the number of positions at which they differ. Looking for sets of substrings that achieves a low parsimony score $P(T)$ corresponds to searching for highly conserved regions.

Blanchette *et al*. (2000) introduced a dynamic programming algorithm to solve the substring parsimony problem optimally in time $O(n\times k\times(4^{2k}+l))$ (which was further improved as $O(n\times\min(l\times(3k)^{d/2}, k\times(4^k\times l)))$ by Blanchette (2001)), where $l$ is the average length of all sequences in $S$. Let $C(v)$ be the set of children of $v$ and $\Sigma = \{\mathtt{A}, \mathtt{C}, \mathtt{G}, \mathtt{T}\}$. The dynamic programming algorithm for this problem is as follow.

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

The algorithm proceeds from the leaves up to the root. At each node $v$ of the tree, computing a table $W_v$ containing $4^k$ entries, one for each possible sequence of length $k$. For a string $s$ of length $k$, $W_v[s]$ is defined as the best parsimony score that can be achieved for the subtree rooted at $v$, if $v$ was to be labeled with $s$.

Please refer to Blanchette (2001) and Blanchette *et al.* (2002) for a more detail review of the substring parsimony problem and some improved techniques for solving this problem.

Figure 1 illustrates an example of the substring parsimony problem where the parsimony score of the input five sequences under the given tree is 1 which is the minimum score (number of substations) for `AGTCG` (substring labeled at all of the internal nodes) to be the substring of all the five input. Note that the value annotated on the edge $(u, v)$ denotes the minimum score from the ancestor node $u$ to the descendant node $v$ and the substring labeled at the internal node, say $u$, is the one with the minimum score within the subtree rooted at $u$.

Figure 2 shows the straightforward algorithm.

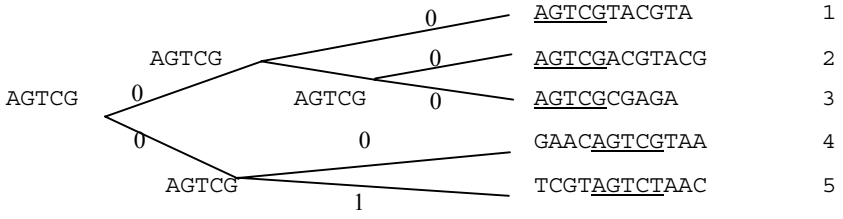


**Figure 1.** Example of a substring parsimony problem

**Algorithm** substring parsimony
**Input**: $S=\{s_1, s_2\ldots, s_n\}$, $k\in IN$, $T = (V, E)$ with the root labeled as $r$
**Output**: find all sets of substrings $t_1, t_2,\ldots, t_n$ in $s_1, s_2\ldots, s_n$ respectively, each of length $k$, such that the parsimony score of $t_1, t_2, \ldots, t_n$ on $T$ is minimized.

```
1   for (each leaf v of T) do
2       let W_v[t] = 0 for each k-substring t of S_v; W_v[t] = ∞ otherwise;
3   for (each internal node u of T, from the leaves toward the root r) do
4       for (each sequence t∈Σ^k) do
5           compute W_u[t] = Σ_{v∈C(u)} min_{t'∈Σ^k}(W_u[t']+δ(t, t'));
6   select a t_r∈Σ^k such that W_u[t_r] is minimal;
7   for (each child v of a node u, from the root r toward the leaves) do
8           choose t_v such that W_v[t_v]+d(t_u, t_v) is minimal;
```

**Figure 2.** Straightforward algorithm by Blanchette *et al*. (2000)

## 3. Our Method for the Substring Parsimony Problem

From Figure 2, we find that the algorithm generates all substrings with length $k$ (step 5) for each sequence (step 4) and all internal nodes (step 3). When the input sequences are highly conserved or we only need to find motifs among the sequences with a small parsimony score, it might be more efficient to compute the parsimony score of each node on the evolutionary tree by starting from some motif, say $t$, of a sequence and test whether itself or its *neighbors* (substrings $t'$ such that $\delta(t, t') \le d$) are already contained in all the other sequences, assuming the test process can be accelerated effectively. That is, our idea is to focus on a substring $t$ with length $k$ with its $3k$ neighbors, referred to as $N$ containing the substrings $t'$ such that $\delta(t, t') \le d$, and to check first whether these candidates need to compute their scores further. Our expectation is that no further scoring computations are needed if some, even most, of them are no longer legal candidates.

There are two elementary operations that occur heavily in the substring parsimony problem: (1) checking whether a substring is in some other sequences, and (2) determining the Hamming distance of two substrings. We apply the hashing technique and some data structures to speedup the first search process and an efficient approach to compute the Hamming distance of two substrings.

We adopts a code to represent the consecutive $k$ characters (a $k$-substring) in a sequence, thus we only use code calculation for the Hamming distance instead of using one-to-one character comparisons. Each DNA sequence with $l$ bases long would contain $(l–k+1)$ overlapping $k$-characters. We associate each letter of nucleotides with a specific code respectively, namely $s_i[j] = 0, 1, 2$ or 3 for A, C, G, T, respectively, $1 \le i \le n$ and $1 \le j \le |\, s_i \,| - k + 1$. Thus the $j$th substring of sequence $s_i$ can be coded as a number $K[i, j]$, which is defined as

$$K[i, j] = \sum_{l=0}^{k-1} s_i[j+l] \,|\Sigma|^{k-l-1}$$

where $s_i[j+l]$ is the code corresponding to the $(j+l)$th character of sequence $s_i$, $|\,\Sigma\,|$ is the number of symbols in the alphabet, for DNA sequences $|\Sigma| = 4$, $0 \le K[i, j] \le |\,\Sigma\,|^k - 1$. For example, when $k = 3$, substring ACG in sequence $s_i$ is represented as 6. We then score $(i, 6)$ in a hash table $H$ with size *hash_size*, in fact, $(i, 6)$ is append into a list pointed by $H[K[i, j] \bmod hash\_size]$. Through such a structure, the existence of some $k$-substring in a sequence can be easily detected.

Since the parsimony score is required to be less than $d$, the substrings we are looking for should satisfy $\delta(t_i, t_j) \le d$ where $t_i$ and $t_j$ are substrings of $s_i$ and $s_j$ respectively, $1 \le i, j \le n$. Starting from some $k$-substring $p$ of a sequence $s_t$, our approach generates all of the $k$-substrings with distance at most $d$ for a certain substring $p$, then we calculate the parsimony scores between $p$ and all of the substrings in the sequences other than $s_t$.

The detail of our approach is described in the pseudo codes as follows:

**Algorithm** *Our approach*
**Input:** A set of orthologous sequences $S=\{s_1, s_2 \ldots, s_n\}$ from $n$ different species, the length $k$ of motifs to look for, and an integer $d \ge 1$
**Output:** find all sets of substrings $t_1, t_2, \ldots, t_n$ in $s_1, s_2 \ldots, s_n$ respectively, each of length $k$, such that the parsimony score on $T$ is at most $d$.

1 **for** ($s_i \in S$, $1 \le i \le n$) // coding all $k$-substrings among $S$
  $b = 0$
  **for** ($1 \le j \le (|\, s_i \,| - k + 1)$)
    **if** ($s_i[j \sim j+k-1]$ is a new $k$-substring of $s_i$) // Calculate the code $K_i[j]$ for the $j$th substring of $s_i$
      $K[i, b] = \sum_{l=0}^{k-1} s_i[j+l] |\Sigma|^{k-l-1}$
      insert $(i, K[i, b])$ into $H[K[i, b] \bmod hash\_size]$
      $b$++
    **endif**
  **endfor**
**endfor**
2 // Initialization
  Let $s_t$ be the sequence with the minimum number of $k$-substrings of $S$
  Let $L$ be the number of distinct $k$-substrings of $s_t$
  $j = 0$
3 **while** ($j < L$)
3.1 generate all $m$, $d(m, K[t, j]) \le d$, into a set ***N*** // ***N*** contains the codes of substrings $x$'s, $x \in N$
3.2 *Compute_Score* = **true**
3.3 **for** (each $s_i \in S$ and *Compute_Score*) **do**

```
            if (s_i does not contain all substrings in N)  Compute_Score = false
        endfor
3.4     if (Compute_Score) then
            Let α be the leave of T which is labeling with s_t
            W_α[K[t, j]] = 0                                  // assume K[t, j] to be the candidate motif in s_t
            for (each m ∈ N) do  X_α[m] = δ(K[t, j], m)       // N is the set of codes of the neighbors of α
            for (each leave v≠α of T) do
                for (each m_v∈N and m_v is a substring in v) do W_v[m_v] = 0
                for (each m ∈ N) do X_v[m] = min δ(m_v, m), where m_v∈N and m_v is a substring in v
            endfor
            for (each internal node u≠r of T and Compute_Score) do
                for (each m ∈ N) do W_u[m] = X_u[m] = Σ_{v∈C(u)} X_v[m]
                if (min_{m∈N}(W_u[m]) > d) then  Compute_Score = false
                for (each m′ ∈ N with W_u[m′] ≤ d) do
                    for (each m ∈ N with X_u[m] > W_u[m′]) do
                        X_u[m] = min(X_u[m], min(W_u[m′]+δ(m, m′)))
            endfor
            if (Compute_Score) then
                for (each m ∈ N) do  W_r[m] = Σ_{v∈C(r)} X_v[m];
                select all m_r∈ N such that W_r[m_r] ≤ d;
                for (each v∈C(u) of an internal node u, from the root r toward the leaves) do
                    Choose m_v such that W_v[m_v] + δ(m_v, m_u) is the minimal;
            endif
        endif
3.5     j++
    endwhile
```

Note that the above algorithm deals the cases with $d \geq 1$. When the parsimony score is restricted to be 0, we have a more efficient algorithm to this restricted problem. The pseudo codes are described as follows:

**Algorithm** *Our approach for d=0*
**Input:** A set of orthologous sequences $S=\{s_1, s_2, \ldots, s_n\}$ from $n$ species, the length $k$ of motifs to look for
**Output:** find all sets of substrings $t_1, t_2, \ldots, t_n$ in $s_1, s_2 \ldots, s_n$ respectively, each of length $k$, such that the mutation of substrings is zero

```
1       Let l_i be the length of ith sequence in s_1, s_2, …, s_n
2       for (i = 0 to 4^k–1) do P[i] = 0
3       for (i = 1 to n) do
4           for (j = 0 to l_i–k) do
4.1             K_j^i = Σ_{l=0}^{k–1} c_{j+l}^i | Σ |^{k-l-1}   // code K_i for the sunstring at j position in the ith sequence
4.2             if (P[K_j^i] == i–1) then
                    P[K_j^i]++
                    if (P[K_j^i] == n) then the substring represented by K_i is a solution; report it
                end_if
            end_for
        end_for
```

## 4. Experimental Results

To examine the performance of our approach when applied to find regulatory elements among orthogonal sequences, we use two kinds of test sets: (1) DNA sequences randomly selected from NCBI (National Center for Biotechnology Information); and (2) real sequences reported in the literature, namely (Blanchette and Tompa, 2002) whose regulatory elements have been known.

The experimental platform is a personal computer with an AMD Athlon 2100+ CPU and 1GB RAM running the red head Linux. Our program is coded in C. Table 1 shows four test sets in our experiments. All of these data sets are DNA sequences randomly selected from the query results of searching for *rbc-L* in NCBI. Note that $n$ denotes the number of sequences and Length indicates the range of the minimum and maximum lengths of the sequences in the data set.

**Table 1.** Test sets

| Data Set | *n* | Length | Source |
|---|---|---|---|
| #1 | 20 | 628~2367 | randomly selected from the results of querying *rbc-L* in NCBI |
| #2 | 30 | 310~782 | |
| #3 | 50 | 310~820 | |
| #4 | 100 | 999~2095 | |

**Table 2.** CPU time results

| *d* | Data set | *k* | CPU time (sec.) | |
|---|---|---|---|---|
| | | | *Our* | *FP* |
| 0 | #1 | 8 | 0.00 | 1.05 |
| | | 10 | 0.00 | 1.05 |
| | | 12 | 0.01 | 1.05 |
| | #2 | 8 | 0.00 | 1.23 |
| | | 10 | 0.00 | 1.23 |
| | | 12 | 0.00 | 1.22 |
| | #3 | 8 | 0.01 | 2.12 |
| | | 10 | 0.00 | 2.11 |
| | | 12 | 0.00 | 2.09 |
| | #4 | 8 | 0.02 | N/A |
| | | 10 | 0.04 | N/A |
| | | 12 | 0.04 | N/A |
| 1 | #1 | 8 | 0.06 | 1.69 |
| | | 10 | 0.09 | 1.85 |
| | | 12 | 0.33 | 2.01 |
| | #2 | 8 | 0.04 | 7.48 |
| | | 10 | 0.05 | 1.36 |
| | | 12 | 0.22 | 1.39 |
| | #3 | 8 | 0.04 | 2.30 |
| | | 10 | 0.06 | 2.34 |
| | | 12 | 0.24 | 2.36 |
| | #4 | 8 | 0.36 | N/A |
| | | 10 | 0.31 | N/A |
| | | 12 | 0.52 | N/A |
| 2 | #1 | 8 | 0.96 | 5.18 |
| | | 10 | 0.61 | 5.24 |
| | | 12 | 1.29 | 5.99 |
| | #2 | 8 | 0.93 | 2.99 |
| | | 10 | 0.46 | 2.77 |
| | | 12 | 0.80 | 3.08 |
| | #3 | 8 | 1.43 | 4.26 |
| | | 10 | 0.50 | 4.71 |
| | | 12 | 0.85 | 5.23 |
| | #4 | 8 | 9.28 | N/A |
| | | 10 | 2.55 | N/A |
| | | 12 | 2.21 | N/A |
| 3 | #1 | 8 | 78.09 | 28.21 |
| | | 10 | 13.24 | 35.72 |
| | | 12 | 10.12 | 45.15 |
| | #2 | 8 | 41.75 | 120.07 |
| | | 10 | 20.90 | 15.58 |
| | | 12 | 12.11 | 9.89 |
| | #3 | 8 | 57.16 | 11.81 |
| | | 10 | 29.04 | 12.74 |
| | | 12 | 12.96 | 15.68 |
| | #4 | 8 | 520.46 | N/A |
| | | 10 | 117.75 | N/A |
| | | 12 | 18.19 | N/A |

Table 2 summaries the CPU time results of the program of FootPrinter (*FP*) and the proposed method (*Our*) on the four data sets where $d$ is the allowable parsimony score and $k$ is the length of the substrings needed.

As can be seen from Table 2, our approach outperforms *FP* when the allowed parsimony score is small (i.e., $d \leq 2$). It means that if our aim focuses on finding highly conserved substrings from a set of orthologous sequences, our approach is more efficient than *FP*. When $d$ is small, the neighbor code numbers $m$ of $M_i$ in the shortest sequence of input sequences would be quite few so that the computation time can be reduced. For the cases of $d = 0$, our method has an appealing improvement over *FP*. However, for the cases that some high parsimony score is required ($d>3$, or cases that do not look for highly conserved substrings), it would be better to use *FP*. When $d$ is large, the size of neighbor code numbers $m$ of $N_i$ in our approach increases considerably. This consumes execution time. Note that *FP* stops running due to memory faults when dealing with data sets #4 which contains 100 sequences.

We further examine the computational results of our method and *FP* on the test data sets reported in (Blanchette and Tompa, 2002) which are accessible via http://bio.cs.washington.edu/GR/. Table 3 reveals the results including the DNA regions investigated, names of the tested species, highly conserved motifs found by both *Our* and *FP* (capitalized nucleotides), parsimony score ($d$) of the capitalized motifs with respect to the whole set of species, known functional information (column "Ref." of Table 3) about the motif. Note that we tested all data sets in *FP* by choosing the option that the motifs should appear in all the sequences.

Again, from Table 3, our method is slower than *FP* for only one test instance which requires $d=4$, while our method is more efficient than *FP* when $d$ is small. Both programs report the same set of motifs for the test data sets.

## 5. Concluding Remarks

We propose a new method for the substring parsimony problem. It is more efficient than the well-known tool, FootPrinter, when the needed substrings are highly conserved with a low mutating rate among the given sequences. Experimental results reveal the efficiency of the proposed algorithm in identifying the regulatory elements in a set of orthologous sequences, especially for highly conserved substrings. For larger sets of sequences, our approach also shows the feasibility as compared to FootPrinter.

In the near future, the author would like to apply some rules to filter out substrings which are impossible to be solutions as soon as possible in order to reduce the computation time of our method

for the cases that some large parsimony score is demanded.

**Table 3**. Regulatory elements found by *Our and FP*

| DNA Region | Species | Motif (length) (position) | *d* | Ref.[#] | CPU time (sec.) | |
|---|---|---|---|---|---|---|
| | | | | | *Our* | *FP* |
| Insulin family 5' promoter (500bp) | Human, chimp, *aotus*, pig, rat (I, II), mouse (I, II) | TcagcccccaGCCATCTGCC (10)(-122) | 1 | 1.1 | 0.06 | 0.54 |
| | | CTATAAAGcc (8) (-32) | 0 | 1.2 | 0.00 | 0.31 |
| c-myc second intron (971 to 1376 bp) | Chicken, pig, rat, marmoset, gibbon, human | TAGGGAGTTG (10) (670) | 2 | | 1.12 | 3.26 |
| | | ATTTGCAGCTat (10) (698) | 2 | | | |
| | | GAAGTGTTCT (10) (725) | 2 | | | |
| | | TTCCTTTCTT (10) (1362) | 2 | 3.1 | | |
| c-fos 5' UTR + promoter (800bp) | Tetraodon, chicken, mouse, hamster, pig, human | CACAGGATGTcc (10) (-479) | 4 | 4.1 | 641.64 | 39.27 |
| c-fos first intron (376 to 758 bp) | Fugu, tetraodon, chicken, pig, mouse, hamster, human | agcgcagacgtcAGGGATATTTA (10) (472) | 1 | 5.1 | 0.05 | 0.48 |
| Interleukin-3 5' UTR + promoter (490 bp) | Rat, mouse, cow, sheep, human, macaca | GTCTGTGGTTTtCTATGGAGGT TCCATGTCAGATAAAG (8) (-195) | 0 | 6.1 | 0.00 | 0.22 |
| | | TTGAGTACTagaaagt (8) (-228) | 1 | | | |
| | | GATGAATAATt (8) (-208) | 1 | 6.2 | 0.03 | 0.38 |
| | | TCTTCAGAGc (8) (-56) | 1 | | | |
| | | AGGACCAG (8) (-40) | 1 | | | |

[#] The information comes from TRANSFAC (Wingender *et al*. 1996) with accession number in brackets. **Insulin:** 1.1 IEB1 [R04457], 1.2 TATA-box [GenBank annotation]. **C-myc:** 2.1 NHE [R01804]. **C-myc second intron:** 3.1 Part of 3' splice site. **C-fos:** 4.1 [many factors bind in this region; R00466, R00465, R00464, R01889]. **C-fos first intron:** 5.1 (Transcription elongation signals; Mechti *et al*. 1991). **IL-3:** 6.1 [R02682, R05026, R05027], 6.2 [R02736].

## 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] T. L. Bailey, and C. Elkan, Unsupervised learning of multiple motifs in biopolymers using expectation maximization, *Machine Learning*, 21:51-80, 1995.

[3] M. Blanchette, B. Schwikowski, and M. Tompa, An exact algorithm to identify motifs in orthologous sequences from multiple species, *Proc. 8th Intl. Conf. on Intelligent Systems for Molecular Biology*, 37-45, AAAI Press, La Jolla, USA, 2000.

[4] M. Blanchette, B. Schwikowski, and M. Tompa, Algorithms for phylogenetic footprinting, *J. Comput. Biol*, 9(2), 211-223, 2002.

[5] M. Blanchette, and M. Tompa, Discovery of Regulatory Elements by a Computational Method for Phylogenetic Footprinting, *Genome Research*, 12(5), 739-748, 2002.

[6] M. Blanchette, and M. Tompa, FootPrinter: a program designed for phylogenetic footprinting, *Nucleic Acids Research*, 31(13), 3840-3842, 2003.

[7] M. Blanchette, S. Kwong, and M. Tompa, An Empirical Comparison of Tools for Phylogenetic Footprinting, *Proc. 3rd IEEE Symposium on Bioinformatics and BioEngineering*, 69-78, 2003.

[8] B. Morgenstern, K. Frech, A. Dress, and T. Werner, DIALIGN: Finding local similarities by multiple sequence alignment, *Bioinformatics,* 14(3), 290-294, 1998.

[9] 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.

[10] M. Tompa, An exact method for finding short motifs in sequences, with application to the Ribosome Binding Site problem, *Proc. 7th Intl. Conf. Intelligent Systems for Molecular Biology*, 262-271, Heidelberg, Germany, 1999

[11] E. Wingender, P. Dietze, H. Karas and R. Knuppel, TRANSFAC: A database on transcription factors and their DNA binding sites. *Nucleic Acids Res.*, 24:238-241, 1996.