On Suboptimal Alignments of Biological Sequences

Dalit Naor * and Douglas Brutlag

Department of Biochemistry
Stanford University Medical Center
Stanford, California 94305-5307

Abstract. It is widely accepted that the optimal alignment between a pair of proteins or nucleic acid sequences that *minimizes the edit distance* may not necessarily reflect the correct *biological* alignment. Alignments of proteins based on their structures or of DNA sequences based on evolutionary changes are often different from alignments that minimize edit distance. However, in many cases (e.g. when the sequences are close), the edit distance alignment is a good approximation to the biological one. Since, for most sequences, the true alignment is unknown, a method that either assesses the significance of the optimal alignment, or that provides few "close" alternatives to the optimal one, is of great importance.

A suboptimal alignment is an alignment whose score lies within the *neighborhood* of the optimal score. Enumeration of suboptimal alignments [Wa83, WaBy] is not very practical since there are many such alignments. Other approaches [Zuk, Vi, ViAr] that use only partial information about suboptimal alignments are more successful in practice.

We present a method for representing all alignments whose score is within any given delta from the optimal score. It represents a large number of alignments by a compact graph which makes it easy to impose additional biological constraints and select one desirable alignment from this large set. We study the combinatorial nature of suboptimal alignments. We define a set of "canonical" suboptimal alignments, and argue that these are the essential ones since any other suboptimal alignment is a combination of few canonical ones. We then show how to efficiently enumerate suboptimal alignments in order of their score, and count their numbers. Examples are presented to motivate the problem.

Since alignments are essentially (s,t)-paths in a directed acyclic graph with (possibly negative) weights on its edges, our solution gives an extremely simple method to enumerate all K shortest (or longest) paths from s to t in such graphs in increasing order, as well as all (s,t) paths that are within δ of the optimum, for any δ. We compare this solution with known algorithms that find the K-best shortest paths in a graph.

* Supported by a Postdoctoral Fellowship from the Program in Mathematics and Molecular Biology of the University of California at Berkeley, under National Science Foundation Grant DMS-9720208

1 Introduction

1.1 Motivation

Protein and Nucleic Acid sequences are regularly compared agains one another, since it is assumed that sequences with similar biological function have similar physical characteristics, and therefore are similar, or homologous, at the amino or nucleic acid sequence level [D1, D2]. Since a good similarity measure between sequences that predicts the structural or evolutionary similarity is not yet known, biologists resort to the classical string comparisons methods [Sel, SK]. The most commonly used measure of similarity between strings is the well known edit distance measure, which is the minimum-weight set of edit operations (substitutions or insertions/deletions of gaps) that are needed to transform one sequence to another. The minimum edit distance alignment can also be expressed as the best scored alignment; throughout the paper we use the maximization terminology. Biological constraints are imposed via the weights that are assigned to the different edit operations (i.e., the PAM matrix for proteins). It is still controversial how to correctly derive these weights. With any set of weights, the edit distance measure optimizes a single simple objective function, and there is no evidence to believe that this function is being optimized by nature.

Despite these limitations, this technique has become the method of choice in molecular biology for aligning sets of sequences. The main reason is that structural or evolutionary alignments are very hard to find; they involve the determination of the three-dimensional structures of molecules (via crystallography, NMR techniques or phylogenies), which are extremely laborious tasks. String comparison techniques, on the other hand, are much faster and in many cases have proven to be good predictors of the correct alignment. However, lacking experimental data, it is difficult to assess the significance of an alignment that was obtained purely by a computational method.

In summary, the edit distance measure is a well-defined, rigorous combinatorial measure that can be efficiently optimized, and which can be justified biologically (by a set of parameters, whose values are determined empirically). However, the edit distance does not always yield the biological alignment when optimized. It is therefore useful to intelligently explore a larger set of solutions. One approach is the *Parametric* approach, which looks for different optimal solutions obtained for different sets of parameters [FS, GBN, WEL, Wa92]. The second is to consider a larger, but still a manageable, set of alignments that are in the vicinity of the optimum. These are "Suboptimal Alignments" [Zuk, ViAr, WaBy, Wa83, WaEg].

1.2 Summary of Results

In this paper we study the combinatorial nature of suboptimal alignments under a simple scoring function. For every Δ, we define the *minimum* set of edges E_Δ in the edit distance graph that includes *every* suboptimal alignment in the Δ neighborhood, and then show that this set of edges is *exactly* the union of certain types of alignments

which we call "canonical" alignments. We argue that these alignments can be viewed as a *canonical* set for all suboptimal alignments, since any suboptimal alignment is a "combination" of few canonical ones. We denote by $\Delta_0 < \Delta_1 < \Delta_2 < \ldots$ the maximal sequence of values such that

$$E_{\Delta_0} \subset E_{\Delta_1} \subset E_{\Delta_2} \subset \ldots$$

and show that the information revealed by this set of graphs is much richer than merely a single optimal alignment, or a list of few suboptimal ones. It allows a *graphical* view of:

- suboptimal alignments with almost optimal score that are *substantially different* than the optimal alignment.
- the regions that are *common to many* suboptimal alignments, which may indicate that these regions are significant [Zuk, Vi, ViAr].
- the best alignment which satisfies additional biological constraints.

We developed a program, called SUBOPT, that computes and displays these graphs, and counts the number of δ-suboptimal alignments for each δ.

We also define a transformation on the weights of the edit distance graph with which suboptimal alignments can be output efficiently (ordered or unordered). Let n and m be the lengths of the sequences, $m \geq n$. Given the transformed weights, the next best alignment (canonical or non-canonical) can be output in $O(m)$ time. If alignments are to be enumerated in increasing order, then the space requirement is $O(Km+mn)$; otherwise, $O(mn)$ space suffices. We also show how to efficiently *count* the number of suboptimal alignments, rather than enumerate them. This method is extremely simple. It requires only one additional edit distance computation between the reversed strings. Computing the edit distance between the sequences, and between the reversed sequences, was used in the algorithms of [KIM, Vi, ViAr, Wa83, WaBy, Zuk]. This information is sufficient to represent canonical as well as non-canonical suboptimal alignments and can be obtained in $O(nm)$ time and space.

Examples To motivate this problem, we show two examples. The first demonstrates that the biologically correct alignment is not necessarily the optimal one. The second example shows how a compact representation of a *set* of alignments between a *single* pair of sequences may reveal a wealth of information that is traditionally obtained from a multiple alignment of a large set of sequences. In both examples an alignment is represented by a path in a grid graph that starts at the upper-left corner and ends at the bottom-right corner. A diagonal edge corresponds to a pair of aligned characters, and a horizontal or a vertical line corresponds to a gap in one of the two sequences.

To demonstrate how misleading the optimality criteria can be, consider the example in Figure 2. Here, two 25-long amino acid sequences are compared against each other. The two are prealigned substrings of longer "Leucine Zippers" sequences, in which amino acid Leucine (L) appears every 7 positions. In this example, L appears at the 4th, 11th, 18th and 25th positions of the two sequences, so we expect a

good alignment to pick up this periodicity. Hence, we define a biologically "correct" alignment as one that coincides with the diagonal at the 4th, 11th, 18th and 25th positions.

When the two sequences were compared (with a PAM80, gap score -1) the best score was 43. All alignments with this optimal score are represented as paths in the top part of Figure 2. Note that no optimal alignment is the biologically "correct" one under our definition. The bottom part of Figure 2 shows that when the score drops by 2, another large set of paths which are closer to the diagonal are introduced, among them a biologically "correct" one.

In the second example, the variable regions of the heavy and the light chain of a Human Immunoglobulin F_{ab} (the first 120 amino acids of PDB sequence 2fb4_H and 2fb4_L) have been aligned (Figure 3). The graph that represents optimal alignments reveals that all optimal alignments agree on three well-aligned regions ("diagonals"). This phenomena is reinforced by the consideration of suboptimal alignments (for $\delta = 0, 1, 2$), as they all share the same well-aligned regions, and introduce more alternatives at the remaining parts of the alignment. The fact that these two sequences are conserved in three region and are variable between them is the well known fact of "hypervariable regions" in Immunoglobulins. Hence, the hypervariable regions which were discovered in the '70s by aligning *multiple* immunoglobulin sequences can be obtained from this *single* pair of sequences and their suboptimal alignments. Since the Human F_{ab} has been structured, we located the turns that correspond to the hypervariable regions, and correlated them with those observed in the suboptimal graphs. In the heavy chain, the turns occur at positions 27-32, 52-55, 72-75 and 99-111, and the corresponding regions revealed by the suboptimal graphs are 24-32, 47-60, 77-79 and 97-111.

1.3 Problem Definition

Let A and B be two sequences (over a finite alphabet) of lengths m and n respectively, $m \geq n$. Let A_i (B_i) denote the first i characters in A (B). An alignment between A and B introduces spaces into the sequences such that the lengths of the two resulting sequences are identical, and places these spaced sequences one upon the other so that no space is ever above another space. The length of an alignment is between m and $n + m$. A column that contains two identical characters is called a *match* and is assigned a high positive weight *mat*. A column that contains two different characters a and b is called a *mismatch*, and is assigned a weight w_{ab} ($w_{aa} = mat$). A column that contains a space is called a *space*, and is assigned a small weight of *sp*. We define the score of an alignment as the sum over all weights assigned to its columns. (There are many possible definitions for the alignment score – in this paper we only consider this simple definition [2]). The edit distance problem is to find the alignment that maximizes its score.

[2] our results hold for any scoring function with the property that $D(m, n) - D(i, j)$ is the score of optimal alignment between the last $m - i$ characters from A and the last $n - j$ characters from B

If $D(i,j)$ is the score of the best alignment between A_i and B_j, and if the i^{th} and j^{th} characters of A and B are a and b respectively, then the following dynamic programming formulation will correctly compute $D(i,j)$ [SK]:

$$D(i,j) = \max\{D(i-1,j) + sp, D(i-1,j-1) + w_{ab}, D(i,j-1) + sp\}$$

All values $D(i,j)$ can be computed in $O(nm)$ time, as well as $D(m,n)$, which is the desired score of the optimal alignment. Also, **all** optimal alignments can be found by backtracing through the matrix of the $D(i,j)$ values.

This problem can be viewed as a longest path problem on a directed acyclic graph G with nm nodes and $3nm$ edges. A node in G corresponds to some cell (i,j) in an $(m \times n)$ table, and each node (i,j) has three edges coming into it from adjacent cells $(i-1,j),(i-1,j-1),(i,j-1)$. The weight of a horizontal or a vertical edge is sp, whereas the weight of a diagonal edge is w_{ab}, a and b are the ith and jth characters of A and B respectively. This graph is called "the edit distance graph of A and B". If node s corresponds to the cell $(0,0)$, and node t corresponds to the cell (m,n), then there is a 1:1 correspondence between alignments and $s-t$ paths in G. Hence, suboptimal alignments are essentially suboptimal paths in this simple graph.

By the graph analogue of the problem, it is well known that all optimal alignments between A and B (that is, optimal paths from s to t) have a compact representation: it is the backtrace graph, which can be constructed in $O(nm)$ time. (This property does not directly carry over for other definitions of scores, e.g. affine gap weights [Go, AlEr]). However, no such elegant representation exists for suboptimal paths, and therefore alignments. Throughout the paper we use the graph notation, where a node (i,j) is simply denoted by u, s is the node $(0,0)$ and t is (m,n). An edge, directed form u to v, is denoted by $\langle u,v \rangle$.

1.4 Previous Work

The problem of enumerating all K-best shortest paths (or longest paths in a DAG) in increasing order was addressed very early on (in the 60's and 70's) in the context of general graphs, and it is a well studied problem in combinatorial optimization. In general, the problem may or may not allow paths with loops. We do not attempt to give a complete overview of this problem, but rather to summarize some relevant ideas that have been suggested. Three main approaches have been suggested. The first is an iterative procedure, that generalizes the Bellman-Ford algorithm for optimal paths, suggested by [BK, Dr] and improved by [Fox, Law76]. The second employs a general scheme for enumerating the K best objects [Yen, Law72, Law76, Pol]. The third exploits the structural relations between suboptimal paths. It is based on the observation of [HP] that the k^{th}-best path P is a "deviation" of some j^{th}-best path Q, $j < k$, i.e. there is an edge $e = \langle u,v \rangle$ on P such that P's segment up to u is the same as Q, $e \notin Q$, and the portion of P from v to t is an optimal path from v to t. The algorithms of [CKR, HP, KIM, Per, Yen] all show how to store the $k-1$ best paths efficiently so that the k^{th} best, which must be a deviation from one of them, can be found easily. All of these approaches do not require the knowledge of K in advance; the fourth approach of [Sh] relies on a known K.

The edit distance graph is a very regular graph. It is a directed (acyclic) grid of degree 3, and its edge-weights may be negative. For this type of graph, the algorithm of [BK, Dr] as implemented by [Fox, Law76] enumerates the K-best suboptimal paths in $O(mn)$ time per path, where K need not be specified in advance. In fact, it finds the K-best suboptimal paths from any node in the graph to t. The space required is $O(Kmn)$. The algorithm of [Yen,Law72] generates the next best path in $O(Knm)$ time, and $O(Km + nm)$ space.

A more relaxed goal is suggested in [Wa83, WaBy]: enumerate all shortest paths (not necessarily in order) that are within δ of the optimum, where δ is known in advance. For this problem, each path can be enumerated in $O(m)$ time, but only $O(nm)$ space is needed. The requirement to know δ in advance is a limitation, since it is not possible to know a priori which level of significance is the interesting one. Another variant, considered in [WaEg] and used in [ScWa], is to list all K-best non-overlapping local alignments. These enumeration methods (of K-best or δ-suboptimal) turn out not to be practical in the biological application, since the number of suboptimal paths grows very fast, and explicit listing of them provides too much information.

In realizing that explicit enumeration is not the desired representation, both [Zuk] and [Vi, ViAr] suggested building $(m \times n)$ 0-1 matrices S, T that store some *partial* information about all δ-suboptimal alignments. S and T are easily computable in $O(nm)$ time, but in both methods δ needs to be specified in advance. $T(i,j) = 1$ iff **there is** a δ-suboptimal alignment in which the i^{th} character of A and the j^{th} character of B are aligned, and $S(i,j) = 1$ iff in *every* δ'-suboptimal alignment, $\delta' < \delta$, the i^{th} character of A and the j^{th} character of B are aligned. These matrices are then used to assess significance of optimal alignments [ViAr, Zuk], to detect alternatives to the optimal one and to construct multiple alignments [Vi]. This representation, although powerful, looses "connectivity information" since, clearly, not every legal path through the 1-entries of the matrices is a suboptimal alignment.

It is therefore clear that it is not the actual enumeration of suboptimal alignments, but rather a representation of their common or uncommon features, that is needed. This is the motivation of the representation suggested in this paper, which contains more information than the partial 0-1 matrices of [Zuk, Vi, ViAr], and yet is compact. We believe that this additional information, which is manageable, can be valuable in many cases. Also, the transformation of the edge weights suggested in the paper provides an alternative intuitive way to explore the search space of suboptimal alignments.

2 The Combinatorial Structure of Suboptimal Paths

Notation - Let G be the edit distance graph between two sequences A and B, and let E be its set of edges. The weight of an edge $e = \langle u, v \rangle$ is denoted by $w(e)$. Let $d(x, y)$ be the length of the optimal (maximal) path from node x to node y in G. If P is a path that goes through nodes x and y, then $d_P(x, y)$ is the length of the portion of P from x to y. For a path P from s to t (an (s,t)-path), define

$\delta(P) = d(s,t) - d_P(s,t)$. If $\delta(P) = \delta$ ($\delta \geq 0$) then P is called a δ-path (or a δ-suboptimal path). Throughout the paper only simple paths are considered (since G is a DAG, it contains only simple paths).

Given a pair of sequences, two edit distance computations (between the sequences and between their reverse) produce all values $d(s,u)$ and $d(u,t)$ for every node u, as well as the optimal path from s to u and from u to t for every node u.

Definition – For an edge $e \in E$, define $\delta(e) = d(s,t) - (d(s,u) + w(u,v) + d(v,t))$. $\delta(e)$ is therefore the difference between the length of the best path from s to t that uses e and the optimal (s,t) path.

Definition – Define E_Δ, a subset of the edge set E in G, as:

$$E_\Delta = \{e | \delta(e) \leq \Delta\}$$

Claim 1. *E_Δ is the smallest set of edges such that every δ' suboptimal path for some $\delta' \leq \Delta$ is a path in E_Δ.*

Proof. Note first that every δ'-suboptimal path, $\delta' \leq \Delta$, is a path in E_Δ. Let P be a δ'-suboptimal path for some $\delta' \leq \Delta$. Assume that P is not a path in E_Δ, so there must be some edge $e = \langle u, v \rangle \notin E_\Delta$ on P. But since P uses e, $\delta(e) = \delta'' \leq \delta' \leq \Delta$. Hence, $e \in E_\Delta$, a contradiction. E_Δ is the smallest such set since every $e \in E_\Delta$ is on some δ' suboptimal path, $\delta' \leq \Delta$. □

Definition – A δ-path P from s to t is called *canonical* (for e) if there exists an edge $e = \langle u, v \rangle \in P$ such that $\delta(P) = \delta(e)$. That is, a canonical path consists of the best path from s to u, followed by e and further followed by the best path from v to t. Note - there is a canonical path for every edge e; however, a path P can be canonical for more than one edge.

We next argue that the set of canonical paths are the essential ones among all suboptimal paths since all non-canonical suboptimal paths can be derived from them.

Lemma 2. *Let $P = e_1, e_2 \ldots, e_k$ be a canonical Δ-suboptimal path. Then*
(1) $\forall e \in P$, $\delta(e) \leq \Delta$,
(2) there is a contiguous segment $e_l, \ldots e_r$ ($l \leq r$) along P such that $\delta(e_i) = \Delta$ for $l \leq i \leq r$ and $\delta(e_j) < \Delta$ for $j < l$ or $j > r$. Hence, P is a canonical path for $e_l, \ldots e_r$.
(3) $\delta(e_{i-1}) \leq \delta(e_i)$ for all $i \leq l$, and $\delta(e_i) \geq \delta(e_{i+1})$ for all $i \geq r$.

Proof. For any $e \in P$, $\delta(e) = d(s,t) - d_{P'}(s,t)$, where P' is the best (s,t)-path that uses e. Since P uses e, $\Delta = \delta(P) = d(s,t) - d_P(s,t) \geq d(s,t) - d_{P'}(s,t) = \delta(e)$, so $\delta(e) \leq \Delta$ and (1) is proven.

To show (2), let e_l be the first edge on P for which $\delta(e_l) = \Delta$ (since P is a canonical path, there must be such an edge), and let e_r be the first edge on P such

that $\delta(e_r) = \Delta$ but $\delta(e_{r+1}) < \Delta$, or $e_r = e_k$ if no such r exists. By definition, $\delta(e_i) < \Delta$ for $i < l$. If $r = k$, then we are done. Otherwise, it is left to be shown that $\delta(e_i) < \Delta$ for $i > r$. Suppose $e_l = \langle u, v \rangle$. Since P is canonical for e_l, $d_P(v,t) = d(v,t)$, and this holds for all nodes that occur after v on P. Let $e_{r+1} = \langle x, y \rangle$. P is not canonical for e_{r+1}, but since y occurs after v on P we know that $d_P(y,t) = d(y,t)$, hence $d_P(s,x) < d(s,x)$, so there is a better path to reach x from s than via P. Take now any edge $e_i = \langle x', y' \rangle$, $i > r$. P is not canonical for e_i since there is always a better way to reach x' from s than along P: reach x optimally, then follow the portion from x to x' on P. Hence $d_P(s,x') < d(s,x')$, so $\delta(e_i) < \delta(P) = \Delta$ for all $i > r$.

Recall that P is optimal for $e_l = \langle u, v \rangle$. Consider some $i \leq l$, and let $e_{i-1} = \langle p, q \rangle$ and $e_i = \langle q, r \rangle$. e_{i-1} and e_i are on the best path from s to u, and therefore e_{i-1} is on the best path from s to q. Hence, e_{i-1} is on the canonical path for e_i and by (2) $\delta(e_{i-1}) \leq \delta(e_i)$. A similar argument shows that for every $i \geq r$, e_{i+1} is on the canonical path for e_i, hence $\delta(e_i) \geq \delta(e_{i+1})$. (3) is therefore proven. □

A canonical path is depicted in Figure 1.

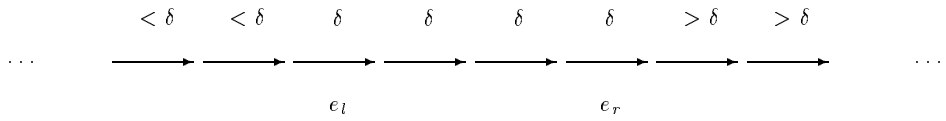

Fig. 1. A Canonical Path

Lemma 3. *If $P = e_1, e_2 \ldots, e_k$ is a non-canonical Δ-suboptimal path, then $\delta(e) < \Delta$ $\forall e \in P$.*

Proof. For any $e \in P$, the length of the best path that uses e is $d(s,t) - \delta(e)$. Since P uses e but is not canonical (therefore not the best) for e, $d_P(s,t) = d(s,t) - \Delta < d(s,t) - \delta(e)$, so $\delta(e) < \Delta$. □

Denote by $\Delta_0 < \Delta_1 < \Delta_2 < \ldots$ the maximal sequence of values such that

$$E_{\Delta_0} \subset E_{\Delta_1} \subset E_{\Delta_2} \subset \ldots$$

Lemma 4. *E_{Δ_i}, which is the smallest set of edges that contains every δ' suboptimal path ($\delta' \leq \Delta_i$), is also the union of all Δ_j canonical paths for $j \leq i$.*

Proof. Recall that a canonical path must be Δ_j-suboptimal for some i. If P is a canonical Δ_j-suboptimal path, then it will first appear as a path in E_{Δ_j} since by Lemma 2 $\delta(e) \leq \Delta_j$ for all $e \in P$, and $\delta(e') = \Delta_j$ for some $e' \in P$. Also, every edge in E_{Δ_i} belongs to some canonical Δ_j-suboptimal path, where $\Delta_j \leq \Delta_i$ $(j \leq i)$. □

Lemma 4 implies that the union of all **canonical suboptimal** paths within Δ of the optimum contain **all** suboptimal paths in this neighborhood. Hence, any non-canonical δ-suboptimal path is a combination of few segments from canonical paths that are δ'-suboptimal, $\delta' < \delta$. In general, let P_1 and P_2 be δ_1 and δ_2 suboptimal paths, respectively, which intersect at some node u. Then, any path P_3 obtained by concatenating the (s,u) segment from one path and the (u,t) from the other is δ_3 suboptimal, $\delta_3 \leq \delta_1 + \delta_2$. In the special case where P_1 and P_2 are both optimal, P_3 is also optimal.

In terms of alignments, in order to cope with this combinatorial explosion, we suggest the subset of alignments that correspond to the canonical paths as the representatives for the entire set of suboptimal alignments, which can be very large. This set fairly represents the entire set of alignments since any non-canonical alignments can be obtained by recombining few canonical ones.

Remark - Define $G_\Delta \equiv (V, E_\Delta)$. The above discussion also implies that any task that involves alignments that are at most Δ suboptimal (such as counting and enumeration) can be done on G_Δ instead of G, taking advantage of its sparsity. No theoretical bounds are known on the ratio of $|E_\Delta|/3nm$. However, it is typically the case that the interesting Δ is the one for which $|E_\Delta| = O(n+m) = O(|E|^{0.5})$, to avoid combinatorial explosion.

3 Transformation of Weights

This paper advocates for representation, rather than enumeration, of alignments. However, in some cases it may be desirable to enumerate alignments. Also, recall that E_Δ may also contain *non-canonical* paths that are worse than Δ. In this section we show that a simple transformation on the weights of E provides a powerful tool to manipulate suboptimal paths and to accomplish the tasks mentioned above. This transformation turns out to be related to the method of Edmonds and Karp [EK72] (see also [Law76] that transforms general weights of a graph to all non-negative weights such that the order of paths are preserved. Our transformation specializes the Edmonds-Karp transformation to (s,t)-paths; it produces a set of non-negative weights (recall that the original set of weights may be negative) which preserves the order of the (s,t) paths, and makes the enumeration extremely easy.

Definition: For every $e = \langle u, v \rangle \in E$ define

$$\epsilon(e) = \delta(e) - \min_{e' = \langle x, u \rangle} \{\delta(e')\}$$

We now prove (Lemma 5[3]) that $\epsilon(e)$ can be interpreted as the "additional penalty for using e on the path from u to t rather than following the optimal path from u to t directly". Theorem 6 shows how these transformed weights can be used.

Lemma 5. *For any $e = \langle u, v \rangle$*
(1) $\epsilon(e) = \delta(e) - \delta(e')$, where e' is the edge preceding e on the canonical path for e.
(2) $\epsilon(e) \geq 0$
(3) $w(e) + d(v,t) = d(u,t) - \epsilon(e)$.

Proof. Let $e_i = \langle x_i, u \rangle$ be the set of edges entering u and assume without loss of generality that $\delta(e_1) \leq \delta(e_2) \leq \ldots$ Note that $\delta(e_i) = d(s,t) - (d(s,x_i) + w(e_i) + d(u,t))$; hence $d(s,x_1) + w(e_1)$ is the optimal way to reach u from s since $d(s,x_1) + w(e_1) \geq d(s,x_2) + w(e_2) \geq \ldots$ This implies that the canonical path for e enters u via edge $e_1 = \langle x_1, u \rangle$, so $\epsilon(e) = \delta(e) - \delta(e_1)$ and (1) follows.

From Lemma 2 we know that if P is a Δ-suboptimal path that is canonical for e, then $\delta(e) = \Delta$ and $\delta(e') \leq \Delta$ for any edge e' preceding e on P. Hence (2) follows.

Let $e_1 = (x_1, u)$ from above. We know
(i) $\delta(e_1) = d(s,t) - (d(s,x_1) + w(e_1) + d(u,t))$
(ii) $\delta(e) = d(s,t) - (d(s,x_1) + w(e_1) + w(e) + d(v,t))$
hence

$$\epsilon(e) = \delta(e) - \delta(e_1) = d(u,t) - w(e) - d(v,t)$$

or $w(e) + d(v,t) = d(u,t) - \epsilon(e)$. □

Theorem 6. *For any path P from s to t, $\delta(P) = \sum_{e \in P} \epsilon(e)$.*

Proof. Let $P = e_1, e_2 \ldots, e_k$, where $e_i = \langle u_i, v_i \rangle$ ($u_1 = s, v_k = t$ and $v_i = u_{i+1}$). Define the path P_i as the path obtained by concatenating the edges $e_1, \ldots, e_i$ with the optimal path from v_i to t. We claim, by induction on i, $i = 1, \ldots, k$, that $\delta(P_i) = \sum_{j \leq i} \epsilon(e_j)$. Since $P_k = P$, the theorem follows.

Note that P_1 is canonical for e_1; therefore $\delta(P_1) = \delta(e_1)$. Furthermore, $\epsilon(e_1) = \delta(e_1)$ as there are no edges into s. Hence, for $i = 1$, $\delta(P_1) = \epsilon(e_1)$ as claimed.

Assume correctness for $j \leq i$. P_i and P_{i+1} share the first i edges. Since $d_{P_i}(s,t) = d_{P_i}(s,v_i) + d(v_i,t)$ we have

$$d_{P_{i+1}}(s,t) = d_{P_i}(s,v_i) + w(e_{i+1}) + d(v_{i+1},t) = \quad (1)$$

$$= d_{P_i}(s,t) - d(v_i,t) + w(e_{i+1}) + d(v_{i+1},t) \quad (2)$$

Recall that $v_i = u_{i+1}$. From Lemma 5 we have

$$w(e_{i+1}) + d(v_{i+1},t) = d(u_{i+1},t) - \epsilon(e_{i+1}) = d(v_i,t) - \epsilon(e_{i+1}) \quad (3)$$

Substituting (3) in (2) we get

$$d_{P_{i+1}}(s,t) = d_{P_i}(s,t) - \epsilon(e_{i+1})$$

By induction $d_{P_i}(s,t) = d(s,t) - \sum_{j \leq i} \epsilon(e_j)$, hence

$$d_{P_{i+1}}(s,t) = d(s,t) - \sum_{j \leq i+1} \epsilon(e_j)$$

so $\delta(P_{i+1}) = \sum_{j=1}^{i+1} \epsilon(e_j)$. □

The proof of Theorem 6 implies that after $\epsilon(e)$ has been computed for every edge e, the "goodness" of a path can be computed "on the fly" as follows. Take a path from s to u for some node u: if δ is the sum of the ϵ's along that path, then there is always a δ-suboptimal path from s to t that begins with this segment from s to u; namely, the one that proceeds with the optimal path from u to t. If an edge e is followed after u, then *any* path to t that uses the segment up to u, followed by e, will be at least $(\delta + \epsilon(e))$-suboptimal.

3.1 Enumerating Suboptimal Paths

A natural enumeration procedure for all suboptimal paths is now readily available. First, compute $\delta(e)$ and $\epsilon(e)$ for every edge in G. This preprocessing takes $O(nm)$ time and space. Then, using the new set of weights $\epsilon(e)$, enumerate the (s,t) paths in order. This can be done as follows: Build a "search tree" that explores all the paths in this graph, starting at s. An internal node in the tree represents a partial path that starts at s and ends at some node u; the leaves represent complete (s,t)-paths. Each internal node is expanded via all edges eminenting from the last node u on the partial path.

With each internal node in the tree we associate a cost δ, which is to the sum of $\epsilon(e)$ of all edges e on the path that is represented by this node. This path can always be extended to an (s,t) path P with $\delta(P) = \delta$. When an internal node is expanded via an edge e, the new node receives the cost of its parent + $\epsilon(e)$.

It is now straightforward to observe (from Theorem 6) that if the tree is expanded in a "best first search" manner (i.e. the next node to expand is the internal node with the minimum δ), then paths are enumerated in increasing order. Hence, to enumerate the K-best paths, simply stop after the K^{th} leaf in the search tree has been reached. A priority queue that maintains the best K costs of the nodes in the search tree needs to be maintained. The size of the tree is at most $O(Km)$ nodes, as every leaf is preceded by at most $m+n$ nodes. After the $O(nm)$ time preprocessing stage, a new path P is enumerated in this manner in $O(|P|) = O(n+m)$ time. The priority queue maintenance takes $O(\log K)$. Since the entire search tree needs to be kept throughout, the space requirement is $O(K(n+m))$ (plus an additional $O(nm)$ space to store the $\epsilon(e)$'s).

The same method can be employed to list all δ'-suboptimal paths for $\delta' \leq \Delta$, where Δ is a specified threshold. First build E_Δ; then enumerate paths in E_Δ by building the search tree, while pruning any extension that leads to a node whose cost exceeds Δ. This is basically the method of [WaBy]. Since any extended node eventually leads to a legal leaf (whose cost is within the threshold), and since at any node there is a constant number of extensions to check, the search tree can be explored in any order. Only the current path needs to be maintained, so the space required is $O(m+n)$, and the time is $O(|P|)$ per path.

A More Efficient Enumeration Algorithm Using the idea of "deviations", together with the transformed weights, another algorithm that outputs suboptimal paths in increasing order can be suggested. It requires only $O(K)$ space and $O(|P|)$ time per path. For simplicity, assume that there are no ties between path lengths (e.g. impose lexicographic order in addition to the length).

The algorithm builds a search tree (which is different from the search tree described above). A node in the tree represents a solution (a path), and its cost is the path length. A node b is a child of a node a in the tree if the path represented by b is a deviation of the path represented by a. Specifically, let a be a node in the tree that represents a path $P = x_1, \ldots x_k$ which deviates from its parent by the edge

$\langle x_i, x_{i+1} \rangle$. Let the cost of a (i.e. $\delta(P)$) be δ. Note that $\epsilon(\langle x_j, x_{j+1} \rangle) = 0$ for all $j > i$. The children of a are found as follows: for every vertex x_j on P, $j > i$, let e_1, e_2 be the two eges that are incident at x_j but **not on** P (i.e. $\langle x_j, x_{j+1} \rangle \neq e_1, e_2$). Create two new nodes, b_1 and b_2, with costs $\delta + \epsilon(e_1)$ and $\delta + \epsilon(e_2)$ respectively, and make them the children of a. b_1 and b_2 share the prefix $x_1, \ldots, x_j$ with a, and deviate from it on the edges e_1 and e_2 respectively.

The algorithm, which outputs the first, second, third ... paths in order, proceeds at follows: Initially, the tree consist of a root, where the root of the tree is the optimal path from s to t, and its cost is 0. To find the next best path, find the leaf a in the tree of minimal cost, and ouput the path represented by a. Expand a by attaching all deviations of it as its children.

Since expanding a node requires $O(n+m)$ time (need to check at most two edges at each vertex of the path), the time to output the next path is $O(n+m)$. The size of the search tree is $O(Km)$ since every new solution creats at most $2(n+m)$ new nodes. However, this space requirement can be improved by using the technique of [Law72]: when a leaf a is first expanded, only its best child $b^\star$ needs to be explicitly stored in the tree. When $b^\star$ is eventually used at some iteration (i.e it is the next best path), $b^\star$ is expanded as before, but also the **next** best child of a is found again. The is repeated for the third, fourth ... child of a. This assures that after K solutions have been output, there are only $2K$ nodes in the tree. Moreover, at each iteration, two nodes (instead of one node) are expanded, hence the time requirement is only doubled whereas the space requirement is reduced by a factor of m.

4 Counting Suboptimal Alignments

Simulations show that the number of suboptimal alignments grows rapidly as the threshold increases. This behavior depends on the scoring system, i.e. on the weights of the edges in the graph, as well as on the actual sequences. Hence, the significance of an alignment can be assessed not only by the distance between its score and the optimal score, but also by its **rank**, i.e. the *number* of alignments with better scores than its own score. Computing the rank of a given alignment is closely related to the problem of *counting* the number of suboptimal alignments. The number of suboptimal alignments can clearly be computed by enumerating them using the methods of Section 3.1. In this section we are interested in counting methods that are more efficient than the corresponding enumeration solution. We show that by using the transformed set of weights $\epsilon(e)$ instead of the original weights, better bounds can be achieved. As before, we use the graph notation, where alignments correspond to paths from s to t in the edit distance graph.

The number of *optimal* paths from s to t is found in time and space $O(|E_0|)$ as follows. Let $\mathcal{N}(v)$ be the number of optimal paths from s to some node v. Then,

$$\mathcal{N}(v) = \sum_{u | \langle u,v \rangle \in E_0} \mathcal{N}(u)$$

$\mathcal{N}(t)$ is the desired count. A natural generalization of this method to count all suboptimal paths within Δ of the optimum is to maintain, at each node v, a list

of all possible lengths of paths from s to v, as well as their count. A node creates its own list by inheriting and merging the lists and the counts of its predecessors [Wa92]. This evaluation requires $O(C|E_\Delta|) = O(Cnm)$ time and space, where C is the maximum list size (C can be very large for arbitrary weights).

Let Δ be a specified threshold. If the number of paths which are δ suboptimal for $\delta \leq \Delta$ is sought, then the $O(Cnm)$ bound can be dramatically reduced by using the transformed set of weights. Recall that if the weights on the edges of the graph are transformed as suggested in Section 3, then the paths which are δ suboptimal, $\delta \leq \Delta$, are exactly the paths whose transformed length does not exceed Δ. Hence, if we compute $\epsilon(e)$ for each edge $e \in E_\Delta$, then we can count the number of these paths in the transformed graph as follows when the weights are all integers: Let $C(v,k)$, $k = 0, 1, \ldots, \Delta$, be the number of paths of length k from s to v in the transformed graph. Then,

$$\mathcal{N}(v,k) = \sum_{u|\langle u,v\rangle \in E_\Delta} \mathcal{N}(u, k - \epsilon(\langle u,v\rangle))$$

This recursive formula can be evaluated in $O(\Delta\ |E_\Delta|) = O(\Delta nm)$ time and space if we use the transformed weights $\epsilon(e)$. A similar solution on the original set of weights requires $O((d(s,t) + \Delta)\ nm)$ time and space. Similarly, for non-integral weights, the space requirement will be reduced.

Counting Canonical Suboptimal Path

The set of *canonical* suboptimal paths is a smaller, restricted and more structured set of suboptimal paths. To count the number of canonical δ suboptimal paths we need the following lemma:

Lemma 7. *$P = e_1, e_2, \ldots, e_k$ is a canonical path iff there is some l, $1 \leq l \leq k$, such that*
(1) for any $i \leq l$, $\sum_{j \leq i} \epsilon(e_j) = \delta(e_i)$, and
(2) $\epsilon(e_i) = 0$ for all $i > l$.

Proof. Suppose first that (1) and (2) hold. Then, $\sum_{j \leq k} \epsilon(e_j) = \delta(e_l)$, and from Theorem 6 $\delta(P) = \delta(e_l)$, hence P is canonical for e_l.

We prove the only if part by induction on Δ. Assume that for any canonical δ-suboptimal path, $\delta < \Delta$, (1) and (2) hold. Let P be a canonical Δ-suboptimal path, and let $e_l = \langle v,w\rangle$ be the first edge on P such that $\delta(P) = \delta(e_l) = \Delta$. We will show that (1) and (2) hold for this l. Suppose that $e_{l-1} = \langle u,v\rangle$ and let P' be a canonical path for e_{l-1}. Consider the segment on P from s to v; it is the best path from s to v and it also goes through u, hence its portion up to u is the best path from s to u. Therefore, P' coincides with P on the segment up to u. Lemma 2 implies that $\delta(P') = \delta(e_{l-1}) < \delta(e_l) = \Delta$. Hence, by induction, for any $i \leq l-1$, $\sum_{j \leq i} \epsilon(e_j) = \delta(e_i)$. e_{l-1} precedes e_l on the canonical path for e_l, so from Lemma 5 $\epsilon(e_l) = \delta(e_l) - \delta(e_{l-1})$. Hence, (1) holds since

$$\sum_{j \leq l} \epsilon(e_j) = \delta(e_{l-1}) + \epsilon(e_l) = \delta(e_{l-1}) + \delta(e_l) - \delta(e_{l-1}) = \delta(e_l)$$

Now (2) follows directly, since $\delta(P) = \delta(e_l)$ by definition, and $\delta(P) = \sum_{j \le k} \epsilon(e_j) = \delta(e_l) + \sum_{l < j \le k} \epsilon(e_j)$ which implies $\epsilon(e_i) = 0$ for all $i > l$. □

The above lemma implies that every canonical path $P = e_1, e_2, \ldots, e_k$ can be decomposed into two parts, $P_1 = e_1, e_2, \ldots, e_l$ and $P_2 = e_{l+1}, \ldots, e_k$, such that in $P1$ the property $\sum_{j \le i} \epsilon(e_j) = \delta(e_i)$ holds for any i, and in P_2 $\epsilon(e_i) = 0$ for any i. If we choose P_1 as the maximal segment with this property, then this decomposition is unique. We therefore associate with each canonical path P a pair of edges (e, f), where e is the last edge on the P_1 (maximal) segment, and f is the first edge on the P_2 segment. Note that $\delta(P) = \delta(e)$, and that from the maximality of P_1, $\delta(e) + \epsilon(f) > \delta(f)$.

Many canonical paths may be associated with the same pair of edges (e, f). Denote by $\mathcal{N}(e, f)$ the number of canonical ($\delta(e)$-suboptimal) paths that are associated with the pair (e, f). Since each canonical path is associated with a unique pair, all we need is to count how many canonical paths are associated with each pair of adjacent edges (e, f).

Define $\mathcal{N}_{can}(e)$ as the number of paths that start at s and end with the edge e, which have the property that $\sum_{j \le i} \epsilon(e_j) = \delta(e_i)$ for any edge e_i along the path. Also, define $\mathcal{N}_{opt}(f)$ as the number of paths that start with the edge f and end at t such that $\epsilon(e) = 0$ for any edge e along the path. Then,

$$\mathcal{N}(e, f) = \begin{cases} \mathcal{N}_{can}(e) + \mathcal{N}_{opt}(f) & \text{if } \delta(e) + \epsilon(f) > \delta(f) \\ 0 & \text{otherwise} \end{cases}$$

$\mathcal{N}_{can}(e)$ and $\mathcal{N}_{opt}(f)$ are computed as follows. Let e_1, e_2, e_3 be the three edges that are immediate predecessors of e, and f_1, f_2, f_3 be the three edges that are immediate successors of f. Set $I_j = 1$, $j = 1, 2, 3$, if $\delta(e_j) + \epsilon(e) = \delta(e)$, otherwise $I_j = 0$. Then, initially, $\mathcal{N}(e) = 1$ for $e = \langle s, u \rangle$, and

$$\mathcal{N}_{can}(e) = \sum_{j=1,2,3} I_j \mathcal{N}_{can}(e_j)$$

To compute $\mathcal{N}_{opt}(f)$, initially set $\mathcal{N}_{opt}(f) = 1$ if $f = \langle u, t \rangle$ and $\epsilon(f) = 0$, and

$$\mathcal{N}_{opt}(f) = \begin{cases} \sum_{j=1,2,3} \mathcal{N}_{opt}(f_j) & \text{if } \epsilon(f) = 0 \\ 0 & \text{otherwise} \end{cases}$$

There are $3nm$ pairs of edges to consider. The recursive evaluation of $\mathcal{N}_{can}(e)$ and of $\mathcal{N}_{opt}(f)$ requires $O(nm)$ time and space. Hence the total time and space is $O(nm)$. This can be reduced to $O(|E_\Delta|)$ if we are only concerned with canonical paths that are δ-suboptimal for some $\delta \le \Delta$.

5 Acknowledgements

We would like to thank Dan Gusfield, Gene Lawler, Frank Olken and Martin Vingron for valuable discussions, and Tod Klingler for many helps, and in particular in interpreting the graphs of Figure 3.

References

[AlEr] S.F Altschul and B. W. Erickson, *Optimal Sequence Alignment Using Affine Gap Costs,* Bull. Math. Biol. 48: 603–616, 1986.

[BK] R. Bellman and R. Kalaba, *On Kth Best Policies,* J. SIAM, 8(4):582–588, 1960.

[CKR] S. Clarke, A. Krikorian and J. Rausen, *Computing the N best Loopless Paths in a Network,* J. Siam, 11 (4): 1096–1102, 1963.

[D1] R.F. Doolittle, *Of Urfs and Orfs: A Primer on How to Analyze Derived Amino Acid Sequences,* University Science Books, 1986.

[D2] R.F. Doolittle, edt. *Methods in Enzymology, Volume 183: Molecular Evolution: Computer Analysis of Protein and Nucleic Acid Sequences*, 1990.

[Dr] S.E. Dreyfus, *An Appraisal of Some Shortest-Path Algorithms,* Operations Research, 17 (1969), 395–412.

[EK72] J. Edmonds and R.M. Karp, *Theoretical Improvements in Algorithmic Efficiency for Network Flow Problems,* JACM, 19(2):248–264, 1972.

[FS] W.M. Fitch and T.F. Smith, *Optimal Sequence Alignments,* Proc. Natl. Acad. Sci. USA 80(1983) 1382–1386.

[Fox] B. Fox, *Calculating the Kth Shortest Paths,* INFOR, Canad. J. Oper. Infor. Process., 11:66–70, 1973.

[Go] O. Gotoh, *An Improved Method for Matching Biological Sequences,* J. Mol. Biol., 162:705–708, 1982.

[GBN] D. Gusfield, K. Balasubramanian and D. Naor, *Parametric Optimization of Sequence Alignment,* Proceedings of the third annual ACM–SIAM Joint Symposium on Discrete Algorithms, Orlando, Florida, Jan. 1992.

[HP] W. Hoffman and R. Pavley, *A Method for the Solution of the Nth Best Path Problem,* J. ACM 6, 506–514 (1959).

[KIM] N. Katoh, T. Ibaraki and H. Mine, *An Efficient Algorithm for K Shortest Simple Paths,* Networks, 12:411–427, 1982.

[Law72] E. L. Lawler, *A Procedure for Computing the K-best Solutions to Discrete Optimization Problems and its Applications to the Shortest Paths Problem,* Manag. Sci. 18:401–405, 1972.

[Law76] E. L. Lawler, *Combinatorial Optimization, Networks and Matroids,* Holt, Rinehart and Winston, 1976.

[Per] A. Perko, *Implementation of Algorithms for K Shortest Loopless Paths,* Networks 16:149–160, 1986.

[Pol] M. Pollack, *The kth Best Route Through A Network,* OR, 9 (4):578–580, 1961.

[SK] D. Sankoff and J. Kruskal, Editors, *Time Warps, String Edits, and Macromolecules: The Theory and Practice of Sequence Comparison,* Addison-Wesley, 1983.

[ScWa] M. Schoniger and M.S. Waterman, *A Local Algorithm for DNA Sequence Alignment with Inversions,* Bull. Math. Bio. 54(4):521–536, 1992.

[Sel] P. H. Sellers, *An Algorithm for the Distance Between Two Finite Sequences,* J. Comb. Theory. (A), 16:253–258, 1974.

[Sh] D.R. Shier, *On Algorithms for Finding the K Shortest Paths in a Network,* Networks, 9:195-214, 1979.

[Vi] M. Vingron, *Multiple Sequence Alignment and Applications in Molecular Biology,* Preprint 91–12, Universitat Heidelberg, 1991.

[ViAr] M. Vingron and P. Argos, *Determination of Reliable Regions in Protein Sequence Alignments,* Protein Engin., 7: 565–569, 1990.

[Wa83] M.S. Waterman, *Sequence Alignments in the Neighborhood of the Optimum with General Application to Dynamic Programming,* Proc. Nat, Acad, Sci. USA. 80:3123–3124, 1983.

[Wa92] M.S. Waterman, *Parametric and Ensemble Sequence Alignment Algorithms,* manuscript.

[WaBy] M.S. Waterman and T.H. Byers, *A Dynamic Programming Algorithm to Find All Solutions in a Neighborhood of the Optimum,* Math, Biosci. 77:179–188, 1985.

[WaEg] M.S. Waterman and M. Eggert, *A New Algorithm for Best Subsequence Alignments With Applications to tRNA-rRNA Comparisons,* J. Mol. Biol. 197:723–728, 1987.

[WEL] M.S. Waterman, M. Eggert and E. Lander, *Parametric Sequence Comparisons,* PNAS 89:6090–6093, July 1992.

[Yen] J.Y. Yen, *Finding the K Shortest Loopless Paths in a Network,* Management Science, 17 (11):712–716, 1971.

[Zuk] M. Zuker, *Suboptimal Sequence Alignment in Molecular Biology, Alignment with Error Analysis,* J. Mol. Biol., 221:403–420, 1991.

This article was processed using the LaTeX macro package with LLNCS style

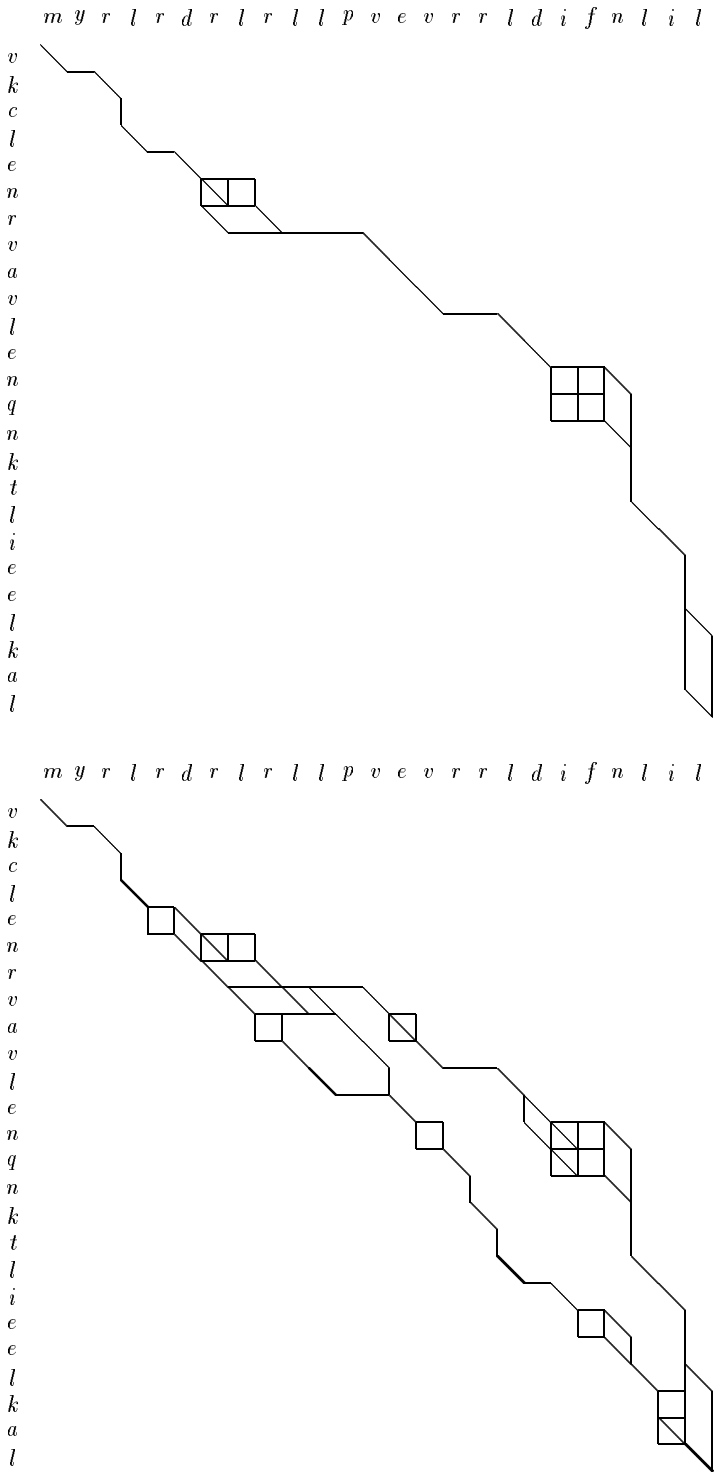

Fig. 2. Aligning two Leucine Zippers: the top figure depicts E_0, and the bottom figure depicts E_2. Note that the biologically "correct" alignment, the one that coincides with the diagonal at the 4th, 11th, 18th and 25th positions, is in E_2.

Fig. 3. The compact representation of suboptimal alignments between the variable regions of a heavy and a light chain of a Human Immunoglobulin reveals their hypervariable regions.