BIOINFORMATICS

Vol. 1 no. 1 2002
Pages 1–12

Contextual Alignment of Biological Sequences (Extended Abstract) †

Anna Gambin, Sławomir Lasota, Radosław Szklarczyk, Jerzy Tiuryn and Jerzy Tyszkiewicz

Institute of Informatics, Warsaw University, Banacha 2, 02-097, Warsaw, POLAND

ABSTRACT
We present a model of contextual alignment of biological sequences. It is an extension of the classical alignment, in which we assume that the cost of a substitution depends on the surrounding symbols. In this model the cost of transforming one sequence into another depends on the order of editing operations. We present efficient algorithms for calculating this cost, as well as reconstructing (the representation of) all the orders of operations which yield this optimal cost. A precise characterization of the families of linear orders which can emerge this way is given.†

Contact: jty@mimuw.edu.pl

INTRODUCTION

A large portion of modern computational biology is concerned with measuring the degree of similarity of biological sequences, the most prominent examples of which are DNA and proteins.

Generally, to get such a model of similarity, one assumes a set of operations, which can change sequences, and a score function, assigning a score to each operation performed on a sequence. Then each set of operations transforming a biological sequence V into another such sequence W is assigned a score — typically the sum of the scores of individual operations. The operations correspond to evolutionary changes, higher score reflects that the event is more likely to appear. Several values are then of interest, the crucial ones being the maximal possible score of a transformation of V into W and the maximal score of a transformation of a contiguous fragment of V into a fragment of W (maximized over such fragments, too).

The model dominating in the field (the so called alignment model) (Durbin *et al.*, 1998; Gusfield , 1997), used for DNA and proteins, assumes the operation of substitution of one letter for another, as well as an insertion or a deletion of a sequence of letters. The score of a substitution depends only on the two residues exchanged. It is provided by the so-called substitution tables (Dayhoff *et al.*, 1978; Henikoff *et al.*, 1992). The score for insertions and deletions depends solely on the length of the inserted/deleted subsequence (it is quite often an affine function).

Of course, the above score model is a great oversimplification from the biological point of view. However, it is the most commonly used, because it permits very efficient algorithms to compute the key values, called the maximal global and local alignment scores, respectively. Other models, which are biologically more realistic, are computationally very hard (or even provably intractable). For example, a very important property of proteins is their fold (i.e., the 3-D shape they assume in the cell), which can decide homology between proteins of quite different sequences. However, despite very intense efforts, predicting the shape of a protein, based on its sequence, is viewed as an extremely difficult problem.

Results. In this paper we offer a model, which extends the classical alignment model, with the intention to bring it a step closer to the biological reality without sacrificing its algorithmic properties. The set of operations is the same as that of the alignment model but the score function of a substitution changes. In our model the score of a substitution *depends on the surrounding letters in the sequence*, too. I.e., the score of substituting b by d in abc can differ from the score of substituting b by d in $a'bc'$. The score for insertions and deletions is inherited from the classical model. We call our model the *contextual alignment model*.

The aim of this paper is to present an efficient algorithm for calculating the maximal contextual alignment score, assuming affinity of gap penalty function. The running time of the algorithm is $O(|\Sigma|mn)$, where $|\Sigma|$ is the size of the alphabet (4 in the case of DNA, 20 in the case of proteins) and m, n are the lengths of the sequences. Hence the complexity is, up to a constant factor, the same as that of the algorithms of the classical context-free alignment model.

Another topic of our paper is the order in which operations are performed. As it is easy to see, in the contextual alignment model the score of a set of operations depends on the order. Indeed, an operation may change the contexts of future operations.

† This work was partly supported by the Polish KBN grant 7 T11F 016 21.

EXAMPLE 1. *Here we see that the relative order of two substitutions applied to the same sequence affects the score, if a contextual scoring function is used.*

$$
\begin{tikzcd}
& \boxed{abcd} \arrow[dl, "-3"'] \arrow[dr, "2"] \arrow[dd, dashed, bend right=80, "-1"'] \arrow[dd, dashed, bend left=80, "0"] & \\
\boxed{ab'cd} \arrow[dr, "2"'] & & \boxed{abc'd} \arrow[dl, "-2"] \\
& \boxed{ab'c'd} &
\end{tikzcd}
$$

Often operations performed at distant fragments of the sequence are independent in the sense that neither of them changes the context of the other. Independent operations can be performed in any order. Therefore, there are typically many orders, which give the maximal score. Thus, our algorithms find not only an optimal set of operations, but also reconstruct a precise characterization of the set of all possible orders (we call them admissible orders), in which the operations may be performed to yield the maximal score.

Summarizing, the main contributions of the paper are:

- Contextual alignment model—a new approach to measuring similarity of biological sequences.
- Efficient algorithms for constructing contextual alignments of maximal score, their scores, and (the representations of) the sets of all admissible orders of operations, which give that maximal score.
- Precise characterization of all the sets of admissible chains which correspond to complete sets of operations of a maximal score.

The paper is organized as follows. In the first section we introduce the concept of a contextual alignment. Then we describe an algorithm which finds an optimal global contextual alignment of two sequences, assuming an affine gap penalty function. The next section is devoted to discussion of the validation of our model on biological data. We discuss there the issues related to construction of contextual substitution matrices, as well as the results of comparing the scores obtained by our method with those obtained by the standard context-free method. The paper is concluded with some open problems and possible ways of continuing this approach. Due to space limitations we have skipped in this extended abstract many formal definitions, some mathematical results and proofs. They can be obtained from the full version of this paper at http://www.mimuw.edu.pl/~tiuryn/papers.html. A poster of this paper appeared as Gambin *et al.* (2002a).

Biological motivation and related work. There are numerous known examples in biology, showing that indeed a context may affect the likelihood of changes in biological sequences. One of them is the elimination of adjacent pairs cytosine-guanine in DNA, caused by biochemical mechanisms of replication. Another one is observed in proteins: substitution of a hydrophobic amino acid by a hydrophilic one in hydrophobic context with much higher probability changes the fold of the protein than an identical substitution in a hydrophilic context. If a protein changes its fold, it may lose its biological activity and thus the underlying mutation is more likely eliminated in the evolution.

We should mention that the contextual alignment we consider is an algorithmic counterpart of work already undertaken in probability theory. Recently several papers have been published (Schöniger *et al.*, 1994; von Haeseler *et al.*, 1998; Jensen *et al.*, 2000), which consider a probabilistic model, in which a biological sequence undergoes random changes due to substitutions, whose probability is context-dependent. This leads to a Markov chain model of quite a complicated structure. The questions considered in the papers are existence and characterization of the steady-state distribution, estimation of the rate of evolution, as well as estimating the size of the context, which significantly affects the substitution probabilities. E.g., Tavaré *et al.* (1989) estimate the size of the significant context for the DNA evolution in the bacteriophage λ to be 1 or 2 bases (but not 0!). This gives us another argument for considering contextual alignments, as well as for restricting our attention to contexts of size 1.

The paper Wilbur *et al.* (1984) considers contexts for comparing a pair of biological sequences. This is achieved by trying to align without gaps, in various ways, short blocks (the term used there is *aligned fragments*) of characters from each of the two sequences. Scoring of the aligned blocks is given by an external scoring function. Since each pair of blocks receives its own score, the concept of a context is thus present in that approach. This is different understanding of the context than in the present paper — our context is understood as flanking characters which may influence the likelihood of symbol substitution, while in Wilbur *et al.* (1984) the context is understood as having a direct effect on scoring pairs of aligned blocks. It follows that the two approaches, despite of using similar names, have nothing in common.

CONTEXTUAL ALIGNMENT

Let V and W be strings over an alphabet Σ. A *gap*, denoted $-$, is a symbol assumed not to belong to Σ. Let us fix an alignment $(V^\#, W^\#)$ of these two sequences. The concept of an alignment used in this paper is standard. We omit it from this presentation for the sake of space. An alignment induces three kinds of blocks, each block has its unique *address* and unique *length*:

- A *substitution* has an address i, if $V_i^\#, W_i^\# \in \Sigma$, i.e. if they are not gaps. Substitutions are one element blocks.
- An *insertion* has an address i, if $V_{i-1}^\# \in \Sigma$, $V_i^\# = -$. The length of the insertion block is the least $j > 0$ such that $V_{i+j}^\# \in \Sigma$.
- A *deletion* has an address i, if $W_{i-1}^\# \in \Sigma$, $W_i^\# = -$. The length of the deletion block is the least $j > 0$ such that $W_{i+j}^\# \in \Sigma$.

Following the above classification of blocks we have three kinds of *operations*, each associated with one block.

- (*Substitutions*) $S_{i,a,b}$, where $1 < i < n$ is an address of a substitution block and $a, b \in \Sigma$. The characters a, b are called *contexts*. There are also two outermost substitutions: S_1 and S_n.
- (*Insertions*) I_i, where i is an address of an insertion block.
- (*Deletions*) D_i, where i is an address of a deletion block.

The insertions, deletions and outermost substitutions are taken without any context.

The *cost* $c(o)$ of each operation o can be read off from the address of the corresponding block and from the alignment $(V^\#, W^\#)$. It also depends on the *gap penalty function* g and on the *contextual substitution tables* $M_{a,b}(\cdot,\cdot)$, where a, b ranges over Σ. The definition follows.

$$c(S_1) = c(S_n) = 0.$$

$$c(S_{i,a,b}) = M_{a,b}(V_i^\#, W_i^\#), \text{ for } 1 < i < n.$$

$$c(I_i) = g(j),$$

where j is the length of the insertion block with address i.

$$c(D_i) = g(j),$$

where j is the length of the deletion block with address i.

A complete set of operations, CSO, is any set of operations which correspond to all blocks, one operation for each block. Hence each CSO has the same cardinality.

Since the cost of a substitution may depend on the context, it follows that when transforming $V^\#$ into $W^\#$ the order in which the operations are performed may influence the total cost of the transformation. We first define what it means to perform an operation on a string $X = x_1 \ldots x_n \in (\Sigma \cup \{-\})^*$.

- For $1 < i < n$, a substitution $S_{i,a,b}$ is *admissible* for X, if $x_{i-1} = a$ and $x_{i+1} = b$. The substitutions S_1 and S_n are always admissible. The result of performing the substitution (either $S_{i,a,b}$, or S_1, or S_n) on X is $X' = x_1 \ldots x_{i-1} W_i^\# x_{i+1} \ldots x_n$.
- I_i is always admissible for X and the resulting string is $X' = x_1 \ldots x_{i-1} W_i^\# \ldots W_{i+j-1}^\# x_{i+j} \ldots x_n$.
- D_i is always admissible for X and the resulting string is $X' = x_1 \ldots x_{i-1} \underbrace{- \ldots -}_{j} x_{i+j} \ldots x_n$.

Let $O = \{o_1, \ldots, o_k\}$ be a CSO. A linear order $o_1 < o_2 < \ldots < o_k$ is said to be *admissible*, if starting from V and performing the operations from O in the ascending order yields W without ever performing an inadmissible substitution. More formally, we define a sequence of strings $X_0, \ldots, X_k$ such that $X_0 = V$, $X_k = W$ and for every $1 \le i \le k$, the operation o_i is admissible for X_{i-1} and yields X_i. A CSO is called *admissible* if it has an admissible linear order. In general, an admissible CSO may have many admissible linear orders. The aim of this section is to characterize the structure of admissible linear orders on a given admissible CSO.

Before this, we give the definition of an *optimal contextual alignment*. Given two strings V, W, we maximize over all alignments $(V^\#, W^\#)$ and over all admissible CSO's O the cost

$$c(O) = \sum_{o \in O} c(o).$$

Hence the optimal solution consists not only of an alignment but also of a family of admissible linear orders for this alignment. We will see that in many situations this family of orders can be conveniently represented by *one* principal partial order P, all admissible linear orders being the linear extensions of P.

Consider the following example which illustrates the issues we have to deal with.

EXAMPLE 2. *Consider the following alignment.*

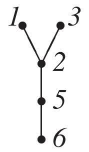
$$\begin{array}{cccccc} 1 & 2 & 3 & 4 & 5 & 6 \\ e & a & - & - & c & t \\ f & a & c & d & b & u \end{array}$$

Numbers in the above alignment represent positions. The upper string ea−−ct is a sample $V^\#$ and the lower string is a sample $W^\#$. The operations transform the upper string into the lower string.

The following set is a CSO

$$O_1 = \{S_1, S_{2,e,b}, I_3, S_{5,a,u}, S_6\}.$$

There are exactly two admissible linear orders for O_1: $6 \leq 5 \leq 2 \leq 3 \leq 1$ and $6 \leq 5 \leq 2 \leq 1 \leq 3$. These chains can represented as all linear extensions of the following principal poset.



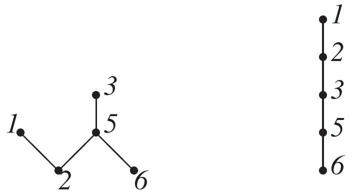
The two admissible linear orders are all (and only) linear extensions of the the above poset. In general the number of linear orders can be bigger and the structure of the principal poset can be more complicated.
Consider now the following CSO:

$$O_2 = \{S_1, S_{2,e,b}, I_3, S_{5,d,u}, S_6\}.$$

O_2 imposes the following constraints: 3 must be performed before 5, 5 must be performed before 2 and 2 must be performed before 3. Hence there is no admissible linear order and O_2 is inadmissible.
Finally let us consider the CSO:

$$O_3 = \{S_1, S_{2,e,c}, I_3, S_{5,a,u}, S_6\}.$$

The constraints generated by O_3 are: $6 \leq 5 \leq 3$ and $2 \leq 1$, plus the proviso that if 5 was performed before 2, then 3 has to be performed also before 2 (i.e. 2 cannot be between 5 and 3). It is easy to check that in this case there is no single principal poset generating all (and only) admissible linear orders. However, two generating posets can do the job.



Again, the representation works as follows. Every extension to a linear order of any of the above posets is an admissible linear order for O_3 and every admissible linear order is obtained in such a way. This example is little degenerated since the second generating poset is already a chain, so there is nothing to extend. In general, though, the generating posets can be more complicated and the number of such posets can be larger than 2.

Now let us consider a general situation of an insertion surrounded by two substitutions (we call it an *insertion block*). Constraints for a deletion block are obtained in a completely dual way.

$$\begin{array}{cccccccc} i & j & & & & & & k \\ a & - & - & \cdot & \cdot & \cdot & - & b \\ a' & c_1 & c_2 & \cdot & \cdot & \cdot & c_m & b' \end{array}$$

Fig. 1. A typical triple of blocks: substitution, insertion, substitution.

The second and third strings in the above Figure are assumed to be parts of $V^\#$ and $W^\#$, respectively. The numbers i, j, k stand for the addresses of the blocks. Clearly we have $j = i + 1$ and $k = i + m + 1$ but for the ease of presentation we choose to work with i, j, k as if they were independent. It is more convenient to examine the mutual constraints which come from choosing the right context for the left substitution *and* the left context for the right substitution.
The substitution i has three possible right contexts: b, b' and c_1. Likewise, the substitution k has three possible left contexts: a, a' and c_m. The case when $m = 0$, i.e. when there is no insertion between i and k will also be covered by our analysis.
Let us briefly discuss ways of representing families of linear orders on a three element set $\{i, j, k\}$. An explicit way is just to represent a given family by listing all of its elements. Another, more concise way is to represent a family by a finite poset whose all linear extensions form exactly the given family. Such a poset will be called a *principal poset*. Not every family of linear orders can be represented this way. It is easy to show that for a family of linear orders which has a principal poset, the intersection of all linear orders in that family yields this poset. We will use the following notation. $C_{x<y<z}$ stands for the constraint $x < y < z$. $C_{x<y}$ stands for the constraint $x < y$. It corresponds to the poset where the third element is not comparable to the other two. $\vee_x^{y,z}$ stands for the constraint $x < y, x < z$. $\wedge_{y,z}^x$ stands for the dual constraint $y < x, z < x$. $\perp$ stands for the contradictory constraint, i.e. it generates no linear order. It does not correspond to any poset. Finally $\top$ stands for the constraint which generates all linear orders (on the three element set). It corresponds to the discrete order on the three element set. The constraints are naturally ordered by comparing the sets of linear orders they generate. In fact, under this order the constraints, viewed as sets of linear orders on the three element set, form a Boolean algebra. If $\oplus$ stands for the least upper bound and $\otimes$ stands for the greatest lower bound, then we have the following sample identities:

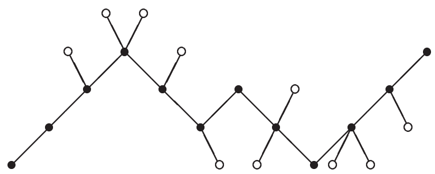
$$C_{i<j} \otimes C_{i<k} = \vee_i^{j,k},$$
$$\vee_i^{j,k} \oplus C_{k<i<j} = C_{i<j}.$$

Table 2 (Table 3, resp.) in Appendix lists three basic constraints for the three possible choices of the right context for i (left context for k, resp.).

If we view constraints as propositions, then $\oplus$ corresponds to disjunction, $\otimes$ corresponds to conjunction, and $\perp$ and $\top$ correspond to the truth values '*false*' and '*true*', respectively. This observation is useful when we want to generate from Tables 2 and 3 in Appendix the constraints which come from assuming simultaneous contexts: a right context for i and a left context for k. This corresponds to taking a conjunction of the constraints. For example, choosing (b, a) as a pair of contexts yields the constraint $\vee_i^{j,k} \otimes \vee_k^{i,j} = \perp$, while choosing (b, a') yields the constraint $\vee_i^{j,k} \otimes C_{i<k<j} = C_{i<k<j}$. The constraints for all possible pairs of contexts are listed in Chart 1 in Appendix.

So far, we have assumed that the symbols b, b', c_1 are pairwise different. Likewise for the symbols a, a', c_m. If, for example, the symbols b and b' happen to be equal[‡] in Figure 1, then the constraint associated with choosing b (which is the same as b') as the right context for i is obtained from Table 2 in Appendix by taking disjunction of the constraints which correspond to b and b'. Thus the constraint becomes $\vee_i^{j,k} \oplus C_{k<i<j}$. This constraint cannot be replaced by a constraint which describes a single poset. The above described situation of collapsing symbols b, b' corresponds to taking the partition $\{\{b, b'\}, \{c_1\}\}$ of the set $\{b, b', c_1\}$. Clearly there is a one-to-one correspondence between all possible equalities among the symbols b, b', c_1 and partitions of the set $\{b, b', c_1\}$. There are 5 possible partitions and therefore there are 25 possible tables, one for each pair of partitions on sets $\{b, b', c_1\}$ and $\{a, a', c_m\}$. Chart 1 in Appendix is just the table which corresponds to the finest partitions of $\{b, b', c_1\}$ and $\{a, a', c_m\}$. All the 25 tables, each containing all possible pairs of contexts, are listed in Appendix (Charts 1-25).

Finally let us discuss briefly a general shape of a poset which generates all (and only) admissible linear orders for a given alignment. The crucial notion here is that of a *hairy zig-zag*. Instead of giving a full definition (the reader may consult the full version of this paper) let us consider a typical example of a hairy zig-zag.



There is a one-to-one correspondance between operations associated with an alignment and the nodes of the hairy zig-zag. Disks in the above picture correspond to substitutions and to some insertions and deletions. They form a backbone of the zig-zag. Upper circles correspond to the remaining insertion blocks (they form upper *hairs*); while lower circles correspond to the remaining deletion blocks (they form lower hairs). A hairy zig-zag is just a zig-zag if it does not have any hairs. $p \leq q$ in the above poset means that the operation which corresponds to p has to be performed before the operation corresponding to q. Given an alignment $(V^{\#}, W^{\#})$ and an admissible complete set O of operations. A poset P is called a *generating poset* for $(V^{\#}, W^{\#})$ and O if all linear extensions of P are admissible for O. A *principal poset* P is a generating poset such that every admissible linear order for O is an extension of P. We have the following characterization of generating posets.

THEOREM 1. *Every generating poset is a disjoint union of hairy zig-zags.*
Moreover every hairy zig-zag can be obtained as a principal poset for a certain alignment and a certain CSO O.
For every alignment $(V^{\#}, W^{\#})$ without gaps, every admissible complete set of operations has a principal poset which is a disjoint union of zig-zags.

CONTEXTUAL ALIGNMENT ALGORITHMS

Below we present a *quadratic* (precisely, of $O(|\Sigma|mn)$ time complexity) algorithm for contextual alignment and an affine gap penalty function. The efficiency of the algorithm is based on the observation that for an affine gap penalty function we can compute the score of an indel, by gradually extending the indel one symbol at a time. This way, when extending the alignment, it suffices to consider a constant number of cases. Let $Open$ and Ext be the costs of opening and extending the gap, respectively.

The main idea of the algorithm is to walk along both sequences, choosing the optimal extension of the already found alignment of prefixes. At each step we have the following possible extensions: (i) a single substitution, (ii) an insertion (starting a new one or extending the existing one), (iii) a deletion, analogously.

The algorithm uses the dynamic programming approach and works with 7 three-dimensional arrays indexed by the positions from the word V, positions from the word W and the elements of Σ. These are 3 *insertion arrays*, 3 *deletion arrays* and one *substitution array*.

Let us consider a type (ii) extension, i.e. we focus our attention on a single insertion block:

$$\begin{array}{ccccccc} i & j & & & & & k \\ a & - & - & \cdot & \cdot & \cdot & - & b \\ a' & c_1 & c_2 & \cdot & \cdot & \cdot & c_m & \boldsymbol{x} \\ \alpha & & & & & & \beta & \end{array}$$

‡ Here we implicitly assume that c_1 is different from b and b'.

For the simplicity of presentation we use, as before, numbers i, j, k as the addresses of the blocks, i.e. positions in the alignment. The index $\alpha = 1 \ldots n$ is used to number the positions in the word V, and $\beta = 1 \ldots m$ for the positions in the word W. The letter $\boldsymbol{x}$ in the above figure stands for an arbitrary letter from Σ, which replaces W_k. Considering all $\boldsymbol{x} \in \Sigma$ is necessary to profit from the affinity of gap penalty function and to extend an insertion gradually in the course of computation. In fact $\boldsymbol{x}$ corresponds to the third dimension in our insertion arrays. Roughly, this third dimension enables us to treat left substitution i and right substitution k separately.

Each insertion array stores at position $(\alpha, \beta, \boldsymbol{x})$ the maximal score of alignment of $V_1 \ldots V_\alpha$ and $W_1 \ldots W_\beta$ which ends with an insertion, under the assumption that $W_{\beta+1} = \boldsymbol{x}$ and that the insertion is immediately followed by a substitution on the right (substitution k in the above figure). We use three variants of the insertion array, one for each possible selection of the left context for the right substitution k (see Table 3 in Appendix).

- $I_{\vee_k^{i,j}}(\alpha, \beta, \boldsymbol{x})$ stores the maximal score as explained above under the additional condition that a is the left context for the right substitution k. This selection results in the constraint $\vee_k^{i,j}$, i.e. the rightmost substitution k precedes insertion j and left substitution i.
- $I_{C_{i<k<j}}(\alpha, \beta, \boldsymbol{x})$ stores the maximal score as explained above under the additional condition that a' is the left context for the right substitution k. This selection results in the constraint $C_{i<k<j}$, i.e. the left substitution is followed by the insertion which is followed the right substitution.
- $I_{C_{j<k}}(\alpha, \beta, \boldsymbol{x})$ stores the maximal score as explained above under the additional condition that c_m is the left context for the right substitution k. This selection results in the constraint $C_{j<k}$, i.e. the rightmost substitution k is performed after the insertion j.

Note that parameter $\boldsymbol{x}$ together with the subscript of I contain the information which is necessary and sufficient to correctly evaluate an alignment ending with an insertion. The arrays $D_{\wedge_{i,j}^k}, D_{C_{k<j}}, D_{C_{j<k<i}}$ built for a deletion block are obtained in a completely dual way.

For the case of the extension by a single substitution we have the following array:

- $S(\alpha, \beta, \boldsymbol{x})$ stores the maximal score of an alignment with gaps of words $V_1 \ldots V_\alpha$ and $W_1 \ldots W_\beta$ which ends with a substitution $V_\alpha \mapsto W_\beta$ whose right context is $\boldsymbol{x}$.

Since our algorithm requires always left and right context for each considered substitution, we need to extend arrays V and W by the new entries V_0, V_{n+1} and W_0, W_{m+1} on the ends. Recall that the leftmost and rightmost substitutions do not contribute to the score of the alignment, they serve only as contexts for the neighboring operations. Hence the choice of the new entries has a negligible effect on the resulting score — we decided to choose this values in an ad-hoc manner. In the particular implementation of our algorithm we choose methionine $M = V_0 = W_0$ as the leftmost entry and alanine $A = V_{n+1} = W_{m+1}$ as the rightmost entry.

In the course of computation there are two possibilities to update insertion arrays: (i) starting a new insertion; (ii) extending the existing insertion. The latter case is simpler: the score of extended alignment is calculated as follows:

$$I_*(\alpha, \beta - 1, \boldsymbol{x}) + Ext \quad \text{for } * \in \{\vee_k^{i,j}, C_{i<k<j}, C_{j<k}\}$$

When starting a new insertion we have to calculate the score of the substitution which precedes it. To achieve this aim we consider all possible selections of the right context for this substitution. The corresponding scores and constraints are summarized in Table 1.

right context	constraint	score of extended alignment
b	$\vee_i^{j,k}$	$S(\alpha, \beta - 1, b) + Open$
$\boldsymbol{x}$	$C_{k<i<j}$	$S(\alpha, \beta - 1, \boldsymbol{x}) + Open$
c_1	$C_{j<i}$	$S(i, j' - 1, c_1) + Open$

Table 1. Constrains and scores determined by right context for i.

As we are looking for the optimal alignment of prefixes $V_1 \ldots V_\alpha$ and $W_1 \ldots W_\beta$, we choose the context pair (i.e. the right context for the left substitution and the left context for the right substitution) which does not generate the contradiction and which maximizes the score. The constraints resulting from all possible pairs of contexts for insertion arrays are listed in Chart 1 of Appendix (with $\boldsymbol{x}$ substituted for b').

E.g. for the left context a there are two admissible pairs $(\boldsymbol{x}, a)$ and (c_1, a) and we should compare the corresponding scores $S(\alpha, \beta-1, \boldsymbol{x}) + Open$ and $S(\alpha, \beta-1, c_1) + Open$ with the score of extended insertion $I_{\vee_k^{i,j}}(\alpha, \beta - 1, \boldsymbol{x}) + Ext$ and choose the maximal value.

The update rules for three insertion arrays are as follows:

$$I_{\vee_k^{i,j}}(\alpha,\beta,\boldsymbol{x}) := \max \begin{cases} I_{\vee_k^{i,j}}(\alpha,\beta-1,\boldsymbol{x}) + Ext \\ S(\alpha,\beta-1,\boldsymbol{x}) + Open, \\ S(\alpha,\beta-1,c_1) + Open \end{cases}$$

$$I_{C_{i<k<j}}(\alpha,\beta,\boldsymbol{x}) := \max \begin{cases} I_{C_{i<k<j}}(\alpha,\beta-1,\boldsymbol{x}) + Ext, \\ S(\alpha,\beta-1,b) + Open \end{cases}$$

$$I_{C_{j<k}}(\alpha,\beta,\boldsymbol{x}) := \max \begin{cases} I_{C_{j<k}}(\alpha,\beta-1,\boldsymbol{x}) + Ext, \\ S(\alpha,\beta-1,b) + Open, \\ S(\alpha,\beta-1,c_1) + Open \end{cases}$$

To calculate a new entry in the substitution table, say $S(\alpha,\beta,\boldsymbol{x})$, we choose the maximal value among the following 8 possibilities:

- 2 cases for the scenario when the substitution $W_\alpha \mapsto W_\beta$ follows another substitution:
  - left substitution precedes the right one: $S(\alpha-1,\beta-1,V_\alpha) + M_{W_{\beta-1},\boldsymbol{x}}(V_\alpha,W_\beta)$
  - right substitution precedes the left one: $S(\alpha-1,\beta-1,W_\beta) + M_{V_{\alpha-1},\boldsymbol{x}}(V_\alpha,W_\beta)$
- 3 cases for the scenario when the substitution follows an insertion.

  $$I_*(\alpha-1,\beta-1,W_\beta) + M_{l_*,\boldsymbol{x}}(V_\alpha,W_\beta)$$

  $$\text{for } * \in \{\vee_k^{i,j}, C_{i<k<j}, C_{j<k}\},$$

  where l_* denotes the corresponding left context in each case. This context is stored together with the array I_*.
- 3 cases for the scenario when the substitution follows a deletion.

  $$D_\bullet(\alpha-1,\beta-1,V_\alpha) + M_{l_\bullet,\boldsymbol{x}}(V_\alpha,W_\beta)$$

  $$\text{for } \bullet \in \{\wedge_{i,k}^{j}, C_{j<k<i}, C_{k<j}\},$$

  where $l_\bullet$ denotes the corresponding left context in each case.

The generating posets are reconstructed in the following way. Starting from the maximal position in substitution array, and backtracking in all 7 arrays determines the set of operations performed to achieve the maximal score, and the corresponding alignment. By comparing the contexts we also determine the set of generating posets. Each step of the algorithm, in which the substitution is considered corresponds to a small (two- or three-element) poset, and by concatenating these posets in reversed order, we reconstruct the generating poset. Each small poset is determined (tables from Appendix are used here) by the partition of the set of symbols which serve as the contexts for the corresponding pair of substitutions.

Algorithm 1 the overview

{ *initialization* }
for $\boldsymbol{x} \in \Sigma$ **do** $S(0,0,\boldsymbol{x}) := 0;$
for $\boldsymbol{x} \in \Sigma$, $\alpha > 0, \beta > 0$ **do** $S(\alpha,0,\boldsymbol{x}) := -\infty;$
$S(0,\beta,\boldsymbol{x}) := -\infty;$
for $\boldsymbol{x} \in \Sigma$, $\alpha = 0 \dots n$, $* \in \{\vee_k^{i,j}, C_{i<k<j}, C_{j<k}\}$ **do**
$I_*(\alpha,0,\boldsymbol{x}) := -\infty;$
for $\boldsymbol{x} \in \Sigma$, $\beta = 0 \dots m$, $\bullet \in \{\wedge_{j,k}^{i}, C_{j<k<i}, C_{k<j}\}$ **do**
$D_\bullet(0,\beta,\boldsymbol{x}) := -\infty;$
{ *end of initialization* }
for $\alpha = 1 \dots n$ **do**
  for $\beta := 1 \dots m$ **do**
    for $\boldsymbol{x} \in \Sigma$ **do**
      calculate the 7 arrays at position $(\alpha,\beta,\boldsymbol{x})$.
    end for
  end for
end for

$$\textbf{Score} := \max \begin{cases} S(n,m,V_{n+1}), S(n,m,W_{m+1}), \\ I_*(n,m,W_{m+1}), D_\bullet(n,m,V_{n+1}) \\ * \in \{\vee_k^{i,j}, C_{i<k<j}, C_{j<k}\} \\ \bullet \in \{\wedge_{j,k}^{i}, C_{j<k<i}, C_{k<j}\} \end{cases}$$

Algorithms for local alignment. As with the context-free case (see Smith *et al.* (1981)), the algorithm for local alignment is in each case a minor modification of the global alignment algorithm. The idea is to add an additional option while filling in the arrays: to give up with the so far constructed alignment and start all over in the middle of the sequences by resetting the score to 0. The local versions of the above algorithms inherit the complexity of their global counterparts.

EXPERIMENTAL RESULTS

We have performed elementary validation of the contextual alignment model. We have found some interesting phenomena, suggesting that further work in this direction is worthwhile.

Substitution tables The first, preparatory (but by no means trivial) task has been the construction of appropriate contextual substitution tables. We have developed a procedure to construct such tables, which is described in Gambin *et al.* (2002b). It appears that there is a fundamental difficulty: the amount of data necessary to construct complete contextual substitution tables exceeds by an order of magnitude the data presently available. Indeed, the tables have $20^4 = 160\,000$ entries, and each entry should be calculated based on a sufficient statistical sample. It appears that while for the most common contexts the sample is rich enough, for the less common ones it is insufficient and the values are statistically biased, and about one third of the entries remain completely undetermined. In order to remedy the situation we have grouped the amino acids into 6 groups, based on accepted point mutation data (Dayhoff *et al.*, 1978).

GROUP NAME	AMINO ACID RESIDUE
Small Aliphatic	Alanine, Proline, Glycine
Acid amide	Glutamine, Aspargine, Glutamic A., Aspartic A.
Hydroxyl & Sulfhydryl	Serine,Threonine, Cysteine
Aliphatic	Valine, Isoleucine, Methionine, Leucine
Basic	Lysine, Arginine, Histidine
Aromatic	Phenylalanine, Tyrosine, Tryptophan

The molecular sizes and shapes are very similar within each group. This is a crucial factor in determining which amino acid interchanges are acceptable to natural selection. Then we have calculated the substitution tables for the groups rather than individual amino acids, assuming that all amino acids in one group behave identically as contexts. The values taken from such group-context tables have been used to fill in the missing values in the full-context substitution tables.

While interpreting our experimental results, it is important to remember about the limitations of our tables.

Experiments The experiments reported here are based purely on score values. We decided to assess mainly the influence of the context on the score value. We achieved this by producing four contextual tables, for increasing number of context groups. We used here the same approach as in the procedure of completing the tables by 6 groups values, described above. The first table assumed all amino acids to be identical, hence the table is indeed context-free, the second table is based on two context groups (of hydrophobic and hydrophilic amino acids), the third one is based on six context groups, and the fourth one used all 20 amino acids as contexts (but, as explained above, about one third of the entries had to be taken from the six-group tables). Then we aligned several proteins each with each, using all four tables. To estimate the statistical significance of our alignments we calculated Z-value (Comet *et al.*, 1999) in the case of global alignment and we adopted the method of Vingron and Waterman (Vingron *et al.*, 1995) of calculating P-value for local alignments.

In the next phase of model validation we plan to build the phylogenetic trees based on contextual similarity data. Then we will compare two sets of trees (contextual vs. non-contextual) using various agreement methods. The complete experiments illustrating the impact of the contextual approach for phylogenetic studies will be described in the forthcoming paper.

We decided to use the database of Clusters of Orthologous Groups of proteins (Tatusov *et al.*, 2001) (see also the NIH COG page http://www.ncbi.nlm.nih.gov/COG). It consists currently of 3307 COGs including 74059 proteins from 43 genomes of bacteria, archaea and the yeast *Saccharomyces cerevisiae*. COG database represents an attempt of a phylogenetic classification of the proteins encoded in complete genomes. Each COG includes proteins that are thought to be orthologous, i.e. connected by vertical evolutionary descent.

We restrict our attention to the list of 84 COGs, which contain at least one protein from each genome. From this list 27 COGs (which include as few paralogs as possible) are selected to our analysis. They can be partitioned into 2 groups; the first one consists of 12 COGs, which represents a wide spectrum of functional categories, and the second consists of 15 *t-RNA synthetases* families. The sequences from each COG are pairwise aligned globally and locally and the statistical significance (Z-value and P-value, respectively) are computed.

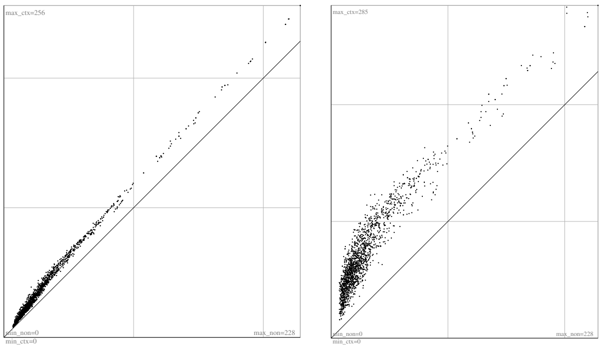
This experiment had the following goals.

1. Estimation of the significance of contextual alignment vs. non-contextual one.
2. Observation of the impact the number of context groups has on the discriminating power of contextual approach.
3. Verification of the accuracy of contextual scores w.r.t. the classification of proteins inside COGs and evolutionary relationships between proteins.

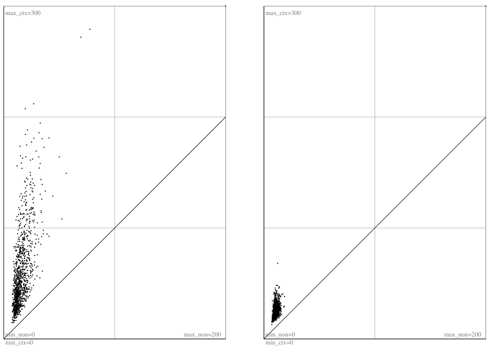
The majority of aligned pairs of proteins gives better statistical significance measures for contextual approach. A typical outcome is listed below: + denotes that the calculated P-value was 2 orders of magnitude smaller for contextual alignment when compared with context-free one. The table below contains the results for locally aligned proteins from COG00030. The names of aligned proteins are listed in the first column.

AF1783			+	+	+	+		+	+	+	+	+		+	+		+	+		+	+	+		+	+	+			+
APE0553								+	+			+				+		+		+				+	+		+	+	
aq_1816	+	+		+	+	+	+	+	+	+	+	+	+	+		+	+	+	+	+	+	+	+	+	+	+	+	+	+
BB0590	+	+	+		+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+
BH0057	+	+	+	+			+	+	+	+	+	+	+	+	+	+		+	+	+	+	+	+	+	+	+	+	+	+
BS_ksgA	+	+	+	+			+	+	+	+	+	+	+	+	+	+		+	+	+	+	+	+	+	+	+	+		+
BU141	+	+	+	+	+	+		+	+	+	+	+		+	+		+	+	+	+	+	+	+			+	+	+	
CC1685	+	+	+	+	+	+	+		+	+	+	+	+	+	+		+	+	+	+		+	+	+	+	+	+	+	+
Cj1711c	+	+	+	+	+	+	+	+		+	+	+	+			+	+	+	+	+		+	+	+	+	+	+	+	+
CPn1059	+	+	+	+	+	+	+	+	+			+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+
CT354	+		+	+	+	+	+	+	+			+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+
DR1526	+	+	+	+	+	+	+	+	+	+	+		+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+
HI0549	+	+	+	+	+	+			+	+	+	+		+	+		+	+	+	+	+	+	+				+	+	
HP1431	+		+	+	+	+	+	+	+	+	+	+	+			+	+	+		+				+	+	+	+	+	+
jhp1322	+		+	+	+	+	+	+	+	+	+	+	+			+	+	+		+				+	+	+	+	+	+
ksgA	+	+	+	+	+	+			+	+	+	+		+	+		+	+	+	+		+	+				+	+	
L0363	+	+	+	+			+	+	+	+	+	+	+	+	+	+		+	+	+	+	+	+	+	+	+	+	+	+
MG463	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+		+	+	+		+	+	+	+	+	+	+
MJ1029	+		+	+	+	+	+	+	+	+	+	+	+		+	+	+	+		+	+	+		+	+	+			+
ML0241	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+		+	+	+	+	+	+	+	+	+
mll7860	+		+	+	+	+	+			+	+	+	+				+	+	+	+			+	+	+	+	+	+	+
MPN679	+	+	+	+	+	+	+	+	+	+	+	+	+			+	+		+	+	+		+	+	+	+	+	+	+
MTH1326		+	+	+	+	+	+	+	+	+		+	+		+	+	+			+	+	+		+	+	+			+
NMA0902	+	+	+	+	+	+	+	+	+	+	+	+		+	+		+	+	+	+	+	+	+				+	+	
NMB0697	+	+	+	+	+	+	+	+	+	+	+	+		+	+		+	+	+	+	+	+	+				+	+	
PA0592	+	+	+	+	+	+			+	+	+	+		+	+		+	+	+	+	+	+	+				+	+	
PAB0253		+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+		+	+	+		+	+	+			+
PH1823		+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+		+	+	+		+	+	+			+
PM1209	+	+	+	+	+	+			+	+	+	+		+	+		+	+	+	+	+	+	+				+	+	

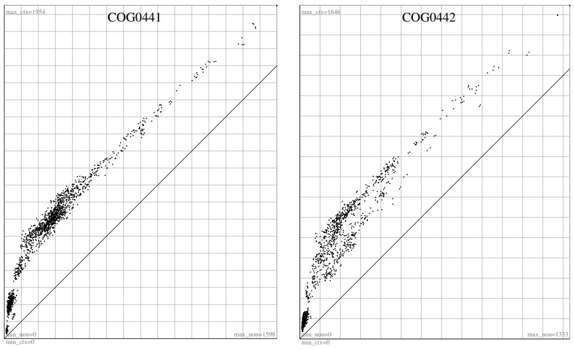
Now, let us observe that the discriminating power indeed increases with the number of context groups. It is particularly visible in the area of low scores (close to the bottom left corner of the plot), where the vertical cut through the area occupied by the alignments' scores is particularly large.The plots below have the context-free score as the x-axis, and the y-axis is, from left to right, the local alignment score with 2 and 20 context groups, respectively. The aligned sequences are all proteins from COG0089 (Ribosomal proteins—large subunit A). The grid on the plots marks multiples of 100.

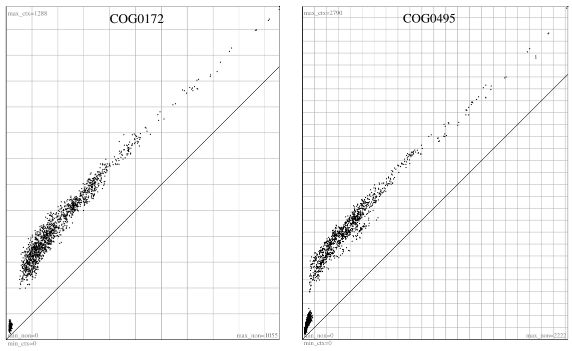


This influence of context should also be verified w.r.t. unrelated sequences. Two plots below illustrate the result: on the left 1176 pairs of distantly related homologous pairs (i.e. having less than 25% sequence identity) are aligned, and on the right plot 1176 pairs of completely unrelated proteins are aligned. All sequences are selected from COG database. As before plots have the context-free score as the x-axis, and the contextual score as the y-axis. It is clearly visible that the discriminating power of contextual approach decreases, when less similar sequences are investigated.



An interesting phenomenon is observed for 4 RNA synthetase families, namely for COGs 172, 441, 442, and 495. On the plot for all pairwise local comparisons of sequences from each of these 4 COGs, two separated groups of points are clearly visible. This indicates that the contextual algorithm subdivides the alignments into subsets, while the non-contextual alignment score alone does not distinguish these groups.





A detailed analysis leads to an interesting observation that in each case, the subdivision of scores gives rise to a partition of proteins into two subsets. The group with lower contextual scores contains precisely scores of pairs from different subsets, while the group of higher contextual scores contains scores of pairs from the same subset. It always turns out that one of the subsets is much smaller than the other. Moreover, in COG0441 one can distinguish even 3 groups. In that case, the group with the smallest scores corresponds to alignments of protein APE0117 with all others. A promising fact is that our partitions are in agreement with known groups of evolutionary close proteins inside COGs. This can be verified by comparison with the existing phylogenetic trees. This readily confirms the accuracy of the contextual approach. In the table below we summarize the cardinalities of the smaller subset for each of COGs considered:

COG	number of proteins
COG0172	2 proteins (MJ1077, MTH1122)
COG0441	6 (APE0117 and the middle class of 5 proteins)
COG0442	16 proteins
COG0495	9 proteins

The role of order The model of contextual alignment allows us to determine not only the maximal score of an alignment of two proteins, but also the partial order among the substitutions and indels, which yields that maximal score.

The role of the order in determining the statistical significance of a score remains unknown. It might be possible that an alignment of marginal score has statistically quite unusual shape of the order of operations, suggesting a kind of distant homology between the two aligned proteins – despite low score.

However, part of the work reported in this paper has been done in preparation for this research. The precise characterization of possible principal posets of contextual alignments, given in Theorem 1, provides two crucial tools for that:

- It gives the possibility of developing specific, very efficient algorithms for dealing with principal posets, which take advantage of the *a priori* known shape of their arguments.

- Gives the possibility of determining what is the statistically typical shape of a principal poset.

CONCLUSIONS AND OPEN PROBLEMS

We would like to remark that there is no need to restrict the context to consist of one letter to the left and one letter to the right, only. Indeed, our algorithms extend without much difficulty to the case with contexts consisting of k letters on each side, for a fixed k. The algorithms are still of $O(f(|\Sigma|, k) \cdot mn)$ complexity. However, the factor $f(|\Sigma|, k)$ grows extremely fast with k, which makes algorithms even for $k = 2$ impractically slow. An interesting observation is that the local sequence around each amino acid can be used to approximate the "3 D context" i.e. the local environment in secondary and tertiary structure of the protein. For this aim the context consisting only of one pair of letters is sufficient. For example, in β-sheets only amino acids at positions $i \pm 2$ contact the ith one (Schulz *et al.*, 1979). It would be interesting to adopt our methodology to deal with this situation.

An additional problem, mentioned already above, is that already for $k = 1$ there is hardly enough biological data available to construct the substitution tables. For $k = 2$ the situation becomes completely hopeless. Therefore, based on purely pragmatic reasons, we decided to limit our presentation to the case $k = 1$.

Another possible extension is to permit the scores of insertions and deletions to depend on the context, too. It is again well motivated from the biological point of view (Stryer , 1995). To conform to this more general situation, the overall idea of the algorithms need not be changed, but the details become more complicated.

ACKNOWLEDGEMENTS

We thank Piotr Slonimski for many fruitful discussions and comments. We also thank Maricel Kann for providing us with some data for testing contextual alignment. This work was done during the postdoctoral fellowship of A. Gambin from the CNRS within the framework of the Centre Franco-Polonais de Biotechnologie des Plantes. A. Gambin was also supported by the NATO science fellowship, and S. Lasota by the fellowship of Foundation for Polish Science.

REFERENCES

Comet,J.P., Aude,J.C., Glemet, E., Henaut,J.L., Slonimski,P.P. and Codani,J.J (1999) Significance of Z-value statistics of Smith-Waterman scores for protein alignments. *Comput. Chemistry*, **23**, 317–331.

Durbin,R., Eddy,S.R., Krogh, A. and Mitchison,G. (1998) Biological sequence analysis. Probabilistic models of proteins and nucleid acids. *Cambridge University Press*.

Dayhoff,M.O., Schwartz,R.M. and Orcutt,B.C. (1978) A model of evolutionary change in proteins. *In M.O. Dayhoff, editor,* Atlas of Protein Sequence and Structure, **5**, 345–352.

Gusfield,D. (1997) Algorithms on Strings, Trees and Sequences: Computer Science and Computational Biology. *Cambridge University Press*.

Gambin,A., Lasota,S., Szklarczyk,R., Tiuryn,J. and Tyszkiewicz,J. (2002a) Contextual alignment of biological sequences. *Currents in Computational Molecular Biology. Posters from RECOMB 2002*.

Gambin,A. and Tyszkiewicz,J. (2002b) Substitution matrices for contextual alignment. *Journees Ouvertes Biologie Informatique Mathematique (JOBIM 2002), J. Nicolas and C. Thermes, (eds.)*, 227–238.

Henikoff,S. and Henikoff,J.G. (1992) Amino acid substitution matrices from protein blocks. *Proc. Natl. Acad. Sci. USA*, **89**, 10915–10919.

Ioerger,T.R. (1997) The context-dependence of amino acid properties. *Intelligent Systems in Molecular Biology, AAAI Press*, 157–166.

Jensen,J.L. and Pedersen,A.M.K (2000) Probabilistic models of DNA sequence evolution with context dependent rates of substitution. *Advences in Applied Probability*, **32**, 499–517.

Schulz,G. and Schirmer,R. (1979) Principles of Protein Structure. *Springer Verlag, New York*.

Stryer,L. (1995) Biochemistry. *W.H. Freeman and Company*.

Schöniger,M. and von Haeseler,A. (1994) A stochastic model for the evolution of autocorrelated DNA sequences. *Molecular Phylogenetics and Evolution*, **3**, 240–247.

Smith,T.F. and Waterman,M.S. (1981) The identification of common molecular subsequences. *Journal of Molecular Biology*, **147**, 195–197.

Tavaré,S. and Giddings,B.W. (1989) Some statistical aspects of the primary structure of nucleotide sequences. *In M.S. Waterman, editor, Mathematical methods for DNA sequences*, 117–132.

von Haeseler,A. and Schöniger,M. (1998) Evolution of DNA or amino acid sequences with dependent sites. *Journal of Computational Biology*, **5**, 149–163.

Tatusov,R.L., Natale,D.A., Garkavtsev,I.V., Tatusova,T.A., Shankavaram,U.T., Rao,B.S., Kiryutin,B., Galperin,M.Y., Fedorova,N.D. and Koonin,E.V. (2001) The COG database: new developments in phylogenetic classification of proteins from complete genomes. *Nucleic Acids Res.*, **29(1)**, 22–28.

Vingron,M. and Waterman,M.S. (1995) Statistical significance of local alignments with gaps. *Bioinformatics: from Nucleid Acids and Proteins to Cell Methabolism*, 75–84.

Wilbur,W.J. and Lipman,D.J. (1984) The context dependent comparison of biological sequences. *SIAM J. Appl. Math.*, **44(3)**, 557–567.

APPENDIX

Constraint Tables for an Insertion Block

For the tables in this Appendix we use the following notation (same as in Figure 1).

$$\begin{array}{cccccccc} i & j & & & & & & k \\ a & - & - & \cdot & \cdot & \cdot & - & b \\ a' & c_1 & c_2 & \cdot & \cdot & \cdot & c_m & b' \end{array}$$

First we give for right contexts for i and separately for left contexts for k.

b	b'	c_1
$\vee_i^{j,k}$	$C_{k<i<j}$	$C_{j<i}$

Table 2. Constraints for right contexts for i.

a	a'	c_m
$\vee_k^{i,j}$	$C_{i<k<j}$	$C_{j<k}$

Table 3. Constraints for left contexts for k.

Next we give the constraint tables for all possible relationships between the characters in the alignment (all equalities are explicitly mentioned for each table).

(b,a)	(b,a')	(b,c_m)	(b',a)	(b',a')	(b',c_m)	(c_1,a)	(c_1,a')	(c_1,c_m)
$\perp$	$C_{i<k<j}$	$C_{i<j<k}$	$C_{k<i<j}$	$\perp$	$\perp$	$C_{k<j<i}$	$\perp$	$\vee_j^{i,k}$

Chart 1: Constraints for all possible pairs of contexts (no equalities assumed).

(b,a)	(b,a')	(b,c_m)	(b',a)	(b',a')	(b',c_m)
$C_{k<j<i}$	$C_{i<k<j}$	$C_{j<k}$	$C_{k<i<j}$	$\perp$	$\perp$

Chart 2: Constraints for the situation $b = c_1$.

(b,a)	(b,a')	(b,c_m)	(b',a)	(b',a')	(b',c_m)
$\perp$	$C_{i<k<j}$	$C_{i<j<k}$	$\vee_k^{i,j}$	$\perp$	$\vee_j^{i,k}$

Chart 3: Constraints for the situation $b' = c_1$.

(b,a)	(b,a')	(b,c_m)	(c_1,a)	(c_1,a')	(c_1,c_m)
$C_{k<i<j}$	$C_{i<k<j}$	$C_{i<j<k}$	$C_{k<j<i}$	$\perp$	$\vee_j^{i,k}$

Chart 4: Constraints for the situation $b = b'$.

(b,a)	(b,a')	(b,c_m)
$\vee_k^{i,j}$	$C_{i<k<j}$	$C_{j<k}$

Chart 5: Constraints for the situation $b = b' = c_1$.

(b,a)	(b,a')	(b',a)	(b',a')	(c_1,a)	(c_1,a')
$C_{i<j<k}$	$C_{i<k<j}$	$C_{k<i<j}$	$\perp$	$C_{j<i}$	$\perp$

Chart 6: Constraints for the situation $a = c_m$.

(b,a)	(b,a')	(b',a)	(b',a')	(c_1,a)	(c_1,a')
$\perp$	$\vee_i^{k,j}$	$C_{k<i<j}$	$\perp$	$C_{k<j<i}$	$\vee_j^{i,k}$

Chart 7: Constraints for the situation $a' = c_m$.

(b,a)	(b,c_m)	(b',a)	(b',c_m)	(c_1,a)	(c_1,c_m)
$C_{i<k<j}$	$C_{i<j<k}$	$C_{k<i<j}$	$\perp$	$C_{k<j<i}$	$\vee_j^{i,k}$

Chart 8: Constraints for the situation $a = a'$.

(b,a)	(b',a)	(c_1,a)
$\vee_i^{k,j}$	$C_{k<i<j}$	$C_{j<i}$

Chart 9: Constraints for the situation $a = a' = c_m$.

(b,a)	(b,a')	(b',a)	(b',a')
$C_{i<j<k} \oplus C_{j<i}$	$C_{i<k<j}$	$C_{k<i<j}$	$\perp$

Chart 10: Constraints for the situation $b = c_1$ and $a = c_m$.

(b,a)	(b,a')	(b',a)	(b',a')
$C_{k<j<i}$	$\vee_i^{j,k} \oplus \vee_j^{i,k}$	$C_{k<i<j}$	$\perp$

Chart 11: Constraints for the situation $b = c_1$ and $a' = c_m$.

(b,a)	(b,c_m)	(b',a)	(b',c_m)
$C_{k<j<i} \oplus C_{i<k<j}$	$C_{j<k}$	$C_{k<i<j}$	$\bot$

Chart 12: Constraints for the situation $b = c_1$ and $a = a'$.

(b,a)	(b',a)
$\vee_i^{j,k} \oplus C_{j<i}$	$C_{k<i<j}$

Chart 13: Constraints for the situation $b = c_1$ and $a = a' = c_m$.

(b,a)	(b,a')	(b',a)	(b',a')
$C_{i<j<k}$	$C_{i<k<j}$	$\vee_k^{i,j} \oplus \vee_j^{i,k}$	$\bot$

Chart 14: Constraints for the situation $b' = c_1$ and $a = c_m$.

(b,a)	(b,a')	(b',a)	(b',a')
$\bot$	$\vee_i^{j,k}$	$\vee_k^{i,j}$	$\vee_j^{i,k}$

Chart 15: Constraints for the situation $b' = c_1$ and $a' = c_m$.

(b,a)	(b,c_m)	(b',a)	(b',c_m)
$C_{i<k<j}$	$C_{i<j<k}$	$\vee_k^{i,j}$	$\vee_j^{i,k}$

Chart 16: Constraints for the situation $b' = c_1$ and $a = a'$.

(b,a)	(b',a)
$\vee_i^{j,k}$	$\vee_j^{i,k} \oplus \vee_k^{i,j}$

Chart 17: Constraints for the situation $b' = c_1$ and $a = a' = c_m$.

(b,a)	(b,a')	(c_1,a)	(c_1,a')
$C_{k<i<j} \oplus C_{i<j<k}$	$C_{i<k<j}$	$C_{j<i}$	$\bot$

Chart 18: Constraints for the situation $b = b'$ and $a = c_m$.

(b,a)	(b,a')	(c_1,a)	(c_1,a')
$C_{k<i<j}$	$\vee_i^{j,k}$	$C_{k<j<i}$	$\vee_j^{i,k}$

Chart 19: Constraints for the situation $b = b'$ and $a' = c_m$.

(b,a)	(b,c_m)	(c_1,a)	(c_1,c_m)
$\wedge_{k,i}^{j}$	$C_{i<j<k}$	$C_{k<j<i}$	$\vee_j^{i,k}$

Chart 20: Constraints for the situation $b = b'$ and $a = a'$.

(b,a)	(c_1,a)
$C_{i<j}$	$C_{j<i}$

Chart 21: Constraints for the situation $b = b'$ and $a = a' = c_m$.

(b,a)	(b,a')
$\vee_k^{i,j} \oplus C_{j<k}$	$C_{i<k<j}$

Chart 22: Constraints for the situation $b = b' = c_1$ and $a = c_m$.

(b,a)	(b,a')
$\vee_k^{i,j}$	$\vee_i^{j,k} \oplus \vee_j^{i,k}$

Chart 23: Constraints for the situation $b = b' = c_1$ and $a' = c_m$.

(b,a)	(b,c_m)
$C_{k<j}$	$C_{j<k}$

Chart 24: Constraints for the situation $b = b' = c_1$ and $a = a'$.

(b,a)
$\top$

Chart 25: Constraint for the situation $b = b' = c_1$ and $a = a' = c_m$.