A Branch-and-Cut Algorithm for Multiple Sequence Alignment

K. Reinert*† H.-P. Lenhof* P. Mutzel* K. Mehlhorn* J.D. Kececioglu‡

Abstract

Multiple sequence alignment is an important problem in computational biology. We study the Maximum Trace formulation introduced by Kececioglu [Kec91]. We first phrase the problem in terms of forbidden subgraphs, which enables us to express Maximum Trace as an integer linear-programming problem, and then solve the integer linear program using methods from polyhedral combinatorics. The trace *polytope* is the convex hull of all feasible solutions to the Maximum Trace problem; for the case of two sequences, we give a complete characterization of this polytope. This yields a polynomial-time algorithm for a general version of pairwise sequence alignment that, perhaps suprisingly, does not use dynamic programming; this yields, for instance, a non-dynamic-programming algorithm for sequence comparison under the 0-1 metric, which gives another answer to a long-open question in the area of string algorithms [PW93]. For the multiple-sequence case, we derive several classes of facet-defining inequalities and show that for all but one class, the corresponding separation problem can be solved in polynomial time. This leads to a branch-and-cut algorithm for multiple sequence alignment, and we report on our first computational experience. It appears that a polyhedral approach to multiple sequence alignment can solve instances that are beyond present dynamic-programming approaches.

1 Introduction

Let Σ be a finite alphabet and let $\hat{\Sigma} = \Sigma \cup \{-\}$, where "$-$" (dash) is a symbol to represent "gaps" in strings. The input of a multiple-sequence alignment algorithm is a set $S = \{S_1, S_2, \cdots S_k\}$ of finite strings over the alphabet Σ. A set $\hat{S} = \{\hat{S}_1, \hat{S}_2, \cdots, \hat{S}_k\}$ of strings over the alphabet $\hat{\Sigma}$ is called an *alignment* of S if the following two properties hold: (1) the strings in $\hat{S}$ have the same length, and (2) ignoring dashes, string $\hat{S}_i$ is identical with string S_i. An alignment can be interpreted as an array with k rows, one row for each $\hat{S}_i$. Two letters of distinct strings are called aligned under $\hat{S}$ if they are placed into the same column. The multiple-sequence alignment problem asks to find a "good" alignment $\hat{S}$ of S. There are many ways to measure the quality of an alignment. We discuss two.

In the *Complete Maximum Weight Trace* formulation (CMWT) the letters of the strings $S_i = (s_{i1}, \ldots, s_{in_i})$ of S are viewed as vertices V in a *complete* k-partite graph G. Every edge $e \in G$ has a non-negative weight representing the gain of aligning the endpoints of the edge. We say that an alignment realizes an edge if it places the endpoints into the same column of the alignment array. The set of edges realized by an alignment $\hat{S}$ is called the *trace* of $\hat{S}$, denoted trace($\hat{S}$), and the weight of an alignment is the sum of the weights of the edges that it realizes. The goal is to

*MPI für Informatik, Im Stadtwald, 66123 Saarbrücken, Germany

†Corresponding author, email:kreinert@mpi-sb.mpg.de

‡Department of Computer Science, The University of Georgia, Athens, Georgia 30602, USA

compute an alignment $\hat{S}$ of maximum weight. The CMWT formulation is quite general, containing the well-studied Sum of Pairs Alignment problem with uniform insertion and deletion costs as a special case. It has the computational drawback that the number of edges in G is $\sum_{1 \le i < j \le k} n_i n_j$ which becomes large quickly as the number of sequences or their length grows.

The *(Sparse) Maximum Weight Trace* formulation (MWT) is a sparsification of CMWT. It restricts consideration to a judiciously chosen subgraph of the complete k-partite graph. We use $G = (V, E)$ to denote this graph and call it the *alignment graph*. In our computational experiments we put all edges into E that are used in some nearly optimal pairwise alignment (see Section 5 for more details). Alternatively, the edges of the alignment graph may be obtained from pairwise alignment under arbitrarily complex objective functions. They can encode suboptimal pairwise alignments, and may even represent correspondences from 3-d alignments of protein structures. The hope is that MWT is computationally simpler than CMWT (since the input graph is smaller) and yet has a biologically meaningful optimal solution (as E is carefully chosen). MWT was introduced in 1991 by Kececioglu [Kec91] originally as a model for the final alignment phase of DNA sequence assembly and is unique as a graph-theoretic formulation of multiple alignment.

Like most multiple alignment problems, Maximum Weight Trace can be solved by dynamic-programming, and is equivalent to finding a longest-path from a designated source to a designated sink in a k-dimensional acyclic mesh-shaped graph, the so-called dynamic programming graph. The set of paths from the source to the sink code all possible alignments. Each (directed) edge of the dynamic programming graph represents a possible column. The weight of such an edge is the sum of the weights of the edges $e \in E$ (of the input graph) that are realized by the corresponding column. Dynamic programming yields an algorithm with time complexity $O(k^2 2^k N)$ and space complexity $O(N)$, where $N = \prod_i n_i$ which is feasible only for very small problem instances. While MWT is NP-complete, Kececioglu [Kec93] presented a branch-and-bound algorithm whose implementation could optimally align six sequences of length 250 in a few minutes. Larger examples, however, required excessive space. In his approach, a heuristic alignment of the k sequences yields a lower bound for the branch-and-bound procedure. Upper bounds are calculated by adding up the weights of all MWT-optimal pairwise alignments over suffixes of the sequences. On instances that can be solved, the set of subproblems is kept manageable by an elaborate procedure for branching on a small set of columns, and by pruning subproblems via bounds.

In this paper we formulate Maximum Weight Trace as an integer linear program. Define the *trace polytope* of G as the convex hull of all incidence vectors of traces in G, i.e.,

$$P_T(G) := conv\{\chi^T \in \{0,1\}^{|E|} \mid T \subseteq E \text{ is a trace in } G\},$$

where a subset T of E is called a trace if it is the trace of some alignment and χ^T denotes the characteristic vector of T. The polyhedral combinatorics to integer linear programming calls for a description of $P_T(G)$ by linear inequalities (also called constraints) after which linear programming can be used to find the Maximum Weight Trace.

The most effective constraints are the ones that define facets of the trace polytope $P_T(G)$, since these are the constraints that appear in an irredundant description of $P_T(G)$ by linear inequalities. Due to the NP-hardness of our problem, we cannot expect to be able to find a full description of the facial structure of $P_T(G)$ by linear inequalities. Nevertheless, a partial description of the facial structure of $P_T(G)$ by linear inequalities is useful for the design of a branch-and-cut algorithm.

In Section 2 we give a graph-theoretic characterization of traces which we use in Section 3 to give a *complete* description of the trace polytope for the two-sequence case, i.e., we give the complete set of the linear inequalities and (in Section 4) polynomial time separation algorithms for them (a separation algorithm for a class of inequalities takes a point in $\mathrm{Q}^{|E|}$ and returns a

violated inequality from the class if there is one). Our complete description implies the existence of a polynomial-time, non-dynamic-programming algorithm for pairwise sequence alignment and thus gives another answer to a longstanding problem in stringology. The inequalities naturally fall into two classes which we call *trivial* and *clique inequalities* respectively. Furthermore we give a partial description of the trace polytope for the multiple-sequence case. More specifically, we show that all facet-defining inequalities of the two-sequence case lift to the multiple-sequence case and we derive three more classes of facet-defining inequalities (which we call *mixed-cycle inequalities*, *chorded-mixed-cycle inequalities* and *ladder inequalities* respectively). In Section 4 we derive polynomial time *separation algorithms* for all but the last class of inequalities and we outline a branch-and-cut algorithm. In Section 5 we report on initial computational experience with the branch-and-cut algorithm; the implementation makes use of the branch-and-cut framework ABACUS [Thi95].

In the appendix we give an example of 15 optimally aligned sequences of prion proteins. In comparison, to our knowledge there is no dynamic programming algorithm that can align this many sequences optimally, even if the sequences are very similar. This is due to the fact that dynamic programming approaches generally evaluate $O(2^k)$ edges for each vertex visited in the dynamic programming graph. Thus they are highly sensitive to the number of sequences to be aligned, while a polyhedral approach seems to be less sensitive to the number of sequences and more sensitive to the number of variables in the corresponding linear program.

2 A graph-theoretic characterization of traces

In this section we give a graph-theoretic characterization of traces in a form that is helpful for expressing Maximum Weight Trace as an integer linear program. It is convenient to extend the alignment graph to a mixed graph (V, E, H) by adding a set of directed edges

$$H = \{(s_{ij}, s_{ij+1}) |\ 1 \leq i \leq k,\ 1 \leq j < n_i\}.$$

We call this graph the *extended alignment graph (EAG)*. We use S_i to denote all vertices s_{ij} with $1 \leq j \leq n_i$. Recall that the edges in E are undirected and non-negatively weighted. We extend the weight function to $E \cup H$ by assigning weight 0 to all edges in H.

Definition 1 *A path in a mixed graph is a sequence $v_1, v_2, \ldots, v_l$ of vertices such that either $\{v_i, v_{i+1}\} \in E$ or $(v_i, v_{i+1}) \in H$, for all i, $1 \leq i < m$. A path is called a mixed path if it contains at least one arc in H. The size of a mixed path is the number of edges in E it contains.*

A mixed path is called a mixed cycle iff the first and the last vertex on the path is the same. A mixed cycle R is called critical if for all i, $1 \leq i \leq k$, all vertices in $R \cap S_i$ occur consecutively in R.

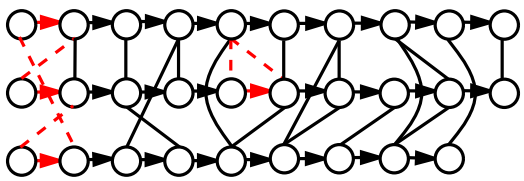

Figure 1: Two mixed cycles (dashed)

Theorem 1 *Let $G = (V, E, H)$ be an EAG, let $T \subseteq E$ and let $G' = (V, T, H)$ be the EAG induced by T. Then T is a trace iff there is no critical mixed cycle in G'.*

Proof: Assume first that T is a trace. Let $\hat{S}$ be an alignment that realizes T. An alignment arranges the vertices of G into columns such that all edges in T connect vertices in the same column and such that all arcs in H run from left to right. Thus G' contains no mixed cycle.

Assume next that G' contains no critical mixed cycle. We show first that G' contains no R-cycle and then construct an alignment with trace T. Assume first that G' contains a mixed cycle. Consider a smallest mixed cycle R and assume that it is not critical. Then there is some i such that the vertices in $R \cap S_i$ are not consecutive in R. Let y be the rightmost vertex in $R \cap S_i$ and let Q be the subpath of R starting in y and ending in the next vertex x on $R \cap S_i$. If $x = y$, then either Q or R without the "loop" Q is a mixed cycle smaller than R. If $x \neq y$, then Q together with the path of arcs between x and y is a mixed cycle smaller than R. In both cases we have a contradiction. Thus G' contains no mixed cycle. Let $C_1, \ldots, C_m$ be the connected components of (V, T). Define a directed graph with vertex set $\{C_1, \ldots, C_m\}$ and edge set (C_i, C_j) iff there is an arc $(x, y) \in H$ with $x \in C_i$ and $y \in C_j$. This graph is acyclic (since G' has no mixed cycle) and hence may be sorted topologically. We obtain an alignment that realizes T by making each component a column of the alignment and by ordering the columns as given by the topological ordering. ∎

It is now easy to formulate MWT as an integer linear program. Let $G = (V, E, H)$ be an extended alignment graph. For every $e \in E$ we have a binary variable $x_e \in \{0, 1\}$ which indicates, whether e is in the trace or not. In view of Theorem 1 the MWT-problem

$$\max \sum_{e \in E} w_e \cdot x_e \quad \text{subject to} \quad x \in P_T(G)$$

is equivalent to

$$\begin{aligned} \text{maximize} \quad & \sum_{e \in E} w_e \cdot x_e \\ \text{subject to} \quad & \sum_{e \in P \cap E} x_e \leq |E \cap P| - 1, \qquad (1) \\ & \forall \text{ critical mixed cycles } P \text{ in } G \\ & x_e \in \{0, 1\}, \ \forall e \in E \end{aligned}$$

We call the inequalities of type (1) *mixed-cycle inequalities*. All mixed cycle inequalties are valid inequalities for the trace polytope of G. In Section 3 we derive conditions under which they are facet-defining inequalities.

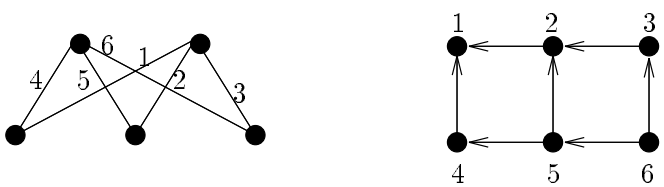

Figure 2: The $K_{2,3}$ and the corresponding pairgraph

3 The structure of the trace polytope

First we review in some well known theorems about independence systems and polyhedral combinatorics Section 3.1. In Section 3.2 we describe some basic properties of the trace polytope and give a complete description of the trace polytope for two sequences. Furthermore, we partially characterize the trace polytope for multiple sequences.

3.1 Mathematical preliminaries

In this section we give some basic definitions and results about independence systems and polyhedral combinatorics. A pair $I = (A, \mathcal{I})$ is called an *independence system on* A if $\mathcal{I}$ is a family of subsets of the finite set A with $\emptyset \in \mathcal{I}$ and the property that $F_1 \subseteq F_2$ and $F_2 \in \mathcal{I}$ implies $F_1 \in \mathcal{I}$. The members of $\mathcal{I}$ are called *independent* and those of $2^A \setminus \mathcal{I}$ *dependent* sets. As every subset of a trace also forms a trace and the empty set is a trace as well, the pair $I_T(G) = (E, \{T \mid T \subseteq E \text{ is a trace in } G\})$ forms an independence system on E.

Let $I = (A, \mathcal{I})$ be an independence system on A. A *circuit* C of I is a minimal dependent subset of A, i.e., a set $C \in 2^A \setminus \mathcal{I}$ satisfying $C \setminus \{e\} \in \mathcal{I}$ for all $e \in C$. An independence system is called *k-regular* if each of its circuits is of size k. A set $F \subseteq A$ is a *clique* of a k-regular system $(A, \mathcal{I})$ if $|F| \geq k$ and all $\binom{|F|}{k}$ k-subsets of F are circuits of $(A, \mathcal{I})$. The *incidence vector* χ^F for a subset $F \subseteq A$ is defined by setting $\chi^F_a = 1$ if $a \in F$ and setting $\chi^F_a = 0$ if $a \notin F$. Let $P_{\mathcal{I}}$ be the polyhedron associated with $(A, \mathcal{I})$.

Theorem 2 *[GP85] Let $(A, \mathcal{I})$ be an independence system and let $F = A - \bigcup \mathcal{I}$. Then the dimension of $P_{\mathcal{I}}$ is $|A| - |F|$.*

This theorem yields a method to determine whether a polytope is full-dimensional.

Theorem 3 *[HJP75] If $P_{\mathcal{I}}$ is a full-dimensional polytope associated with the independence system $(A, \mathcal{I})$, then $x_i \geq 0$ for $i = 1, \ldots, |A|$ are the only facet defining inequalities with right hand side 0. Moreover, all the nontrivial facets of $P_{\mathcal{I}}$ are defined by inequalities $a^T x \leq a_0$ with $a \geq 0$ and $a_0 > 0$.*

The above theorem for full–dimensional polytopes restricts the number of possible facet-defining inequalities.

The next theorem will prove useful in the two-sequence case of the MWT.

Theorem 4 *[NT73] Suppose $F \subseteq A$ is a maximal clique in the k-regular independence system $(A, \mathcal{I})$. Then $\sum_{e \in F} x_e \leq k - 1$ is a facet of $P_{\mathcal{I}}$.*

For $F \subseteq A$ we call $I' = (F, \mathcal{I}')$, where $\mathcal{I}' = \{\mathcal{B} \in \mathcal{I} \mid B \subseteq F\}$, the subsystem generated by F. Given a facet-defining inequality $\sum_{e \in F} a_e x_e \leq a_0$ for the subsystem $(F, \mathcal{I}')$, one may ask whether there is a facet-defining inequality

$$\sum_{e \in F} a_e x_e + \sum_{e \in E \setminus F} a_e x_e \leq a_0$$

for the independence system $(A, \mathcal{I}) \supseteq (F, \mathcal{I}')$. The process for obtaining inequalities from inequalities of subsytems is called *lifting*. For every subset $F \subseteq A$ let $P_{\mathcal{I}}(F)$ denote the polytope $\{x \in P_{\mathcal{I}} | x_e = 0 \text{ for all } e \notin F\}$.

Theorem 5 *[NT73] Let $(A, \mathcal{I})$ be an independence system, let $F \subseteq A$ and $e \notin F$. Suppose $\sum_{k \in F} a_k x_k \leq a_0$ defines a facet of $P_{\mathcal{I}}(F)$ with $a_0 > 0$. Set*

$$a_e := a_0 - \max\{\sum_{k \in F} a_k \chi^I_k | I \subseteq F, \{e\} \cup I \in \mathcal{I}\}.$$

Then $a_e x_e + \sum_{k \in F} a_k x_k \leq a_0$ *defines a facet of* $P_{\mathcal{I}}(F \cup \{e\})$.

Thus, a facet-defining inequality $a^T x \leq a_0$ for $P_{\mathcal{I}}$ can be derived from a facet-defining inequality of $P_{\mathcal{I}}(F)$ by using theorem above for all edges $e \in A \setminus F$.

3.2 The trace polytope

In this section we investigate the structure of the trace polytope. We call the set of edges which have non-zero coefficients in an inequality $c^T x \leq c_0$ the *support* of the inequality. According to the definition of circuits we observe the following:

Observation 1 *Let R be any critical mixed cycle in an extended alignment graph $G = (V, E, H)$. The incidence vectors of $R \cap E$ form a circuit of the independence system $I_T(G)$.*

Lemma 1 *Let $G = (V, E, H)$ be an extended alignment graph. $P_T(G)$ is full-dimensional and the inequalities $x_e \geq 0$, $e \in E$ are facet-defining for $P_T(G)$. Further let $e \in E$ be any edge in G. Then the inequality $x_e \leq 1$ is facet-defining iff e is not contained in any critical mixed cycle of size 2.*

Proof: Since every singleton subset $\{e\}$ of E is independent, it follows by Theorem 2 that $P_T(G)$ is full-dimensional. From Theorem 3 follows, that $x_e \geq 0$, $e \in E$ is facet-defining for $P_T(G)$ for all $e \in E$.

To prove the last statement let us assume that e is not contained in any critical mixed cycle of size 2. Then $T = \{\{e, f\} \subseteq E \mid f \in E \setminus \{e\}\} \cup \{\{e\}\}$ is a set of $|E|$ many traces whose incidence vectors are affinely independent. Thus $x_e \leq 1$ defines a facet of $P_T(G)$.

On the other hand, assume that e is contained in some critical mixed cycle of size 2. Let $f \in E \setminus e$ be the other edge in the mixed cycle. Then each incidence vector χ^B of a trace $B \subseteq E$ satisfying $\chi^B_e = 1$ has to satisfy $\chi^B_f = 0$, so $dim\{x \in P_T(G) | x_e = 1\} \leq |E| - 2$. Thus $x_e \leq 1$ is not facet-defining. ∎

We call the inequalities defined in the Lemma above the *trivial* inequalities. In the following we give a complete characterization of the trace polytope for the case of two sequences. All circuits of $I_T(G)$ (recall that $I_T(G) = (E, \{T \mid T \subseteq E \text{ is a trace in } G\})$) are of cardinality two, as a critical mixed path visits every sequence at most once. Hence the independence system is 2-regular. Theorem 4 implies that the inequalities

$$\sum_{e \in C} x_e \leq 1, \quad C \text{ is a maximal clique of } I_T(G)$$

are facet-defining for $P_T(G)$. We call these inequalities *clique inequalities*. We can now state the main result of this section.

Theorem 6 *In the two sequence case the trivial and the clique inequalities together are a complete description of the trace polytope.*

Proof: Let P be the polytope defined by the trivial and the clique inequalities. Then certainly $P_T(G) \subseteq P$. If we could prove that P is integral, i.e., has only integral vertices, we would have equality since a full-dimensional polytope has – up to a multiplicative factor – a unique description. (Unfortunately the system of inequalities defining P is not totally unimodular and hence the most convenient method of proving integrality does not apply).

For the sequel we need a more intuitive understanding of cliques in the independence system $I_T(G)$. We give a graph-theoretic characterization.

Definition 2 *Let $K_{p,q}$ be the complete bipartite graph with nodes $x_1, \ldots, x_p$ and $y_1, \ldots, y_q$. Define the irreflexiv partial order '$\prec$' on the edges of $K_{p,q}$ as follows:*

$$\{x_i, y_j\} \prec \{x_l, y_k\} \text{ iff}$$

$$(i > l \text{ and } j \leq k) \text{ or } (i = l \text{ and } j < k).$$

Observe that (V, E) is a subgraph of $K_{p,q}$ and that two edges e and f form a circuit in $I_T(G)$ iff either $e \prec f$ or $f \prec e$.

Definition 3 *Let $PG(K_{p,q})$ be the $p \times q$ directed gridgraph, s.t. the arcs go from right to left and from bottom to top. Row r, $1 \leq r \leq p$ of $PG(K_{p,q})$ contains q nodes which correspond from left to right to the q edges that go between node x_{p-r+1} and node $y_1, \ldots, y_q$ in $K_{p,q}$. We call $PG(K_{p,q})$ the pairgraph of $K_{p,q}$ (see Figure 2) and we call a node of the pairgraph essential if it corresponds to an edge in E.*

The graph $PG(K_{p,q})$ has exactly one source and one sink and there is a path from node n_2 to node n_1 in $PG(K_{p,q})$ iff $e_1 \prec e_2$ for the corresponding edges e_1, e_2 in $K_{p,q}$.

Lemma 2 *Let $p = n_1, \ldots, n_{p+q}$ be a source to sink path in $PG(K_{p,q})$ and let $e_1, \ldots, e_l$, $l \leq p + q$ be the edges in E that correspond to essential nodes in p. Then $e_1, \ldots, e_l$ is a clique of $I_T(G)$ if $l \geq 2$. Moreover, every maximal clique of $I_T(G)$ can be obtained in this way.*

Proof: For any two nodes n_i and n_j in $PG(K_{p,q})$ with $i < j$ the corresponding edges e_i and e_j are in relation $e_i \prec e_j$ and hence form a circuit of $I_T(G)$. Thus $\{e_1, \ldots, e_l\}$ is a clique of $I_T(G)$. Conversely, the set of edges in any clique of $I_T(G)$ is linearly ordered by $\prec$ and hence all maximal cliques are induced by source to sink paths in $PG(K_{p,q})$. ∎

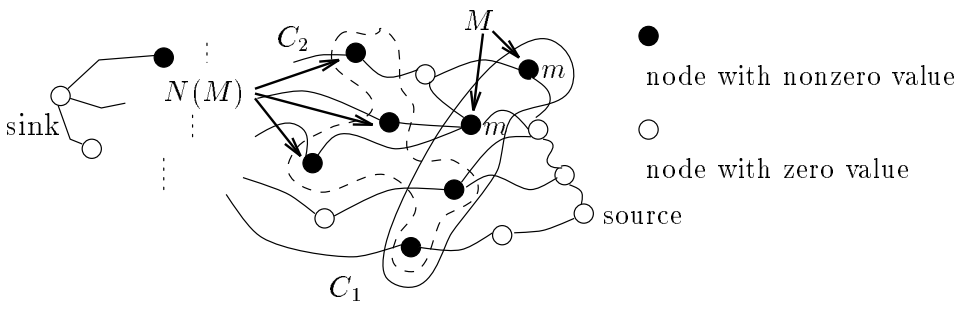

Figure 3: $PG'(K_{p,q})$ with the node sets C_1 and C_2

Lemma 3 *P is integral.*

Proof: Assume that there exists a fractional vertex f. Let w be a vector of weights such that f is the unique optimum solution of $\max\{w^T x \mid x \in P\}$; any w lying in the cone generated by supporting hyperplanes of f works.

Assign to each node n in $PG(K_{p,q})$ that corresponds to an edge e in E the value f_e and assign zero to all other nodes. Now let PG' be the subgraph of $PG(K_{p,q})$ that consists of tight paths, i.e., all source to sink paths, where the values of the nodes on the paths sum up exactly to one. Such a tight path exists since at least one of the constraints defining P must be tight in f. Let

s be the source of PG'. We construct node sets of PG', such that every source to sink path goes exactly once through each node set. Let C_1 be the set of nodes with nonzero value such that the nodes in C_1 are the first nodes with nonzero value on a source to sink path. Such a set exists as we have only tight paths in PG. Let m be the minimal value of the nodes in C_1. Clearly $m < 1$, because we assume a fractional solution. Let $M \subseteq C_1$ be the set of all nodes of C_1 with value m. Further let $N(M)$ be the set of the first nodes with nonzero value reachable from M and let $C_2 = C_1 \setminus M \cup N(M)$. The next observations follow directly (see also Figure 3):

1. There are no arcs of PG' between any two nodes of C_1 or any two nodes of $N(M)$. This would contradict the assumption that the paths are tight. For if there were an arc between node x and node y there would be one tight path which contains y and the part of the path from y to sink and another tight path which additionally would contain x. Because both x and y are positive, this cannot be.

2. The nodes in $C_1 \setminus M$ cannot have an edge to nodes in $N(M)$. Again this would contradict the assumption that the paths are tight, because the weight of the nodes in $C_1 \setminus M$ is greater than m.

From the above observations it follows that every source to sink path visits C_1 and C_2 exactly once. Define $S_1 = \sum_{n \in M} w_n$ and $S_2 = \sum_{n \in N(M)} w_n$. Here w_n is the weight (in the weight vector w) of the edge in E corresponding to n. Assume $S_1 \leq S_2$. We then decrease the value of the nodes in M by m and increase the value of the nodes in $N(M)$ by this amount. Then all tight paths are still tight, as by our invariant every tight path goes once through C_1 and once through C_2. However we have a new fractional solution which achieves at least the optimum value. This is a contradiction to the assumption that we have a unique optimal solution. Therefore the solution must be integral. The case $S_1 > S_2$ can be handled analogously. ∎

The proof of the integrality of P concludes the proof of Theorem 6. ∎

Having a complete description of the trace polytope for the two sequence case, we switch to the case of multiple sequences. For three or more sequences the Maximum Weight Trace problem is NP-hard [Kec93]. Hence we cannot expect to find a complete description of the trace polytope in this case. First we will show that the facet-defining inequalities of the two sequence case are also facet-defining in the multiple-sequence case. Theorem 5 immediately implies the following lemma:

Lemma 4 *(Zero lifting) Let $G = (V, E, H)$ be an extended alignment graph, $U \subseteq E$ and $c^T x \leq c_0$ be a facet-defining inequality for $P_T(G[E \setminus U])$ (where $G[A]$ with $A \subseteq E$ is the subgraph of G induced by A). Choose any $e \in U$ which does not form a mixed cycle with the support of $c^T x \leq x_0$. Then $c^T x \leq x_0$ defines a facet of $P_T(G[(E \setminus U) \cup \{e\}])$.*

If we apply Lemma 4 to the clique inequalities we get the following theorem:

Theorem 7 *Let G be the extended alignment graph for $k > 2$ sequences and let $P_T(G_{ij})$ be the trace polytope for the subgraph $G_{ij} = (V_{ij}, E_{ij}, H_{ij})$ induced by the edges between the sequences S_i and S_j, $1 \leq i < j \leq k$. Every facet-inducing inequality of $P_T(G_{ij})$ is also facet-inducing for $P_T(G)$.*

Proof: Lemma 4 implies that a facet-defining inequality $ax \leq b$ is also facet-defining for $P_T(G)$, because a single edge of $E \setminus E_{ij}$ cannot form a mixed cycle with the support of $ax \leq b$. ∎

We now turn our attention to the next class of inequalities, the mixed cycle inequalities. This is an important class of inequalities, because they appear in the formulation of the problem as an integer linear program. We need some more notation.

Definition 4 *Let C be a critical mixed cycle in an extended alignment graph. We call an edge $e = (v, w) \in E$ a chord of C if C_1, e and e, C_2 are critical mixed cycles where C_1 and C_2 are obtained by splitting C at v and w.*

Lemma 5 *Let $G = (V, E, H)$ be an extended alignment graph and C be a critical mixed cycle of size ℓ. Then the inequality*

$$x(C \cap E) \leq \ell - 1$$

defines a facet of $P_T(G)$ if and only if C has no chord.

Proof: Assume that C is a critical mixed cycle of size ℓ without chord. We obtain ℓ traces by removing one edge from C. The incidence vectors of these traces are linearly independent and satisfy $x(C \cap E) \leq \ell - 1$ with equality. As C has no chord we can add any edge from $E \setminus C$ to one of the above traces without introducing a mixed cycle in G. This yields another $n - \ell$ vectors that fulfill $x(C \cap E) \leq \ell - 1$ with equality.

Moreover, the incidence vectors of all edge sets constructed above are linearly independent. Thus $x(C \cap E) \leq \ell - 1$ is a facet-defining inequality.

On the other hand, if C has a chord e then each incidence vector χ^B of a trace $B \subseteq E$ satisfying $x(C \cap E) = \ell - 1$ has to satisfy $\chi^B_e = 0$, so $dim\{x \in P_T(G) | x(C \cap E) = \ell - 1\} \leq |E| - 2$. Thus $x(C \cap E) \leq \ell - 1$ is not a facet-defining inequality. ∎

The next lemma adresses the case in which we have a mixed cycle with a chord.

Lemma 6 *Let $G = (V, E, H)$ be an extended alignment graph consisting of a critical mixed cycle C of size ℓ with a chord e. Then the inequality*

$$x((C \cup \{e\}) \cap E)) \leq \ell - 1$$

defines a facet of $P_T(G)$.

Proof: ¿From Lemma 5 we know that $x(C \cap E) \leq \ell - 1$ is a facet inducing inequality for $P_{\mathcal{I}}(E \setminus \{e\})$ If we add e, we have to remove two edges from C to obtain a trace. Theorem 5 implies that the coefficient of x_e is 1 which proves the lemma. ∎

There is in fact an even stronger version of Lemma 6 for the case that there are several chords, such that every pair of the chords forms a mixed cycle of size 3 with one of the edges of C. This lemma reads as follows:

Lemma 7 *Let $G = (V, E, H)$ be an extended alignment graph consisting of a critical mixed cycle C of size ℓ with r chords $e_1, \ldots, e_r$, such that for each pair e_i, e_j, $1 \leq i < j \leq r$, there is an edge $e \in C$ that forms a mixed cycle of size 3 with that pair. Then the inequality*

$$x((C \cup \{e_1, \ldots, e_r\}) \cap E)) \leq \ell - 1$$

defines a facet of $P_T(G)$.

We call the inequalities defined in the two preceeding lemmas *mixed-cycle inequalities*, *chorded-mixed-cycle inequalities* and *ladder inequalities* respectively.

4 The branch and cut algorithm

In this section we describe the branch-and-cut algorithm for solving the integer linear program

$$\begin{aligned} \text{maximize} \quad & \sum_{e \in E} w_e \cdot x_e \\ \text{subject to} \quad & \sum_{e \in P \cap E} x_e \leq |E \cap P| - 1, \\ & \forall \text{ critical mixed cycles } P \text{ in } G \\ & x_e \in \{0,1\}, \ \forall e \in E \end{aligned} \tag{2}$$

First we review branch-and-cut algorithms and then give the details of our algorithm.

A review of branch-and-cut algorithms
One relaxes the given integer linear program by dropping the integer condition and solves the resulting linear program. If the solution $\bar{x}$ of the linear program is integral and if all mixed cycle inequalities are satisfied then we have the optimal integral solution. Otherwise one searches for a valid inequality $fx \leq f_0$ that "cuts off" the solution $\bar{x}$, i.e, $fy \leq f_0$ for all $y \in P_T(G)$ and $f\bar{x} > f_0$; $\{x \mid fx = f_0\}$ is called a *cutting plane*. The search for a cutting plane is called the separation problem. Any cutting plane found is added to the linear program and the linear program is resolved. The generation of cutting planes is repeated until either an optimal solution is found or the search for a cutting plane fails. In the second case a branch step follows: One generates two subproblems by setting one fractional variable x_e to 0 in the first subproblem and to 1 in the second subproblem and solves these subproblems recursively. This gives rise to an enumeration tree of subproblems. Lower bounds from heuristics or approximation algorithms are used to limit the size of this tree.

Type	# Seq.	# Var.	max. SCC	Approx.	Exact	Time in sec.
Prion proteins	15	26587	4642	421495	428618	8510
kinases	6	4000	333	40927	41147	905

Figure 4: Results of the branch-and-cut algorithm

A specific branch-and-bound algorithm for the MWT
In order to specialize the generic branch-and-cut algorithm we need to describe separation algorithms for our various classes of inequalities. For the ladder inequalities we use a straightforward heuristic, that is not guaranteed to find a cutting plane but worked for our test examples. For the other classes we have polynomial time separation algorithms.

First we describe how to solve the separation problem for the class of mixed-cycle inequalities. Assume the solution $\bar{x}$ of the linear program is fractional. Our problem is to find a mixed cycle P in the extended alignment graph $G = (V, E, H)$ which violates the mixed-cycle inequality $\sum_{e \in P \cap E} x_e \leq |E \cap P| - 1$. First assign the cost $1 - \bar{x}_e$ to each edge $e \in E$ and 0 to all $a \in H$. Then we compute for each node s_{ij}, $1 \leq i \leq k$, $1 \leq j < n_i$ the shortest path from s_{ij+1} to s_{ij}. If there is such a shortest path P, it must contain $l \geq 2$ edges $e_1, \ldots, e_l$ from E. If the cost of P is less than 1, i.e., $\sum_{e \in P \cap E}(1 - \bar{x}_e) < 1$, we have found a violated inequality, namely $\sum_{e \in P \cap E} \bar{x}_e > |E \cap P| - 1$, since P together with the arc (s_{ij}, s_{ij+1}) forms a mixed cycle.

Theorem 8 *The separation problem for the mixed-cycle inequalities in an extended alignment graph $G = (V, E, H)$ can be solved in polynomial time by computing at most $|H|$ shortest paths in G.*

In the separation algorithm for the class of clique inequalites we make use of the *pairgraph* $P_T(G_{ij})$ between sequence S_i and S_j for $1 \leq i < j \leq k$ (See Definition 3 and Figure 2). Again, assume the solution $\bar{x}$ of the linear program is fractional. Our problem is to find a clique C which violates the clique inequality $\sum_{e \in C \cap E} \bar{x}_e \leq 1$. Assign the cost $\bar{x}_e$ to each essential node v_e in $P_T(G_{ij})$ (essential nodes are the nodes that correspond to the edges in E). Recall that every source to sink path with more than one essential node corresponds exactly to a clique in $I_T(G_{ij})$. We compute the longest source to sink path C in $P_T(G_{ij})$. If the cost of C is greater than 1, i.e. $\sum_{e \in C \cap E} \bar{x}_e > 1$ we have found a violated clique inequality. Since $P_T(G_{ij})$ is acyclic, such a path can be found in polynomial time.

Theorem 9 *The separation problem for the clique inequalities in an extended alignment graph $G = (V, E, H)$ can be solved in polynomial time by computing a longest source to sink path in the $\binom{k}{2}$ pairgraphs $P_T(G_{ij})$ for $1 \leq i < j \leq k$, where k is the number of sequences.*

Since the trivial inequalities and the clique inequalities form a complete description of the trace polytope in the pairwise case and since we can separate both in polynomial time, we have a polynomial time algorithm for pairwise sequence alignment that is not based on dynamic programming. This gives another answer to a longstanding problem in stringology about the existence of such an algorithm [PW93].

5 Computational results

In this section we report the results of the first version of our program. We describe what software tools were used in the implementation and present two characteristic examples of sequences. We coded the algorithm in C++ using the library of efficient datatypes and algorithms LEDA [MN95]. Furtheron we used the branch-and-cut framework ABACUS by Stefan Thienel [Thi95]. We also used the multiple alignment software PRIMAL [Kec96] of Kececioglu for two tasks in our experiments: to compute the input alignment graphs for the experiments, and to obtain lower bounds on the solution value for ABACUS by running PRIMAL's initial approximate alignment phase. To generate an alignment graph, PRIMAL computes all pairwise alignments of the sequences whose score is within a fixed difference of the optimum. (As parameters for PRIMAL we chose the `blosum80` amino acid substitution matrix, shifted to make all similarity values positive and in the range 0 to 24, a gap penalty of 40, and collected all pairwise alignments that scored within 10 of optimum.) PRIMAL then superimposes all the substitution edges in these pairwise alignments to form an alignment graph. Our input is the corresponding extended alignment graph.

As an aside, we remark that PRIMAL solves an alignment problem in four phases: (1) an initial *greedy phase* forms a first approximation by merging multiple alignments according to a dynamically-computed tree; (2) next a *local search phase* polishes the initial approximation by repeatedly finding good cuts of the sequences into two groups and optimally realigning the two groups under the trace objective function; (3) a *bound precomputation phase* then determines bounds over a linear number of alignment subproblems given by all suffixes of the approximate alignment from phase (2); and (4) a final *branch-and-bound phase* finds an exact solution by solving a source-sink shortest-path problem over the dynamic programming graph. The shortest-path computation is pruned using bounds on arbitrary suffixes of the sequences; to determine such a bound, the precomputed bound for the closest subproblem is transformed into a bound for the given

suffixes by table look-up combined with a short computation. In the implementation, phases (3) and (4) are in fact interleaved: to obtain the best possible bounds for the suffix subproblems, each suffix problem is actually solved to optimality using, in turn, bounds for smaller already-solved suffixes. If the branch-and-bound phase cannot find an optimal solution within a given space limit, an approximate solution is returned by concatenating the optimal alignment for the largest-solved suffix from phase (3) onto the approximate alignment for the corresponding prefix from phase (2).

To obtain lower bounds for ABACUS we used the first phase of PRIMAL. At any point in this phase, the algorithm has a partition of the sequences; associated with each block of the partition is a multiple alignment of the sequences in the block. The phase begins with the finest possible partition: each sequence is in its own block. The phase then repeatedly considers merging two blocks. The weight of a merge is the average weight of the optimal trace alignment between the blocks: the weight of the edges in a maximum trace between the two blocks, where the relative alignment of sequences within a block is fixed, divided by the number of pairs of sequences between the two blocks. The merge of maximum weight is chosen, the two blocks are unioned, and the multiple alignment given by the merge is stored with the unioned block. The process stops when a single block remains. This is similar to the so-called "progressive alignment" method used by many multiple alignment programs, except that the merges are not determined by a spanning tree of pairwise alignments of the sequences, but rather by the average weight of optimal alignments between blocks that are effectively recomputed after each merge. Because the alignment graph is sparse and a maximum trace between a pair of strings of length n with m trace edges between them can be computed in time $O(n + m \log n)$, this procedure is fast. Given k sequences each of length n, the greedy phase can be implemented to run in $O(km \log kn)$ time and $O(k^2 n)$ space on an alignment graph with m edges. As m is often $O(k^2 n)$ in practice, this is quite efficient (subquadratic in the length of the strings once the alignment graph has been computed).

Dynamic programming approaches generally evaluate $O(2^k)$ edges for each each vertex that is visited in the dynamic programming graph. Due to this sensitivity to the number of input sequences, to our knowledge no such algorithm can align 10 or more sequences optimally, even if the sequences are very similar.

In Figure 4 we give two results of the implementation of the branch-and-cut algorithm. In the first column the type of the aligned sequences is described. The second column contains the number of sequences followed by the number of variables (i.e. edges) in the third column. As already noted in [Kec93], one can solve the trace problem separately on the strongly connected components of the extended alignment graph. Therefore not only the number of sequences but the size and the structure of the biggest strongly connected component of the extended alignment graph is a measure for the complexity of a problem instance. The fourth column gives the number of variables in the biggest strongly connected component of the extended alignment graph. The next columns show the lower bound on the solution weight from PRIMAL's heuristic, followed by the optimum weight, and the time our algorithm needed to compute the optimum alignment. The program was run on a SparcStation Ultra 1/170. We have tested our program on different datasets and present two characteristic examples. The first example is a sample of 15 prion proteins from the SWISSPROT database, whereas the second example consists of the 6 kinases sequences from [Kec93]. The actual alignments are shown in the appendix in Figure 5. The first dataset consists of relatively similar sequences. Despite the similarity, PRIMAL could not align this dataset optimally as the number of sequences is prohibitive for a dynamic programming approach. The bottleneck for such approaches is normally the space consumption which is not the case for our approach. It is not so sensitive to the number of sequences but to the structure and size of the extended alignment graph. The second example is a set of 6 sequences which are less similar than the prion proteins. PRIMAL could solve this example in about four minutes whereas our algorithm takes fifteen. This is somehow slower,

but shows that our implementation is competitive not only in space consumption but also in terms of computing time.

We hope that the implementation of separation algorithms for more classes of facet-defining inequalities will extend the range of problems we will be able to solve with our approach and speed up the computation, by yielding better upper bounds for the branch-and-cut framework ABACUS.

6 Conclusion

In summary, we have given the first formulation of a multiple-sequence alignment problem using polyhedral combinatorics. We described three classes of valid inequalities for the trace polytope and derived conditions under which these inequalites are facet-defining. We also gave separation algorithms for two of the classes, which were used in an implementation of a branch-and-cut algorithm. Our computational results show, that we are able to solve problem instances to optimality, the size of which is not suitable for a dynamic programming approach.

Acknowledgments

The authors would like to thank Stefan Thienel for installing the branch-and-cut framework ABACUS on our machines and Thomas Christof for his software PORTAF. Moreover, we would like to thank Naveen Garg for his valuable comments on Theorem 4. The fifth author would also like to thank the Max-Planck-Institut für Informatik for its gracious support while conducting this research at the Center.

References

[GP85] M. Grötschel and M.W. Padberg. *Polyhedral theory.* Wiley-Interscience, 1985.

[HJP75] P.L. Hammer, E.L. Johnson, and U.N. Peled. Facets of regular 0-1-polytopes. *Mathematical Programming*, 8:179–206, 1975.

[Kec91] J. D. Kececioglu. *Exact and approximation algorithms for DNA sequence reconstruction.* PhD thesis, U. Arizona, 1991.

[Kec93] J.D. Kececioglu. The maximum weight trace problem in multiple sequence alignment. In *Proc. 4-th Symp. Combinatorial Pattern Matching*, number 684 in Lecture Notes in Computer Science, pages 106–119. Springer-Verlag, 1993.

[Kec96] John D. Kececioglu. PRIMAL: Practical rigorous multiple alignment. Computer software, 1996.

[MN95] Kurt Mehlhorn and Stefan Näher. LEDA, a platform for combinatorial and geometric computing. *Communications of the ACM*, 38(1):96–102, 1995.

[NT73] G.L. Nemhauser and L.E. Trotter. Properties of vertex packing and independence system polyhedra. *Mathematical Programming*, 6:48–61, 1973.

[PW93] P.A. Pevzner and M.S. Waterman. Generalized sequence alignment and duality. *Advances in Applied Mathematics*, 14:139–171, 1993.

[Thi95] Stefan Thienel. *ABACUS : A Branch-And-CUt System.* PhD thesis, Universität Köln, 1995.

Appendix:

```
                mandrill   0 M---------LVLFVATWSDLGLCKKRPKP-GGWNTGGSRYPGQGSPGGNRYPPQGGGGWGQPHGGGWGQPHGGGWGQPHGGGWGQPHG
      presbytis_francoisi   0 M--ANLGCWMLVLFVATWSDLGLCKKRPKP-GGWNTGGSRYPGQGSPGGNRYPPQGGGGWGQPHGGGWGPHGGGWGQPHGGGWGQPHGG
      crab_eating_macaque   0 M--ANLGCWMLVLFVATWSDLGLCKKRPKP-GGWNTGGSRYPGQGSPGGNRYPPQGGGGWGQPHGGGWGQPHGGGWGQPHGGGWGQPHG
            green_monkey   0 M--ANLGCWMLVVFVATWSDLGLCKKRPKP-GGWNTGGSRYPGQGSPGGNRYPPQGGGGWGQPHGGGWGQPHGGGWGQPHGGGWGQ---
    brown_capped_capuchin   0 M--ANLGCWMLVLFVATWSDLGLCKKRPKP-GGWNTGGSRYPGQGSPGGNLYPPQGGG-WGQPHGGGWGQPHGGGWGQPHGGSWGQPHG
               chimpanzee   0 M--ANLGCWMLVLFVATWSDLGLCKKRPKP-GGWNTGGSRYPGQGSPGGNRYPPQGGGGWGQPHGGGWGQPHGGGWGQPHGGGWGQPHG
                orangutan   0 M--ANLGCWMLVLFVATWSNLGLCKKRPKP-GGWNTGGSRYPGQGSPGGNRYPPQGGGGWGQPHGGGWGQPHGGGWGQPHGGGWGQPHG
                  gorilla   0 M--ANLGCWMLVLFVATWSDLGLCKKRPKP-GGWNTGGSRYPGQGSPGGNRYPPQGGGGWGQPHGGGWGQPHGGGWGQPHGGGWGQPHG
                    human   0 M--ANLGCWMLVLFVATWSDLGLCKKRPKP-GGWNTGGSRYPGQGSPGGNRYPPQGGGGWGQPHGGGWGQPHGGGWGQPHGGGWGQPHG
                   bovine   0 MVKSHIGSWILVLFVAMWSDVGLCKKRPKPGGGWNTGGSRYPGQGSPGGNRYPPQGGGGWGQPHGGGWGQPHGGGWGQPHGGGWGQPHG
                    sheep   0 MVKSHIGSWILVLFVAMWSDVGLCKKRPKPGGGWNTGGSRYPGQGSPGGNRYPPQGGGGWGQPHGGGWGQPHGGGWGQPHGGGWGQPHG
                mule_deer   0 MVKSHIGSWILVLFVAMWSDVGLCKKRPKPGGGWNTGGSRYPGQGSPGGNRYPPQGGGGWGQPHGGGWGQPHGGGWGQPHGGGWGQPHG
                      rat   0 -------------------------------GGWNTGGSRYPGQGSPGGNRYPPQSGGTWGQPHGGGWGQPHGGGWGQPHGGGWGQPHG
            golden_hamster   0 M--ANLSYWLLALFVAMWTDVGLCKKRPKP-GGWNTGGSRYPGQGSPGGNRYPPQGGGTWGQPHGGGWGQPHGGGWGQPHGGGWGQPHG
                     mouse   0 M--ANLGYWLLALFVTMWTDVGLCKKRPKP-GGWNTGGSRYPGQGSPGGNRYPPQGG-TWGQPHGGGWGQPHGGSWGQPHGGSWGQPHG

                mandrill 100 --GGGTHNQWHKPNKPKTSMKHMAGAAAAGAVVGGLGGYMLGSAMSRPLIHFGNDYEDRYYRENMYRYPNQVYYRPVDQYSNQNNFVHD
      presbytis_francoisi 100 --GGGTHSQWNKPSKPKSNMKHMAGAAAAGAVVGGLGGYMLGSAMSRPLIHFGNDYEDRYYRENMYRYPNQVYYRPVDQYSNQNNFVHD
      crab_eating_macaque 100 --GGGTHNQWHKPSKPKTSMKHMAGAAAAGAVVGGLGGYMLGSAMSRPLIHFGNDYEDRYYRENMYRYPNQVYYRPVDQYSNQNNFVHD
            green_monkey 100 --GGGTHNQWHKPSKPKTSMKHMAGAAAAGAVVGGLGGYMLGSAMSRPLIHFGNDYEDRYYRENMYRYPNQVYYRPVDQYSNQNNFVHD
    brown_capped_capuchin 100 --GGGTHNQWNKPSKPKTSMKHVAGAAAAGAVVGGLGGYMLGSAMSRPLIHFGNDYEDRYYRENMYRYPNQVYYRPVDQYSNQNNFVHD
               chimpanzee 100 --GGGTHSQWNKPSKPKTNMKHMAGAAAAGAVVGGLGGYMLGSAMSRPIIHFGSDYEDRYYRENMHRYPNQVYYRPMDQYSSQNNFVHD
                orangutan 100 --GGGTHSQWNKPSKPKTNMKHMAGAAAAGAVVGGLGGYMLGSAMSRPIIHFGNDYEDRYYRENMYRYPNQVYYRPVDQYSNQNNFVHD
                  gorilla 100 --GGGTHSQWNKPSKPKTNMKHMAGAAAAGAVVGGLGGYMLGSAMSRPIIHFGSDYEDRYYRENMHRYPNQVYYRPMDQYSNQNNFVHD
                    human 100 --GGGTHSQWNKPSKPKTNMKHMAGAAAAGAVVGGLGGYMLGSAMSRPIIHFGSDYEDRYYRENMHRYPNQVYYRPMDEYSNQNNFVHD
                   bovine 100 WGQGGTHGQWNKPSKPKTNMKHVAGAAAAGAVVGGLGGYMLGSAMSRPLIHFGSDYEDRYYRENMHRYPNQVYYRPVDQYSNQNNFVHD
                    sheep 100 --QGGSHSQWNKPSKPKTNMKHVAGAAAAGAVVGGLGGYMLGSAMSRPLIHFGNDYEDRYYRENMYRYPNQVYYRPVDRYSNQNNFVHD
                mule_deer 100 --QGGTHSQWNKPSKPKTNMKHVAGAAAAGAVVGGLGGYMLGSAMNRPLIHFGNDYEDRYYRENMYRYPNQVYYRPVDQYNNQNTFVHD
                      rat 100 --GGGTHNQWNKPSKPKTNLKHVAGAAAAGAVVGGLGGYMLGSAMSRPMLHFGNDWEDRYYRENMYRYPNQVYYRPVDQYSNQNNFVHD
            golden_hamster 100 --GGGTHNQWNKPSKPKTNMKHMAGAAAAGAVVGGLGGYMLGSAMSRPMMHFGNDWEDRYYRENMNRYPNQVYYRPVDQYNNQNNFVHD
                     mouse 100 --GGGTHNQWNKPSKPKTNLKHVAGAAAAGAVVGGLGGYMLGSAMSRPMIHFGNDWEDRYYRENMYRYPNQVYYRPVDQYSNQNNFVHD

                mandrill 200 TTTTKGENFTETDVKMMERVVEQMCITQYEKESQAYYQ--RGSSMVLFSSPPVILLISFLI-----
      presbytis_francoisi 200 TTTTKGENFTETDVKMMERVVEQMCITQYEKESQAYYQ--RGSSMVFFSSPPVILLISFLIFLIVG
      crab_eating_macaque 200 TTTTKGENFTETDVKMMERVVEQMCITQYEKESQAYYQ--RGSSMVLFSSPPVILLISFLIFLIVG
            green_monkey 200 TTTTKGENFTETDVKMMERVVEQMCITQYEKESQAYYQ--RGSSMVLFSSPPVILLISFLIFLIVG
    brown_capped_capuchin 200 TTTTKGENFTETDVKMMERVVEQMCITQYERESQAYYQ--RGSSMVLFSSPPVILLISFLIFLIVG
               chimpanzee 200 TTTTKGENFTETDVKMMERVVEQMCITQYERESQAYYQ--RGSSMVLFSSPPVILLISFLIFLIVG
                orangutan 200 TTTTKGENFTETDVKMMERVVEQMCITQYERESQAYYQ--RGSSMVLFSSPPVILLISFLIFLIVG
                  gorilla 200 TTTTKGENFTETDVKMMERVVEQMCITQYERESQAYYQ--RGSSMVLFSSPPVILLISFLIFLIVG
                    human 200 TTTTKGENFTETDVKMMERVVEQMCITQYERESQAYYQ--RGSSMVLFSSPPVILLISFLIFLIVG
                   bovine 200 TTTTKGENFTETDIKMMERVVEQMCITQYQRESQAYYQ--RGASVILFSSPPVILLISFLIFLIVG
                    sheep 200 TTTTKGENFTETDIKIMERVVEQMCITQYQRESQAYYQ--RGASVILFSSPPVILLISFLIFLIVG
                mule_deer 200 TTTTKGENFTETDIKMMERVVEQMCITQYQRESQAYYQ--RGASVILFSSPPVILLISFLIFLIVG
                      rat 200 TTTTKGENFTETDVKMMERVVEQMCVTQYQKESQAYYDG-RRSSAVLFSSPPVILLISFLIFLIVG
            golden_hamster 200 TTTTKGENFTETDIKIMERVVEQMCTTQYQKESQAYYDG-RRSSAVLFSSPPVILLISFLIFLMVG
                     mouse 200 TTTTKGENFTETDVKMMERVVEQMCVTQYQKESQAYYDGRRSSSTVLFSSPPVILLISFLIFLIVG

                      v-src   0 -----GLAKDAWEIPRESLRLEAKLGQGCFGEVWMGTWN-DTTRVAI-KTLKPG-TMS-PEAFLQEAQVMKKLRHEKLVQLYAVVSE-E
                      v-yes   0 -----GLAKDAWEIPRESLRLEVKLGQGCFGEVWMGTWN-GTTKVAI-KTLKLG-TM-MPEAFLQEAQIMKKLRHDKLVPLYAVVSE-E
                      v-abl   0 TIYGVSPNYDKWEMERTDITMKHKLGGGQYGEVYEGVWKKYSLTVAV-KTLKED-TMEV-EEFLKEAAVMKEIKHPNLVQLLGVCTREP
                      v-fes   0 -VLNRAVPKDKWVLNHEDLVLGEQIGRGNFGEVFSGRLRADNTLVAV-KSCRETLPPDIKAKFLQEAKILKQYSHPNIVRLIGVCTQKQ
                      v-fps   0 -VLTRAVLKDKWVLNHEDVLLGERIGRGNFGEVFSGRLRADNTPVAV-KSCRETLPPELKAKFLQEARILKQCNHPNIVRLIGVCTQKQ
                      v-raf   0 -------SSYYWKMEASEVMLSTRIGSGSFGTVYKGKWHGD-VAVKILKVVDPT-PEQL-QAFRNEVAVLRKTRHVNILLFMGYMTK-D

                      v-src 100 GSLLDFLKG-EMGKYLRLPQLVDMAAQIASGMAYVERMNYVHRDLRAANILVGENLVCKVADFGLARLIED-NEYTARQGAK-FPIKWT
                      v-yes 100 GSLLDFLKE-GEGKFLKLPQLVDMAAQIADGMAYIERMNYIHRDLRAANILVGDNLVCKIADFGLARLIED-NEYTARQGAK-FPIKWT
                      v-abl 100 GNLLDYLRECNRQE-VSAVVLLYMATQISSAMEYLEKKNFIHRDLAARNCLVGENHLVKVADFGLSRLMTG-DTYTAHAGAK-FPIKWT
                      v-fes 100 GDFLTFLRT-EGAR-LRMKTLLQMVGDAAAGMEYLESKCCIHRDLAARNCLVTEKNVLKISDFGMSREAAD-GIYAASGGLRQVPVKWT
                      v-fps 100 GDFLSFLRS-KGPR-LKMKKLIKMMENAAAGMEYLESKHCIHRDLAARNCLVTEKNTLKISDFGMSRQEED-GVYASTGGMKQIPVKWT
                      v-raf 100 SSLYKHLHV-QETKF-QMFQLIDIARQTAQGMDYLHAKNIIHRDMKSNNIFLHEGLTVKIGDFGLATVKSRWSG-SQQVEQPTGSVLWM

                      v-src 200 RFTIKSDVWSFGILLTELTTK-GRVPYPGMVNRE-VLDQVERGY---RMPCP-PECPESLHDLMCQCWRKDPEERPTFKYLQAQLLPAC
                      v-yes 200 RFTIKSDVWSFGILLTELVT-KGRVPYPGMVNRE-VLEQVERGY---RMPCP-QGCPESLHELMKLCWKKDPDERPTFEYIQSFLEDYF
                      v-abl 200 KFSIKSDVWAFGVLLWEIAT-YGMSPYPGIDLSQ-VYELLEKDY---RMERP-EGCPEKVYELMRACWQWNPSDRPSFAEIHQAFETMF
```