A Polyhedral Approach to Sequence Alignment Problems

John D. Kececioglu [a] Hans-Peter Lenhof [b] Kurt Mehlhorn [b]
Petra Mutzel [b] Knut Reinert [b,1] Martin Vingron [c]

[a] *Department of Computer Science, The University of Georgia, Athens, GA 30602, USA*

[b] *MPI für Informatik, Im Stadtwald, 66123 Saarbrücken, Germany*

[c] *Deutsches Krebsforschungszentrum, Abt. Theoretische Bioinformatik, INF 280, D-69120 Heidelberg, Germany*

Abstract

We study two new problems in sequence alignment both from a practical and a theoretical view, using tools from combinatorial optimization to develop branch-and-cut algorithms. The Generalized Maximum Trace formulation captures several forms of multiple sequence alignment problems in a common framework, among them the original formulation of Maximum Trace. The RNA Sequence Alignment Problem captures the comparison of RNA molecules on the basis of their primary sequence and their secondary structure. Both problems have a characterization in terms of graphs which we reformulate in terms of integer linear programming. We then study the polytopes (or convex hulls of all feasible solutions) associated with the integer linear program for both problems. For each polytope we derive several classes of facet-defining inequalities and show that for some of these classes the corresponding separation problem can be solved in polynomial time. This leads to a polynomial time algorithm for pairwise sequence alignment that is not based on dynamic programming. Moreover, for multiple sequences the branch-and-cut algorithms for both sequence alignment problems are able to solve to optimality instances that are beyond the range of present dynamic programming approaches.

Key words: Computational Biology; Multiple Sequence Alignment; RNA Sequence Alignment; Combinatorial Optimization; Branch-and-Cut

---

[1] corresponding author (current address: Celera Genomics, Informatics Research, Rockville, MD 20850 , USA, email: ReinerKn@celera.com)

1 Introduction

The study of the functional relatedness of biological macromolecules heavily relies upon sequence comparison techniques. Among the most important are algorithms that align two or more sequences in order to exhibit their commonalities. It is interesting that while the diversity of alignment problems and their associated algorithms has grown tremendously since Needleman and Wunsch [28] first published their paper on two-sequence alignment in 1970, most alignment problems that have been studied have been solved by dynamic programming. This technique, while quite powerful, has the drawback that it generally yields an algorithm with a time and space complexity that is exponential in the number of sequences in the input.

We study a new approach to solving sequence alignment problems based on an area of combinatorial optimization known as *polyhedral combinatorics* [37,29]. We demonstrate how this approach when applied to the Generalized Maximum Trace (GMT) and RNA Sequence Alignment (RSA) problems yields an algorithm for each problem that is not based on dynamic programming but is known as a *branch-and-cut algorithm* [16]. Branch-and-cut algorithms combine linear programming with the branch-and-bound paradigm, and are currently the most successful algorithms for solving hard combinatorial problems such as the famous Traveling Salesman Problem [15,2].

We view as one of the contributions of our work the introduction of the polyhedral approach to the area of sequence alignment, and our experience with these relatively new techniques has helped us to appreciate some of their unique advantages. With a polyhedral approach, one formulates the alignment problem to be studied as an integer linear program; once such a formulation is found, variations of the problem can often be conveniently modeled through the addition of further constraints to the basic linear program. With dynamic programming, on the other hand, accommodating variations such as considering secondary structure in sequence alignment, as in Bafna, Muthukrishnan and Ravi [3], can cause at a minimum a significant restructuring of the basic recurrences. With the polyhedral approach, much of the code developed for the basic problem can be reused for the problem variations; for example, both the Generalized Maximum Trace and RNA Sequence Alignment problems are based on the same integer linear programming formulation, and so-called separation routines for their basic formulations are reused in the code for both problems. Finally, a polyhedral approach to a problem creates many research avenues for future investigators, as each researcher is able to build on prior theoretical work and practical software by discovering new classes of so-called facet-defining inequalities and devising new separation routines for both known and newly-discovered classes.

Graphs, traces and multiple alignment

To describe the GMT and RSA problem we first review a formulation of multiple alignment in terms of graphs introduced by Kececioglu [19] and show how to extend this formulation to model the two new problems.

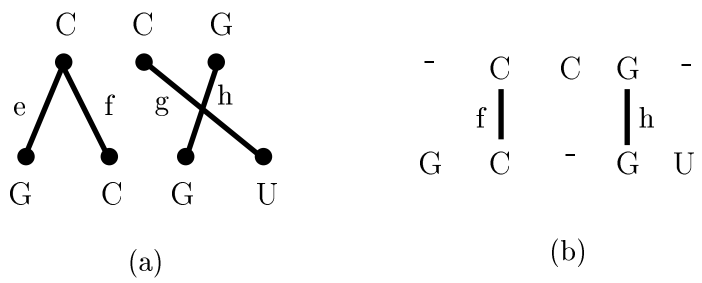


Fig. 1. (a) An alignment graph of two sequences `CGC` and `GCGU`. Edges e, f and g, h are in conflict. (b) The trace $\{f, h\}$ is realized by the alignment shown.

Let $S = \{S_1, S_2, \ldots, S_k\}$ be a set of k strings over an alphabet Σ and let $\hat{\Sigma} = \Sigma \cup \{-\}$, where "$-$" (dash) is a symbol to represent "gaps" in strings. An *alignment* of S is a set $\hat{S} = \{\hat{S}_1, \hat{S}_2, \cdots, \hat{S}_k\}$ of strings over the alphabet $\hat{\Sigma}$ that satisfies the following two properties: (1) the strings in $\hat{S}$ all have the same length, and (2) ignoring dashes, string $\hat{S}_i$ is identical to string S_i. An alignment in which each string $\hat{S}_i$ has length l can be interpreted as an array of k rows and l columns where row i corresponds to string $\hat{S}_i$. Two characters of distinct strings in S are said to be *aligned* under $\hat{S}$ if they are placed into the same column of the alignment array. We view the character positions of the k input strings in S as the vertex set V of a k-partite graph $G = (V, E)$ called the input *alignment graph.* The edge set E represents pairs of characters that one would like to have aligned in an alignment of the input strings. We say that an edge is *realized* by an alignment if the endpoints of the edge are placed into the same column of the alignment array.

The subset of E realized by an alignment $\hat{S}$ is called the *trace* of $\hat{S}$, denoted trace($\hat{S}$). Figure 1 shows an alignment graph of two strings containing four edges and an alignment that realizes two of the edges. Note that several alignments can have exactly the same trace, though such alignments differ only in their arrangement of unaligned regions.

The notion of a trace of two strings as illustrated in Figure 1 is a basic concept in sequence comparison (see for instance Sankoff and Kruskal [36] pp. 10-18) which Kececioglu [19] generalized to multiple sequence alignment with the notion of a trace of an alignment graph. The relationship between multiple alignment and multipartite graphs was also examined by Vingron and Pevzner [39] in the context of filtering pairwise dot-plots of a set of sequences.

The Generalized Maximum Trace Problem

In the *Maximum Trace Problem* (MT), introduced originally to model the final multiple alignment phase of DNA sequence assembly, every edge in the alignment graph has a positive weight representing the benefit of aligning the endpoints of the edge. The goal is to compute an alignment $\hat{S}$ whose trace has maximum weight. Kececioglu showed that MT is NP-complete [19] and developed a branch-and-bound algorithm for the problem based on dynamic programming, with worst-case time complexity $O(k^3 2^k N)$ and space complexity $O(kN)$, where $N = \prod_i |S_i|$, which is able to solve to optimality relatively small problem instances. The Maximum Trace Problem can be generalized to accommodate different scoring schemes. In the *Generalized Maximum Trace Problem* (GMT) we allow multiple edges between two vertices in the alignment graph G and we partition the edge set E into a set D of so called *blocks*. A block is a trace in which every edge is incident to nodes in the same pair of sequences. We regard a block $d \in D$ as realized if all the edges in d are realized.

Every block $d \in D$ has a weight w_d representing the benefit of realizing that block, and the weight of an alignment is the sum of the weights of the blocks it realizes. The goal is to compute an alignment $\hat{S}$ of maximum weight. Notice that this captures the construction of a multiple alignment out of local pairwise alignments.

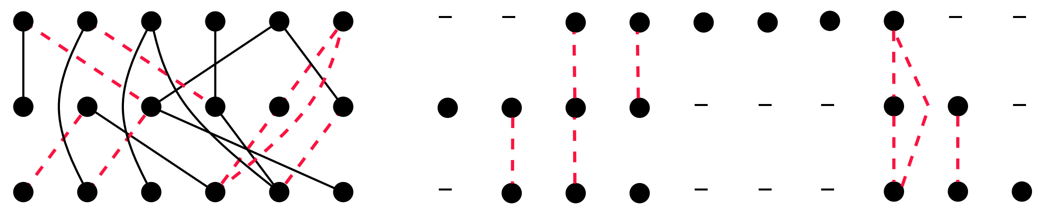


Fig. 2. An alignment graph with 18 edges. $D = \{\{e_1\}, \{e_2\}, \dots, \{e_{18}\}\}$ is a partition into singleton sets. On the right is an alignment that realizes 8 edges (and therefore 8 singleton sets).

Most commonly-used scoring schemes are based on the similarity of single pairs of characters (for instance Dayhoff et al. [5] or Hennikoff and Hennikoff [14]). This corresponds to a partition of the edges into singleton sets as in Figure 2 and is equivalent to the original MT formulation. It is worth noting that the singleton case includes as a special case the well-studied sum-of-pairs multiple alignment problem.

GMT also captures more general scoring schemes based on the similarity of pairs of whole segments of the sequences pairs (see for instance Altschul and Erickson [1], Morgenstern et al. [26], and Wilbur and Lipman [41]). To illustrate how this can be done, Figure 3 shows a partition into sets of edges that form consecutive runs of matches. Here the edges of a run from a block.

Note that the two blocks d_5 and d_1 each contains a *different* edge that runs between the same vertices. Hence, any alignment that realizes either d_1 or d_5 must match the corresponding characters.

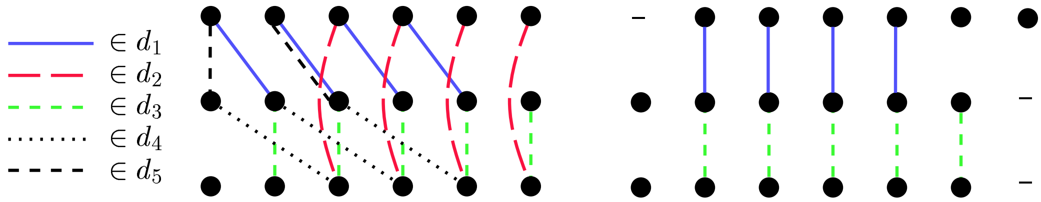


Fig. 3. An alignment graph with 18 edges. D is a partition into five blocks d_1, d_2, d_3, d_4, and d_5. On the right is an alignment that realizes 2 blocks consisting of a total of 9 edges.

The RNA Sequence Alignment Problem

The second alignment problem we address is the *RNA Sequence Alignment Problem* (RSA). An RNA molecule, unlike DNA, is a generally single-stranded nucleic acid molecule that folds in space due to the formation of hydrogen base pairs between its bases. Conventional sequence alignment algorithms can only account for the primary sequence and thus ignore structural aspects. The RSA problem deals with the comparison of RNA sequences when for one of them base pairings are known. The aim then is to align the sequences of unknown structure in such a way that the known structure carries over to the unknown sequences and as much sequence similarity is maintained as possible.

With the prediction of tRNA structure from a set of similar sequences, Levitt [23] had strikingly demonstrated that sets of similar sequences can yield convincing evidence for how an RNA molecule folds. The computational problem of considering sequence and structure of an RNA molecule simultaneously was first addressed by Sankoff [35] who proposed a dynamic programming algorithm that aligns a set of RNA sequences while at the same time predicting their common fold. Algorithms similar in spirit were proposed later on for the problem of comparing one RNA sequence to one or more of known structure. Corpet and Michot [4] align simultaneously a sequence with a number of other sequences using both primary and secondary structure. Their dynamic programming algorithm requires $O(n^5)$ running time and $O(n^4)$ space (n is the length of the sequences) and thus can handle only short sequences. Corpet and Michot propose an anchor-point heuristic to divide large alignment problems by fixed alignment regions into small subproblems that the dynamic programming algorithm can then be applied to. Bafna et al. [3] improved the dynamic programming algorithm to a running time of $O(n^4)$ which still does not make it applicable to real-life problems. Gorodkin et al. [8] iterate Sankoff's dynamic programming algorithm to find motifs among many RNA sequences."

Instead of using dynamic programming the algorithm of Waterman [40] searches for common motifs among several sequences. Eddy and Durbin [7] describe probabilistic models for measuring the secondary structure and primary sequence consensus of RNA sequence families. They present algorithms for analyzing and comparing RNA sequences as well as database search techniques. Since the basic operation in their approach is an expensive dynamic programming procedure, their algorithms cannot analyze sequences longer than 150–200 nucleotides. Notredame et al. [31] implemented a genetic algorithm for the optimization of both alignment and structure correspondence between two RNA molecules. Their procedure produces biologically good results although at the expense of considerable running time.

The input to the RSA problem can also be viewed in form of an alignment graph where for one sequence, say S_1, we additionally are given a list of base pairs, e.g. as output of a secondary structure prediction program (see Figure 4 for two sequences). The only condition on those base pairs is that a base can be involved in at most one base pair. We thus allow tertiary interactions or pseudo knots. The goal is to compute an (optimal) alignment that maximizes

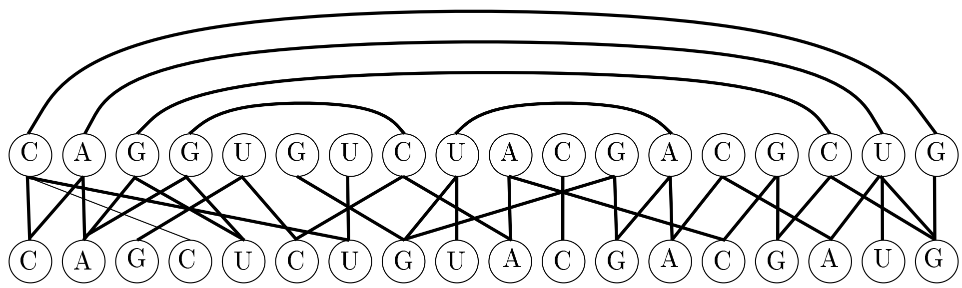


Fig. 4. Alignment graph with base pairs of the first sequence.

sequence and structure consensus simultaneously. To be more precise, the score that is optimized is a weighted sum of a sequence alignment score and a base pairing score which measures the quantity and quality of the base pairs of the sequences that are preserved by the sequence alignment.

Guide to the paper

In this paper we apply methods from polyhedral combinatorics to the GMT and the RSA problem. We formulate both problems in terms of an integer linear program and derive several classes of facet-defining inequalities for the associated polytope.

We show that for some of these classes the corresponding separation problem can be solved in polynomial time. This leads to another polynomial time algorithm for pairwise alignment that is not based on dynamic programming techniques (see Pevzner and Waterman [33] for a thorough presentation of primal-dual approach for a number of sequence alignment problems that is also not based on dynamic programming).

It turns out that our branch-and-cut algorithms can solve problem instances, the size of which is not tractable for dynamic programming based approaches.

In Section 2 we give a graph-theoretic characterization of traces which in turn is used to formulate the GMT and RSA problem as an ILP. In Section 3 we study the structure of these polytopes and present classes of facet-defining inequalities. In Section 4 we sketch the branch-and-cut algorithms. The results of our computational experiments are given in Section 5. Finally, we discuss our results in Section 6.

The reader who is more interested in the practical results of our work might wish to skip Section 3 except for the introductory part and rather go on to Section 4 in which the algorithms are described. As a final note the corresponding author would like to point out that a more detailed discussion of the problems in this paper can be found in [34].

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 the GMT and RSA problem as integer linear programs. We need some more notations.

A *mixed graph* is a tuple $G = (V, E, A)$, where V is a set of vertices, E is a set of edges and A is a set of arcs. A *path* in a mixed graph is an alternating sequence $v_1, e_1, v_2, e_2, \dots, v_k$ of vertices and arcs or edges such that either $e_i = \{v_i, v_{i+1}\} \in E$ or $e_i = (v_i, v_{i+1}) \in A$, for all i, $1 \leq i < k$. A path is called a *mixed path* if it contains at least one arc in A and one edge in E. A mixed path is called a *mixed cycle* iff the first and the last vertex on the path is the same. Since a mixed path P (or a mixed cycle C) is determined by the set of arcs and edges in P (respectively in C), we often identify paths and cycles by their set of edges and arcs.

The *length of a mixed path P (cycle C)* is the number of edges and arcs it contains. The *size* of a mixed path P (cycle C) is the number of edges in E it contains.

Note that all above notations and definitions hold also for *multigraphs* that are graphs in which we allow multiple edges (arcs) between pairs of nodes.

For our problems it is convenient to extend the alignment graph to a mixed graph (V, E, H) by adding a set of directed "horizontal" arcs

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

where s_{ij} is the vertex that corresponds to letter j in sequence i. We call this graph the *extended alignment graph (EAG)*. We use S_i to denote all vertices s_{ij} with $1 \leq j \leq n_i$. If $e \in E$ is an edge between a vertex in S_i and a vertex in S_j with $i < j$ we denote by $start(e)$ the index of the letter of S_i where the edge e starts and by $end(e)$ the index of the letter of S_j where the edge e ends. For two alignment edges e and f $(e \neq f)$ we define the irreflexive, transitive partial order '$\prec$' as follows:

Definition 1 *Two alignment edges $e, f \in E$ are in relation $e \prec f$ if and only if they both start in S_i and end in S_j for some $i < j$ and if*

$$(start(e) > start(f) \text{ and } end(e) \leq end(f)) \text{ or}$$

$$(start(e) = start(f) \text{ and } end(e) < end(f)) .$$

Two alignment edges e and f are in conflict *if either $e \prec f$ or $f \prec e$ (i.e. they form a mixed cycle of size two).*

For example in Figure 1 the relations $e \prec f$ and $h \prec g$ hold. We call a mixed cycle R in G *critical* if for all i, $1 \leq i \leq k$, all vertices in $R \cap S_i$ occur consecutively in R.

The extended alignment graph gives us a simple way of testing whether its edge set represents a trace or not by simply looking for a critical mixed cycle in it which we prove in the following theorem.

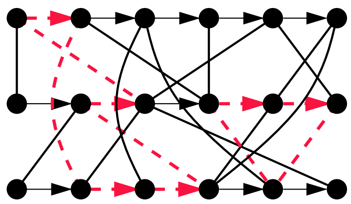


Fig. 5. Extended alignment graph. Two critical mixed cycles (dashed) are shown

Theorem 2 *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 A 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 mixed cycle and then construct an alignment with trace T. Assume first that G' contains a mixed cycle. Consider a smallest size 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) (note that each connected component contains at most one vertex from each sequence). Define a directed graph with vertex set $\{C_1, \ldots, C_m\}$ and arc set $\{(C_i, C_j) \mid \text{ there is an arc } (x, y) \in H \text{ with } x \in C_i \text{ and } y \in C_j \}$. 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. □

2.1 A characterization of the GMT problem as ILP

In this section we give the ILP formulation the GMT. Recall that in the GMT formulation we are given an EAG $G = (V, E, H)$ and a partition D of blocks. Using Theorem 2 we can formulate the GMT problem as follows:

Generalized Maximum Trace Problem:
Given an EAG $G = (V, E, H)$ and a partition D into blocks with weights w_d ($\forall d \in D$). Find a set $M \subseteq D$ of maximum weight such that $\bigcup_{d \in M} d$ does not induce a critical mixed cycle on G.

Note that the only conditions on D are 1) every $d \in D$ is a trace between a pair of sequences, i.e. it does not contain two conflicting edges, and 2) D is a partition, i.e. any edge in E is contained in exactly one block of D. A feasible set over D is a set M of blocks such that the union $U = \bigcup_{d \in M} d$ does not induce a mixed cycle on G. In that case U is a trace according to Theorem 2. Let $\mathcal{T} := \{M \subset D \mid \bigcup_{d \in M} d \text{ is a trace}\}$ be the set of all feasible solutions. We define the *GMT polytope* as the convex hull of all incidence vectors of D that are feasible, i.e.

$$P_{\mathcal{T}}(G) := conv\{\chi^M \in \{0,1\}^{|D|} \mid M \in \mathcal{T}\}$$

where the *incidence vector* χ^F for a subset $F \subseteq D$ is defined by setting $\chi^F_d = 1$ if $d \in F$ and setting $\chi^F_d = 0$ if $d \notin F$. For reasons of clarification we speak in the singleton case also of the *MT polytope*.

The surjective function $v : E \cup H \to D \cup \emptyset$ with

$$v(e) = \begin{cases} d & \text{if } e \in d, \\ \emptyset & \text{if } e \in H \end{cases}$$

maps each edge $e \in E$ to the block $d \in D$ in which e is contained. As a shorthand we write $v(C) := \{v(e)|e \in C\}$, $C \subseteq E \cup H$.

It is now easy to formulate GMT as an integer linear program. For every $d \in D$ we have a binary variable $x_d \in \{0, 1\}$ which indicates whether d is in the solution or not. In view of Theorem 2 the GMT-problem

$$\max \sum_{d \in D} w_d \cdot x_d \ \text{ subject to } \ x \in P_{\mathcal{T}}(G)$$

is equivalent to

$$\begin{aligned} \text{maximize} \quad & \sum_{d \in D} w_d \cdot x_d \\ \text{subject to} \quad & \sum_{d \in v(C)} x_d \le |v(C)| - 1, \qquad (1) \\ & \forall \text{ critical mixed cycles } C \text{ in } G \\ & x_d \in \{0, 1\}, \ \forall d \in D \end{aligned}$$

We call the inequalities of type (1) *mixed-cycle inequalities*. All mixed cycle inequalities are valid inequalities for the GMT polytope of G. In Section 3.2 we derive conditions under which they are facet-defining.

2.2 A characterization of the RSA problem as ILP

The input for the RSA problem can also be modeled as an extended alignment graph G with an additional edge set B, i.e. $G = (V, E, H, B)$. We call G the *RSA graph*. The edges in B represent possible base pairings between residues of the sequences as derived from the given base pairs for sequence S_1. We call the edges in B *base pair edges* (see also Figure 6 for two sequences). There is a base pair edge $b_{ij} \in B$, if the two alignment edges e_i and e_j $(i < j)$ have the following properties:

- e_i and e_j start in S_1 and end in the same sequence S_l, $2 \le l \le k$.
- The two alignment edges e_i and e_j are not in conflict.
- There is a base pair given between the bases *start*(e_i) and *start*(e_j) in sequence S_1.
- The bases *end*(e_i) and *end*(e_j) of S_l are complementary (and thus able to form a base pair).

Note that b_{ij} is unique and that the definition of base pair edges can also incorporate other constraints like for example a minimum distance of $start(e_i)$ and $start(e_j)$ (resp. $end(e_i)$ and $end(e_j)$). Note further that the base pairs of the first sequence are only used to define the base pair edges in B. They are not part of the RSA graph. We call e_i and e_j the *generating* alignment edges

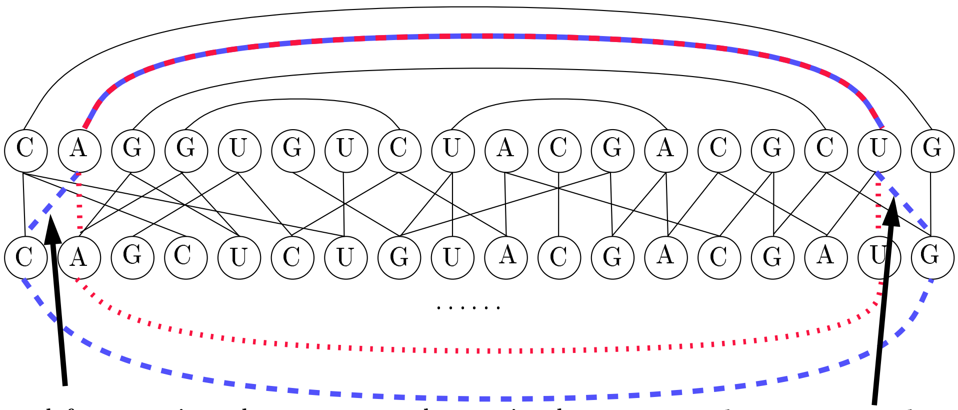


Fig. 6. An RSA graph with the base pairs of the first sequence (shown without arcs in H).

of the base pair edge b_{ij}. Each base pair edge b_{ij} has a positive weight w_{ij} that could, for example, measure the "energy" or the number of the hydrogen bonds of the base pair. We say that an alignment realizes a base pair edge b_{ij} if it realizes its two generating alignment edges e_i and e_j. Two base pair edges are in conflict if there is a conflict between their generating alignment edges. A base pair edge is in conflict with an alignment edge if the alignment edge is in conflict with one of the generating alignment edges of the base pair. A subset $A = \{E' \cup B' \mid E' \subseteq E, B' \subseteq B\}$ of $E \cup B$ is called a *RSA alignment* of G if

- E' is a trace of G and
- the base pair edges in B' are realized by the alignment edges in E'.

Note that B' can be any subset of the base pair edges realized by E', e.g. $B' = \emptyset$ is allowed. The weight $w(A)$ of the RSA alignment A is the sum of the weights of the alignment edges in E' and the weights of the base pair edges in B'. The RSA problem is then defined as follows:

The RNA Sequence Alignment Problem

Given an RSA graph $G = (V, E, H, B)$,
compute the RSA alignment A with maximal weight.

Every RSA alignment $A = E' \cup B'$ can be represented by an $|E \cup B|$-dimensional incidence-vector χ^A. Let $\mathcal{R} := \{A \subseteq E \cup B \mid A \text{ is an RSA alignment of } G\}$ be the set of all feasible solutions. We define the *RSA polytope* of G as the convex hull of all incidence vectors of RSA alignments, i.e.

$$P_{\mathcal{R}}(G) := conv\{\chi^A \in \{0,1\}^{|E \cup B|} \mid A \in \mathcal{R}\}$$

It now is easy to formulate the RSA problem as an integer linear program. The variable x_i represents the alignment edge e_i and the variable x_{ij} the base pair edge b_{ij}. The problem

$$\max \sum_{a \in E \cup B} w_a \cdot x_a \quad \text{subject to} \quad x \in P_{\mathcal{R}}(G)$$

can then be formulated as follows:

$$\begin{aligned}
\text{maximize} \quad & \sum_{e_i \in E} w_i \cdot x_i + \sum_{b_{ij} \in B} w_{ij} \cdot x_{ij} \\
\text{subject to} \quad & \sum_{e \in C} x_e \leq |C \cap E| - 1, && (2) \\
& \forall \text{ critical mixed cycles } C \text{ in } G \\
& x_{ij} \leq x_i && (3) \\
& x_{ij} \leq x_j \\
& \forall \text{ base pair variables } x_{ij} \\
& x_i, x_{ij} \in \{0,1\} && (4)
\end{aligned}$$

The mixed cycle inequalities (2) ensure that the chosen alignment edges form a trace in G. The second class of constraints (3) guarantees that a base pair edge is only realized if its generating alignment edges are realized.

3 Studying the polytopes

We have defined two optimization problems over the GMT polytope and the RSA polytope, respectively, so that the set of vertices of each polytope corresponds to the set of feasible solutions. Due to Farkas, Weyl and Minkowski [37], for every polytope P there exists also a description in form of linear inequalities. The aim of polyhedral combinatorics is to study these descriptions. Of particular interest are the *dimension* and special *faces* of the polytope. The dimension of a polytope is the maximum number of affinely independent points corresponding to vertices of the polytope and a *face* of a polytope P is a subset $F \subseteq P$ such that there exists an inequality $a^T x \leq a_0$ with $P \subseteq \{x \in {}^E \mid a^T x \leq a_0\}$ and $F = \{x \in P \mid a^T x = a_0\}$. Especially important faces of a polyhedron P are the ones of dimension 0, the *vertices*, and the ones of dimension $dim(P) - 1$, the *facets*, which can be found in a minimal description of P in terms of linear inequalities.

Once a *complete* description of a polytope P is known, the associated optimization problem over P can be solved via linear programming.

However, it is unlikely to find a complete description of a polytope associated with a NP–hard combinatorial optimization problem [18]. But experience has shown that partial descriptions already suffice in order to solve given instances to provable optimality.

Concerning the MT polytope for two sequences, we succeeded in finding the complete description (see Section 3.2). For both the GMT and the RSA polytope we present a partial description.

Any partial description defines a relaxation of the original optimization problem and hence provides a lower bound in the case of a maximization problem. Often the lower bounds provided via polyhedral studies are much better than the lower bounds obtained by other methods. Therefore it is useful to combine polyhedral techniques with branch-and-bound techniques; this combination is called *branch-and-cut* (see Section 4).

The performance of a branch-and-cut algorithm crucially depends on the description of the associated polytope. As mentioned above, in a "good" (minimal) description of a polytope, only the so-called *facet-defining* inequalities are present. Hence it is worthwhile investigating the polytope searching for such inequalities.

In Section 3.1 we review some well known theorems about independence systems and polyhedral combinatorics. In Section 3.2 we give a complete description of for the (G)MT polytope in the two-sequence case and partially characterize it for multiple sequences. Finally we characterize the RSA polytope in Section 3.3.

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.

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 $(A, \mathcal{I})$. Let $P_{\mathcal{I}}$ be the polyhedron associated with $(A, \mathcal{I})$. Then the following well known theorems hold.

Theorem 3 *[11] 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 4 *[13] If $P_{\mathcal{I}}$ is a full-dimensional polytope associated with the independence system $(A, \mathcal{I})$, then $x_i \geq 0$ for $i = 1, \dots, |A|$ 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 GMT.

Theorem 5 *[30] 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 $\mathcal{I}' = \{C \in \mathcal{I} | \ C \subseteq F\}$, 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

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

for the independence system $(A, \mathcal{I}) \supseteq (F, \mathcal{I}')$. The process for obtaining inequalities from inequalities of subsystems 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 6 *[30] 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*

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

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 the above theorem for all elements $a \in A \setminus F$. In the case that $a_e = 0$ for all $e \in A \setminus F$, we also say that the inequality $a^T x \leq a_0$ has been derived by *zero-lifting* from an inequality on its subsystems.

The next theorem also holds when the investigated problem does not form an independence system.

Theorem 7 *[30] Let $P \subseteq {}^d$ be a full-dimensional polyhedron. If F is a (nonempty) face of P then the following assertions are equivalent.*

(1) *F is a facet of P.*
(2) $\dim(F) = dim(P) - 1$ *where $dim(P)$ is the maximum number of affinely independent points in P minus one.*
(3) *There exists a valid inequality $c^T x \leq c_0$ with respect to P with the following three properties:*
   (a) $F = \{x \in P | c^T x = c_0\}$
   (b) *There exists a vector $\hat{x} \in P$ such that $c^T \hat{x} < c_0$.*
   (c) *If $a^T x \leq a_0$ is a valid inequality for P such that $F \subseteq \bar{F} = \{x \in P | a^T x = a_0\}$ then there exists a number $\lambda \in$ such that $a^T = \lambda \cdot c^T$ and $a_0 = \lambda \cdot c_0$.*

Assertions 2 and 3 provide the two basic methods to prove that a given inequality $c^T x \leq c_0$ is facet-defining for a polyhedron P. The first method, called the direct method, consists of exhibiting a set of $d = dim(P)$ vectors $x_1, \ldots, x_d$ satisfying $c^T x_i = c_0$ and showing that these vectors are affinely independent. The indirect method is the following: We assume that $\{x \mid c^T x = c_0\} \subseteq \{x \mid a^T x = a_0\}$ for some facet-defining inequality $a^T x \leq a_0$ and prove that there exists a $\lambda > 0$ such that $a^T = \lambda \cdot c^T$ and $a_0 = \lambda \cdot c_0$.

A matrix A is called *totally unimodular* if each subdeterminant of A is 0,+1 or -1. In particular each entry in a totally unimodular matrix is 0,+1 or -1. A link between total unimodularity and integer linear programming is given by the following fundamental theorem.

Theorem 8 *[37] Let A be a totally unimodular matrix and let b be an integral vector. Then the polyhedron $P := \{x \mid Ax \leq b\}$ is integral, i.e. it has only integer-valued vertices.*

Thus proving a constraint matrix to be totally unimodular yields an elegant way to prove the integrality of the associated polytope. The following theorem gives a useful characterization of total unimodularity.

Theorem 9 *[37] Let A be a matrix with entries* 0,+1 *or* -1. *Then the following are equivalent:*

(1) *A is totally unimodular, i.e. each square submatrix of A has determinant* 0,+1 *or* -1;
(2) *each collection of columns of A can be split into two parts so that the sum of the columns in one part minus the sum of the columns in the other part is a vector with entries only* 0,+1 *or* -1.

In this section we investigate the structure of the GMT polytope. First we consider the case of two sequences and then the case of multiple sequences. Recall that D is the set of all blocks that might be realized by an alignment. Since every subset of a feasible set of blocks is also feasible and the empty set is feasible as well, the pair $I_{\mathcal{T}}(G) = (D, \mathcal{T})$ forms an independence system on D. We call the set of blocks 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 $G = (V, E, H)$. Then the incidence vectors of $v(R)$ form a circuit of the independence system $I_{\mathcal{T}}(G)$.*

Lemma 10 *Let $G = (V, E, H)$ be an extended alignment graph. $P_{\mathcal{T}}(G)$ is full-dimensional and the inequalities $x_d \geq 0$, $d \in D$ are facet-defining for $P_{\mathcal{T}}(G)$. Further let d be any block in D. Then the inequality $x_d \leq 1$ is facet-defining iff no edge of d is in conflict with another edge.*

Proof. Since every $d \in D$ is independent, it follows by Theorem 3 that $P_{\mathcal{T}}(G)$ is full-dimensional. From Theorem 4 follows that $x_d \geq 0$, $d \in D$ is facet-defining for $P_{\mathcal{T}}(G)$ for all $d \in D$.

To prove the last statement let us assume that no edge of d is in conflict with another edge. Then $A = \{\{s, d\} \subseteq D \mid s \in D \setminus d\} \cup \{\{d\}\}$ is a set of $|D|$ many feasible solutions whose incidence vectors are affinely independent. Thus $x_d \leq 1$ defines a facet of $P_{\mathcal{T}}(G)$.

On the other hand, assume that there is an edge $e \in d$ in conflict with another edge f. Then each incidence vector χ^M of a feasible solution $M \subseteq D$ satisfying $\chi^M_d = 1$ has to satisfy $\chi^M_{v(f)} = 0$, so $dim\{x \in P_{\mathcal{T}}(G) | x_d = 1\} \leq |D| - 2$. Thus $x_d \leq 1$ is not facet-defining. $\square$

We call the inequalities defined in the lemma above the *trivial* inequalities for the GMT problem.

For two sequences all circuits of $I_{\mathcal{T}}(G)$ (recall that $I_{\mathcal{T}}(G) = (D, \mathcal{T})$) are of cardinality two because a critical mixed cycle visits every sequence at most once (see also Theorem 2). Hence the independence system is 2-regular, which means that in a clique of $I_{\mathcal{T}}$ each pair of blocks contains edges e_1, e_2 with

$e_1 \prec e_2$. Theorem 5 implies that the inequalities

$$\sum_{d \in C} x_d \leq 1, \quad C \text{ is a maximal clique of } I_{\mathcal{T}}(G)$$

are facet-defining for $P_{\mathcal{T}}(G)$. We call these inequalities *clique inequalities.*

It is known that the two-sequence case of MT can be reduced to the problem of computing the heaviest increasing subsequence of an integer sequence. Therefore the question arises whether the trivial and clique inequalities already give a complete description of the (G)MT polytope. In fact it can be shown that for the MT the clique inequalities together with the trivial inequalities build a complete description of the MT polytope.

To prove this we need a more intuitive understanding of cliques in the independence system $I_{\mathcal{T}}(G)$ for the MT problem. Observe that (V, E) is a subgraph of the complete bipartite graph $K_{p,q}$ with nodes $x_1, \ldots, x_p$ and $y_1, \ldots, y_q$.

Definition 11 *Let $PG(K_{p,q})$ be the $p \times q$ directed grid graph, such that 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_r 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 7) 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 $K_{p,q}$. For example in Figure 7 e_3 is the source and $e_5 \prec e_3$ because there is a path from e_3 to e_5. Recall that the two singleton sets $\{e_1\}$ and $\{e_2\}$ form a circuit in $I_{\mathcal{T}}(G)$ iff e_1 and e_2 are in conflict, i.e. if either $e_1 \prec e_2$ or $e_2 \prec e_1$.

Lemma 12 *Let $p = n_1, \ldots, n_{p+q}$ be a source-to-sink path (n_1 is the source) 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 $C := \{\{e_1\}, \ldots, \{e_l\}\}$ is a clique of $I_{\mathcal{T}}(G)$ if $|C| \geq 2$. Moreover, every maximal clique of $I_{\mathcal{T}}(G)$ is represented by a source-to-sink path in $PG(K_{p,q})$.*

Proof. For any two nodes n_i and n_j in p with $i > j$ the corresponding edges e_i and e_j are in relation $e_i \prec e_j$ and hence $\{e_i\}$ and $\{e_j\}$ form a circuit of $I_{\mathcal{T}}(G)$. Thus $\{\{e_1\}, \ldots, \{e_l\}\}$ is a clique of $I_{\mathcal{T}}(G)$. Conversely, the edges in the singleton sets of any clique $C = \{\{e_1\}, \ldots, \{e_l\}\}$ of $I_{\mathcal{T}}(G)$ can be totally ordered with respect to $\prec$ because $\prec$ is transitive and the edges in any two singleton sets of C are in relation $\prec$. We assume w.l.o.g. $e_1 \prec e_2 \prec \cdots \prec e_l$. As noted above that means that there exists a path from n_i to n_{i+1} for

$1 \leq i < l$. This implies the existence of a source-to-sink path containing the essential nodes $n_1, \ldots, n_l$. On this path cannot lie another essential node, because otherwise C would not be maximal. $\square$

Fig. 7. The $K_{2,3}$ and the corresponding pairgraph

The pairgraph is a powerful data structure. It represents $\frac{(2n-2)!}{(n-1)!^2} = \Omega(2^n)$ clique inequalities where n is the number of edges in $K_{p,q}$. However, if G is sparse it is unnecessary to store nonessential nodes. In this case the space consumption can be reduced using a *sparse pairgraph*. In a sparse pairgraph there are only essential nodes and paths consisting of nonessential nodes are replaced by arcs (see also Figure 8). Normally the space consumption of a sparse pairgraph is linear in the number of edges in G, although there are examples, in which the sparse pairgraph needs more space, because of a high number of arcs. To prove the integrality of the MT polytope we could prove that the constraint matrix formed by the trivial and clique inequalities is totally unimodular. Unfortunately this is not the case. To show this we have to identify a set of columns in the constraint matrix with the property, that there is no partition of the set in sets S_+ and S_- such that the sum of the column vectors in S_+ minus the sum of the column vectors in S_- yields a vector with entries 1,0 or -1. Indeed, such an example can be found.

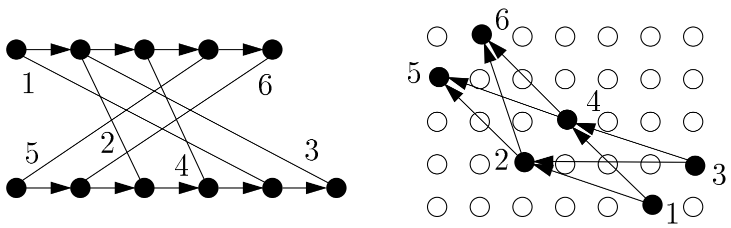


Fig. 8. Instance of MT with corresponding sparse pairgraph (in black) that has no totally unimodular constraint matrix (white nodes are nonessential).

Lemma 13 *There are instances of the MT problem for two sequences such that the constraint matrix formed by the trivial and clique inequalities is not totally unimodular.*

Proof. The proof is conducted by exhibiting an instance of the MT problem which gives rise to a constraint matrix that is not totally unimodular. Figure 8 shows an instance of MT and the corresponding sparse pairgraph. It is easy to verify that the matrix C in Figure 9 gives the coefficients for all clique

$$C = \begin{pmatrix} 1\,1\,0\,0\,1\,0 \\ 1\,1\,0\,0\,0\,1 \\ 1\,0\,0\,1\,1\,0 \\ 1\,0\,0\,1\,0\,1 \\ 0\,1\,1\,0\,1\,0 \\ 0\,1\,1\,0\,0\,1 \\ 0\,0\,1\,1\,1\,0 \\ 0\,0\,1\,1\,0\,1 \end{pmatrix}$$

Fig. 9. Coefficient matrix C

inequalities: If we choose the columns 2, 3 and 6 there are exactly three ways to partition them w.l.o.g. into S_+ and S_-, namely

(1) $S_+ = \{2\}$ and $S_- = \{3, 6\}$.
(2) $S_+ = \{3\}$ and $S_- = \{2, 6\}$.
(3) $S_+ = \{6\}$ and $S_- = \{2, 3\}$.

Summing the vectors in S_+ and subtracting the vectors in S_- yields the following three vectors:

(1) $(1, 0, 0, -1, 0, -1, -1, -2)^T$
(2) $(-1, -2, 0, -1, 0, -1, 1, 0)^T$
(3) $(-1, 0, 0, 1, -2, -1, -1, 0)^T$

Each of these vectors has an entry different from 0,1 and -1. According to Theorem 9 this is not possible for a totally unimodular matrix. $\square$

Deprived of this convenient way of showing that the trivial and cliques inequalities form a complete description of the MT polytope we try a more direct way by using the pairgraph in a constructive proof.

Theorem 14 *In the two-sequence case of the MT problem the trivial and the clique inequalities together form a complete description of the MT polytope.*

Proof. Let P be the polytope defined by the trivial and the clique inequalities. Then certainly $P_{\mathcal{T}}(G) \subseteq P$. 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.

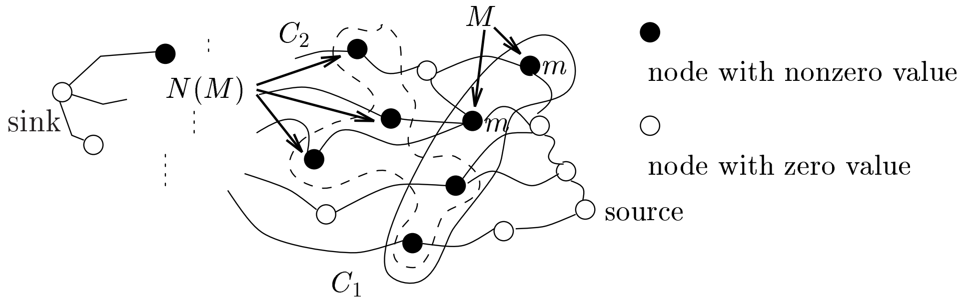


Fig. 10. $PG'(K_{p,q})$ with the node sets C_1 and C_2

Lemma 15 *P is integral.*

Proof. Assume that P has a fractional vertex $\hat{x}$. Let w be a vector of weights such that $\hat{x}$ 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 $\hat{x}$ works.

Assign to each node n_e in $PG(K_{p,q})$ that corresponds to an edge e in E the value $\hat{x}_{\{e\}}$ of the singleton set $\{e\}$ 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. the subgraph that is induced by the edges that are contained in some source-to-sink path, where the values of the nodes on that path sum up exactly to one. Note that such a tight path exists because otherwise $\hat{x}$ would not be optimal. Moreover, all paths in PG' are tight because for any node n_e in PG' it holds that all paths from the source to n_e have the same value and all paths from n_e to the sink have the same value. This follows because otherwise there would exist at least one source-to-sink path that has a value greater than one. This in turn would imply a violated clique inequality which would contradict the feasibility of $\hat{x}$.

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 $C_2 = (C_1 \setminus M) \cup N(M)$. This leads to the following observations (see also Figure 10):

(1) There is no arc from n_e to n_f between any two nodes $n_e, n_f \in C_1$. Otherwise there would be two paths with different value from s to n_f, one with value $x_{\{f\}}$ and one with value $x_{\{f\}} + x_{\{e\}}$ which is impossible.
(2) There is no arc from n_e to n_f between any two nodes $n_e, n_f \in N(M)$. Otherwise there would be two paths with different values from s to n_f,

namely one with value $m + x_{\{f\}}$ and one with value $m + x_{\{f\}} + x_{\{e\}}$.

(3) The nodes in $C_1 \setminus M$ cannot have an edge to a node in $N(M)$. Again, this would result in two paths of different value from the source to an edge in $N(M)$.

From the above observations it follows that every source-to-sink path visits C_1 and C_2 exactly once. Define $S_1 = \sum_{\{e|n_e \in M\}} w_{\{e\}}$ and $S_2 = \sum_{\{e|n_e \in N(M)\}} w_{\{e\}}$. Here $w_{\{e\}}$ is the weight (in the weight vector w) of the singleton set $\{e\}$. 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 weight. 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. $\square$

The proof of the integrality of P concludes the proof of Theorem 14. $\square$

It should be noted that the above lemma can be proved in a different way which we shortly sketch.

Pevzner and Waterman [33] showed that the weight of a maximum trace equals the size of a minimal clique cover which is proven using Dilworths theorem. In other words, if $Ax \leq b$ is the system of the clique inequalities from the lemma above then for each integral vector c holds $\max\{cx \mid Ax \leq b\} = \min\{y^T b \mid y^T A = c, y \geq 0\}$. The equality of the solution of the primal and dual problem for each integral c implies that $Ax \leq b$ is *totally dual integral*. Together with the integrality of b this implies that P is integral (see for example [37]).

Hence the branch-and-cut algorithm as well as Pevzner and Waterman's method provides methods for computing optimal pairwise alignments that are not based on dynamic programming.

It is not clear whether the clique inequalities and the trivial inequalities always form a complete description of the GMT polytope.

We now switch to the case of multiple sequences. For more than three sequences Kececioglu [19] showed that the MT is NP-hard. Hence we cannot expect to find a complete description of the GMT polytope in this case.

First we will show that the facet-defining inequalities of the two-sequence case of the GMT are also facet-defining in the multiple-sequence case. If an inequality is facet-defining for a polytope P_1 associated with some subgraph G_1 of the EAG, then it is still facet-defining for a polytope P_2, if the EAG G_2

associated with P_2 is augmented only by edges that do not induce a mixed cycle with edges in G_1. An application of the lifting theorem (see Theorem 6) yields that the coefficients of all blocks whose edges do not induce a mixed cycle with the edges in G_1 are zero. This reads formally as follows:

Lemma 16 *(Zero lifting) Let $G = (V, E, H)$ be an extended alignment graph, $U \subseteq D$ and $c^T x \leq c_0$ be a facet-defining inequality for $P_{\mathcal{T}}(G[E \setminus \bigcup_{s \in U} s])$ (where $G[A]$ with $A \subseteq E$ is the subgraph of G induced by A). Choose any $d \in U$ whose edges do not induce a mixed cycle with an edge in the support of $c^T x \leq x_0$. Then $c^T x \leq x_0$ defines a facet of $P_{\mathcal{T}}(G[(E \setminus \bigcup_{s \in U} s) \cup d])$.*

If we apply Lemma 16 to the clique inequalities we get the following theorem:

Theorem 17 *Let $G = (V, E, H)$ be the extended alignment graph for $k > 2$ sequences and D be a partition into blocks. Let $D_{i,j}$ be the set of blocks between sequences S_i and S_j, and let $P_{\mathcal{T}}(G_{ij})$ be the GMT polytope for the subgraph $G_{ij} = (V_{ij}, E_{ij}, H_{ij})$ induced by the edges in $\{\bigcup d \mid d \in D_{i,j}\}$. Every facet-defining inequality of $P_{\mathcal{T}}(G_{ij})$ is also facet-defining for $P_{\mathcal{T}}(G)$.*

Proof. Lemma 16 implies that a facet-defining inequality $c^T x \leq b$ is also facet-defining for $P_{\mathcal{T}}(G)$, because no edge in a block in $D \setminus D_{ij}$ can form a mixed cycle with an edge in the support of $c^T x \leq b$. $\square$

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 18 *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 $C_1 \cup \{e\}$ and $C_2 \cup \{e\}$ are critical mixed cycles where C_1 and C_2 are obtained by splitting C at v and w.*

For reasons of convenience we write $x(F) = \sum_{f \in F} x_f$.

Lemma 19 *Let $G = (V, E, H)$ be an extended alignment graph, D be a partition into blocks and C be a critical mixed cycle of size ℓ. Then the inequality*

$$x(v(C)) \leq \ell - 1$$

defines a facet of $P_{\mathcal{T}}(G)$ if and only if C has no chord.

Proof. Assume that C is a critical mixed cycle of size ℓ without a chord. Let $e_1, \ldots, e_\ell$ be ℓ edges on C. Note that by definition of D $v(e_1) \neq v(e_2) \neq \cdots \neq v(e_\ell)$. We obtain ℓ different feasible solutions by taking only the edges in $v(C)$

and removing the edges in $v(e_i)$, $1 \leq i \leq \ell$. The incidence vectors of these solutions are linearly independent and satisfy $x(v(C)) = \ell - 1$. Since C has no chord we can add any block from $D \setminus v(C)$ to one of the above solutions without inducing a mixed cycle on G. This yields another $|D| - \ell$ vectors that fulfill $x(v(C)) = \ell - 1$.

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

On the other hand, if C has a chord e then each incidence vector χ^M of a solution $M \subseteq D$ satisfying $x(v(C)) = \ell - 1$ has to satisfy $\chi^M_{v(e)} = 0$, so $dim\{x \in P_{\mathcal{T}}(G) | x(v(C)) = \ell - 1\} \leq |D| - 2$. Thus $x(v(C)) \leq \ell - 1$ is not a facet-defining inequality. $\square$

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

Lemma 20 *Let $G = (V, E, H)$ be an extended alignment graph consisting of a critical mixed cycle C of size ℓ with a chord e and D be a partition into blocks. Then the inequality*

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

defines a facet of $P_{\mathcal{T}}(G)$.

Proof. From Lemma 19 we know that $x(v(C)) \leq \ell - 1$ is a facet-defining inequality for $P_{\mathcal{T}}(G[E \setminus v(e)])$. Since $I_{\mathcal{T}}$ is an independence system we can use the lifting theorem to obtain the coefficient of the block $v(e)$ in the above facet-defining inequality. If we want to have a feasible solution containing the block $v(e)$, we have to remove another block from any feasible solution of $x(v(C)) \leq \ell - 1$, because e induces two critical mixed cycles together with the edges in C. Theorem 6 implies that the coefficient of $x_{v(e)}$ is 1 which proves the lemma. $\square$

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

3.3 The structure of the RSA polytope

In this section we investigate the structure of the RSA polytope. Recall that in the RSA secondary structure problem we are given an RSA graph

$G = (V, E, H, B)$ and want to compute a maximum weight RSA alignment. Unfortunately the pair $I_{\mathcal{R}}(G) = (E \cup B, \mathcal{R})$ does not form an independence system on $E \cup B$, because the base pair edges are dependent on the alignment edges. This deprives us of an elegant way of proving results about the RSA polytope. We now state some basic results about the RSA polytope and then define four non-trivial classes of valid inequalities and show in which case they are facet-defining.

Lemma 21 *Let $G = (V, E, H, B)$ be an RSA graph with n alignment and m base pair edges. Then $P_{\mathcal{R}}(G)$ is full-dimensional and the inequality $x_i \leq 1$ is facet-defining iff there is no $e_j \in E$ in conflict with e_i.*

Proof. The first part of the lemma is proven by exhibiting $n+m+1$ affinely independent incidence vectors of RSA alignments. We can easily do that by constructing n RSA alignments consisting of one alignment edge and m RSA alignments consisting of one base pair edge together with its generating alignment edges. Together with the zero vector this yields $n+m+1$ affinely independent incidence vectors.

To prove the second part we assume that there is no $e_j \in E$ which is in conflict with e_i. We define $n-1$ sets $\{e_j, e_i\}$, $\forall e_j \in E \setminus \{e_i\}$ and m sets $\{e_i, b, e_l, e_r\}$, $\forall b \in B$ where e_l, e_r are the generating alignment edges of b. Together with the set $\{e\}$ this yields $n+m$ RSA alignments A_k, $k = 1, \ldots, n+m$ whose incidence vectors χ^{A_k} are affinely independent and satisfy $\chi_i^{A_k} = 1$.

On the other hand, if there is a $e_j \in E$ which is in conflict with e_i, then for every incidence vector χ of an RSA alignment, $\chi_i = 1$ would imply $\chi_j = 0$. Therefore $dim\{x \in P_{\mathcal{R}}(G) \mid x_i = 1\} \leq n+m-1$ and hence $x_i \leq 1$ is not a facet-defining inequality. $\square$

Lemma 22 *Let $G = (V, E, H, B)$ be an RSA graph with n alignment and m base pair edges.*

1. *The inequality $x_i \geq 0$ is facet-defining iff e_i is not a generating alignment edge of a base pair edge.*
2. *For each base pair edge b_{ij} the inequality $x_{ij} \geq 0$ is facet-defining.*

Proof. (1) Let $e_i \in E$ be an alignment edge which is not a generating alignment edge of a base pair edge. Then the $n-1$ RSA alignments consisting of one alignment edge other than e_i and the m RSA alignments consisting of one base pair edge and its generating alignment edges form a collection of $n+m-1$ RSA alignments A_k, $k = 1, \ldots, n+m-1$ whose incidence vectors χ^{A_k} are affinely independent. Together with the zero vector this yields $n+m$ affinely independent incidence vectors with $\chi_i^{A_k} = 0$. Therefore $x_i \geq 0$ is a facet-defining inequality. On the other hand, if $e_i \in E$ is a generating alignment edge of a base pair edge b_{ij}, then for every incidence vector χ, $\chi_i = 0$ would imply $\chi_{ij} = 0$. Therefore $dim\{x \in P_{\mathcal{R}}(G) \mid x_i = 0\} \leq n + m - 1$ and hence $x_i \geq 0$ is not a facet-defining inequality.

(2) Let $b_{ij} \in B$ be a base pair edge. The n RSA alignments consisting of one alignment edge and the $m-1$ RSA alignments consisting of one base pair edge other than b_{ij} and its generating edges form $n + m - 1$ RSA alignments A_k, $k = 1, \ldots, n + m - 1$ whose incidence vectors χ^{A_k} are affinely independent. Together with the zero vector this yields $n+m$ affinely independent incidence vectors satisfying $\chi_{ij}^{A_k} = 0$. Therefore $x_{ij} \geq 0$ is a facet-defining inequality. □

We will now see that we can tighten both classes of inequalities that are in the ILP formulation. If one looks at an alignment edge e_i it might be that it is the generating alignment edge of a set $B_i \subseteq B$ of base pair edges. Since all pairs of base pair edges in B_i have a conflict – the generating alignment edges different from e_i start all in the same base of sequence S_1 – an RSA alignment can realize only one base pair edge in B_i. Therefore we can tighten these inequalities in the ILP formulation as follows:

$$\sum_{b_{ij} \in B_i} x_{ij} \leq x_i.$$

We call this class of valid inequalities *base pair inequalities* and show that they are facet-defining for the RSA polytope.

Theorem 23 *Let $G = (V, E, H, B)$ be an RSA graph. Let e_i be an alignment edge and let B_i be the set of base pair edges that have e_i as a generating alignment edge. Then the base pair inequality $\sum_{b_{ij} \in B_i} x_{ij} - x_i \leq 0$ is facet-defining for $P_{\mathcal{R}}(G)$.*

Proof. Denote the base pair inequality by $c^T x \leq c_0$. Obviously condition 3 b) of Theorem 7 holds because for the incidence vector χ_i of the set $\{e_i\}$ holds $c^T \chi_i < c_0$. Therefore it is sufficient to show that every valid inequality $a^T x \leq a_0$ with $\{x \mid c^T x = c_0\} \subseteq \{x \mid a^T x = a_0\}$ is – up to a multiplicative factor – equal to $c^T x \leq c_0$.

Assume that $\{x \mid c^T x = c_0\} \subseteq \{x \mid a^T x = a_0\}$. Since $c_0 = 0$ it follows that $a_0 = 0$. The incidence vectors $\chi^{\{e\}}$ of the $|E| - 1$ sets $\{e\}$, $\forall e \in E \setminus \{e_i\}$ fulfill $c^T \chi^{\{e\}} = a^T \chi^{\{e\}} = 0$. Hence $a_e = 0$, $\forall e \in E \setminus \{e_i\}$. Similarly the $|B| - |B_i|$ incidence vectors χ^{A_b} of the sets $A_b = \{b, e_l, e_r\}$, $\forall b \in B \setminus B_i$ fulfill $c^T \chi^{A_b} = a^T \chi^{A_b} = 0$. Hence $a_b = 0$, $\forall b \in B \setminus B_i$.

The sets $A_{b_{ij}} = \{e_i, b_{ij}, e_j\}$, $\forall b_{ij} \in B_i$ form RSA alignments whose incidence vectors $\chi^{A_{b_{ij}}}$ satisfy $c^T \chi^{A_{b_{ij}}} = 0$ and therefore $a^T \chi^{A_{b_{ij}}} = 0$. If one subtracts

$a^T\chi^{A_{b_{ij}}} = 0$ from $a^T\chi^{A_{b_{ij'}}} = 0$, $\forall b_{ij'} \in B_i \setminus \{b_{ij}\}$ this yields $a_{b_{ij}} = \cdots = a_{b_{ij'}}$, because we have just shown that all other coefficients except a_{e_i} are zero. Since $a_{b_{ij}} = -a_{e_i}$ we can choose $\lambda = \frac{1}{a_{b_{ij}}}$ which yields $\lambda \cdot a^T = c^T$. □

In the ILP formulation the inequalities $x_l + x_k \leq 1$, $\forall l, k$ with e_l in conflict with e_k ensure that only one of the conflicting edges can be realized. We can tighten these inequalities by augmenting them from pairs of edges to maximal sets of edges in which each pair of edges is in conflict.

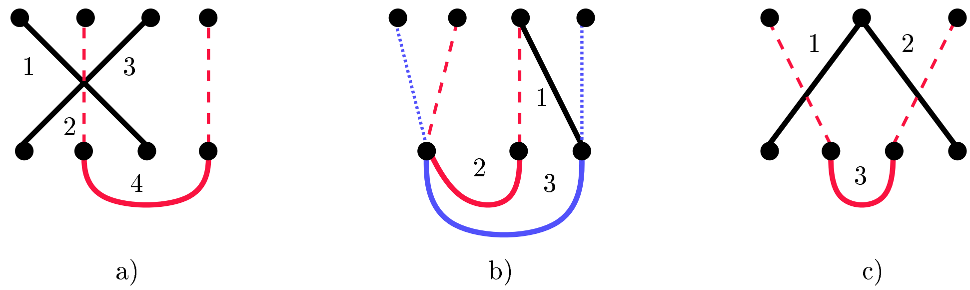


Fig. 11. One redundant (a) and two contributing (b,c) extended cliques (arcs in H not shown)

Definition 24 *Let $G = (V, E, H, B)$ be an RSA graph, $B' \subseteq B$ be a set of base pair edges and $E(B')$ be the set of generating alignment edges of B'. Let $E' \subseteq E \setminus E(B')$ be a set of alignment edges in G. If each pair of edges of the set $C = E' \cup B'$ is in conflict then C is called an extended clique.*

It is clear that only one edge from an extended clique can be realized by an alignment. Therefore the *extended clique* inequality $x(C) = \sum_{e_i \in E'} x_i + \sum_{b_{ij} \in B'} x_{ij} \leq 1$ is valid. We will prove that an extended clique inequality is facet-defining unless it is *redundant*, i.e. if one can replace a base pair edge by one of its generating edges such that the resulting set of edges is still an extended clique. If an extended clique is not redundant, we call it *contributing*. For example in Figure 11 (a) the set $\{1, 3, 4\}$ builds an extended clique. However, when replacing the base pair edge 4 by the base pair edge 2 (one of its generating alignment edges) this also yields an extended clique $\{1, 2, 3\}$. In Figure 11 (b) and (c) there is no way of replacing one of the base pair edges by a generating alignment edge such that the resulting set is still an extended clique. In both cases the set $\{1, 2, 3\}$ is a contributing extended clique.

Theorem 25 *Let $G = (V, E, H, B)$ be an RSA graph. Let $C = E' \cup B'$ be a maximal contributing extended clique in G. Then the inequality $x(C) \leq 1$ is facet-defining for $P_{\mathcal{R}}(G)$.*

Proof. Denote the extended clique inequality by $c^T x \leq c_0$. Condition 3 b) of Theorem 7 holds, because for the zero incidence vector $c^T\chi_i < c_0$. Therefore it is sufficient to show that every valid inequality $a^T x \leq a_0$ with $\{x \mid c^T x = c_0\} \subseteq \{x \mid a^T x = a_0\}$ is – up to a multiplicative factor – equal to $c^T x \leq c_0$.

Assume that $\{x \mid c^T x = c_0\} \subseteq \{x \mid a^T x = a_0\}$. All coefficients a_e of alignment edges $e \in E'$ are equal to a_0 because the set $\{e\}$ is an RSA alignment. Let e be any alignment edge not in C. Then there must be an edge in C such that this edge and e are not in conflict; otherwise C would not be maximal or it would be redundant. There are two cases:

(1) There is an alignment edge $e' \in E'$ such that e' and e are not in conflict. In that case the two sets $A = \{e'\}$ and $A' = \{e, e'\}$ build RSA alignments which satisfy $c^T \chi = c_0$ and therefore $a^T \chi = a_0$ for $\chi = \chi^A$ and $\chi = \chi^{A'}$. Subtracting $a_{e'} = a_0$ from $a_e + a_{e'} = a_0$ yields $a_e = 0$.
(2) There is no alignment edge e' with the above mentioned property. Consequently a base pair edge $b' \in B'$ must exist that is not in conflict with e and whose generating alignment edges e'_l and e'_r are different from e. Otherwise C would not be a maximal contributing extended clique. In this case the two sets $A = \{b', e'_l, e'_r\}$ and $A' = \{b', e'_l, e'_r, e\}$ build RSA alignments which satisfy $c^T \chi = c_0$ and therefore $a^T \chi = a_0$ for $\chi = \chi^A$ and $\chi = \chi^{A'}$. Subtracting $a_{b'} + a_{e'_l} + a_{e'_r} = a_0$ from $a_{b'} + a_{e'_l} + a_{e'_r} + a_e = a_0$ yields $a_e = 0$.

Since the coefficients of all alignment edges in $E \setminus E'$ are zero, the coefficients of base pair edges $b \in B'$ are equal to a_0. Let b be a base pair edge not in B'. The base pair edge b together with its generating alignment edges forms an RSA alignment. If one of its generating alignment edges is in C then the other generating alignment edge is in $E \setminus E'$, because generating edges cannot be in conflict. Since one of the coefficients of the generating edges is a_0 and the other is 0 the coefficient a_b has to be 0. If both generating edges of b are in $E \setminus E'$ then their coefficients are 0 and there are again two cases:

(1) There is an alignment edge $e' \in E'$ such that e' and b are not in conflict. In that case the two sets $A = \{e', e_l, e_r\}$ and $A' = \{b, e_l, e_r, e'\}$ build RSA alignments which satisfy $c^T \chi = c_0$ and therefore $a^T \chi = a_0$ for $\chi = \chi^A$ and $\chi = \chi^{A'}$. Subtracting $a_{e'} + a_{e_l} + a_{e_r} = a_0$ from $a_{e'} + a_b + a_{e_l} + a_{e_r} = a_0$ yields $a_b = 0$
(2) There is no alignment edge e' with the above mentioned property. Consequently a base pair edge $b' \in B'$ must exist that is not in conflict with b and whose generating alignment edges are different from those of b. In this case the two sets $A = \{b', e'_l, e'_r, e_l, e_r\}$ and $A' = \{b', e'_l, e'_r, b, e_l, e_r\}$ build RSA alignments which satisfy $c^T \chi = c_0$ and therefore $a^T \chi = a_0$ for $\chi = \chi^A$ and $\chi = \chi^{A'}$. Subtracting $a_{b'} + a_{e'_l} + a_{e'_r} + a_{e_l} + a_{e_r} = a_0$ from $a_{b'} + a_{e'_l} + a_{e'_r} + a_{e_l} + a_{e_r} + a_b = a_0$ yields $a_b = 0$.

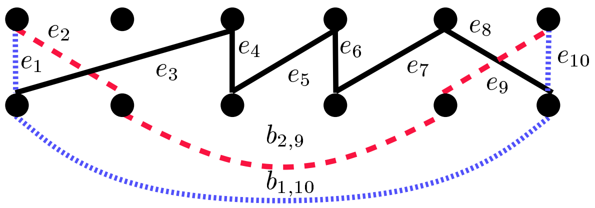
This completes the proof that the coefficients of edges not in C are 0. The rest of the coefficients is equal to a_0. Choosing $\lambda = \frac{a_0}{c_0}$ we have $a_0 = \lambda \cdot c_0$ and $a^T = \lambda \cdot c^T$. Therefore $c^T x \leq c_0$ is a facet-defining inequality for $P_{\mathcal{R}}(G)$. $\square$

The above mentioned classes of facet-defining inequalities do not form a complete description of the RSA polytope for two sequences. It is indeed an open question to find such a complete description. We were able to identify another class of inequalities that is not always facet-defining, the *odd cycle inequalities*. In the following we characterize this class and prove that it is facet-defining for certain RSA graphs. All indices are to be read modulo $2i + 2$.

Definition 26 *Let $G = (V, E, H, B)$ be an RSA graph containing $2i+1$ contributing extended cliques $C_1, \ldots, C_{2i+1}$. The set $C := C_1 \cup C_2 \cup \cdots \cup C_{2i+1}$ is called odd cycle of length i if for all $f \in C_j$, $1 \leq j \leq 2i+1$ holds: f is in conflict with each $g \in C_{j-1} \cup C_j \setminus \{f\} \cup C_{j+1}$ and not in conflict with some $g \in C_k$, $k \neq j-1, j, j+1$.*

Given an RSA graph consisting of an odd cycle C of length i together with the generating edges of the base pair edges contained in C, only i edges in C can be realized simultaneously, namely one out of every other extended clique. Therefore for any odd cycle C the *odd cycle inequality* $x(C) = \sum_{b_{ij} \in C \cap B} x_{ij} + \sum_{e_j \in C \cap E} x_j \leq i$ is valid. Note that an odd cycle must contain at least one base pair edge.

In Figure 12 an odd cycle of length 3 is shown. More specifically $C_1 = \{b_{1,10}, b_{2,9}\}$, $C_2 = \{e_3\}$, $C_3 = \{e_4\}$, $\ldots$, $C_6 = \{e_7\}$ and $C_7 = \{e_8\}$. The



$$x_{1,10} + x_{2,9} + x_3 + x_4 + x_5 + x_6 + x_7 + x_8 \leq 3$$

Fig. 12. An RSA graph forming an odd cycle of length 3 (arcs in H not shown)

odd cycle inequality is indeed facet-defining, if the RSA graph G is an odd cycle together with its generating edges. We prove this in the following theorem using the notations from Definition 26.

Theorem 27 *Let $G = (V, E, H, B)$ be an RSA graph consisting of an odd cycle C and its generating alignment edges. Then the odd cycle inequality $x(C) = \sum_{b_{ij} \in C \cap B} x_{ij} + \sum_{e_j \in C \cap E} x_j \leq i$ is facet-defining for $P_{\mathcal{R}}(G)$.*

Proof. Denote the odd cycle inequality by $c^T x \leq c_0$. Clearly condition 3 b) of Theorem 7 holds. If we realize any edge contained in one of the extended cliques of an odd cycle, we have a valid RSA alignment, the incidence vector of which fulfills $c^T \chi_i < c_0$. Hence it is sufficient to show that every valid inequality $a^T x \leq a_0$ with $\{x \mid c^T x = c_0\} \subseteq \{x \mid a^T x = a_0\}$ is – up to a multiplicative factor – equal to $c^T x \leq c_0$.

Assume that $\{x \mid c^T x = c_0\} \subseteq \{x \mid a^T x = a_0\}$. Throughout the proof d_j denotes a set containing either an alignment edge in C_j or a base pair edge together with its generating alignment edges.

First we show that the coefficients of all generating alignment edges in G are zero. Let b_j be a base pair edge contained in some contributing extended clique C_j and e_l and e_r be its left respectively right generating edge. Since b_j is in conflict with all edges in C_{j-1} and with all edges in C_{j+1} it follows from the definition of conflict that e_l is in conflict with all edges in C_{j-1} and e_r is in conflict with all edges in C_{j+1}. Additionally there must exist an edge in C_{j+1} that is not in conflict with e_l and an edge in C_{j-1} that is not in conflict with e_r. If this was not true, C_j would be a redundant extended clique in the odd cycle, because one could replace b_j by its left or right generating edge.

Therefore there exist sets $L_j := d_{j-2} \cup d_{j+1} \cup d_{j+3} \cup \cdots \cup d_{j-4}$ and $R_j := d_{j-1} \cup d_{j+2} \cup \cdots \cup d_{j-3}$ that form RSA alignments satisfying $c^T \chi = c_0$ and hence $a^T \chi = a_0$. Also the sets $L_j \cup \{e_l\}$ and $R_j \cup \{e_r\}$ form RSA alignments satisfying $c^T \chi = c_0$ and hence $a^T \chi = a_0$. The subtraction of the two equalities yields $a_{e_l} = a_{e_r} = 0$. Since the above argument holds for any base pair edge in C it follows that the coefficients of all generating alignment edges in G are zero.

Next we show that the coefficients of all edge variables in an odd cycle inequality are equal. Define for $k = 1, \ldots, i+1$ the sets

$$M_k := d_2 \cup d_4 \cup \cdots \cup d_{2k-2} \cup d_{2k+1} \cup d_{2k+3} \cup \cdots \cup d_{2i+1}$$

that means M_k contains one edge (possibly together with its generating edges) from each extended clique with even indices from 2 to $2k-2$ and one edge from each extended clique with odd indices from $2k+1$ to $2i+1$. Every M_k forms an RSA alignment and its incidence vector χ_k satisfies $c^T \chi_k = c_0$ and hence $a^T \chi_k = a_0$. Subtracting $a^T \chi_{k+1} = a_0$ from $a^T \chi_k = a_0$ yields $a_{d_{2k+1}} = a_{d_{2k}}$ for $k = 1, \ldots, i$.

Next we define for $k = 0, \ldots, i$ the sets

$$N_k := d_1 \cup d_3 \cup \cdots \cup d_{2k-1} \cup d_{2k+2} \cup d_{2k+2} \cup \cdots \cup d_{2i}$$

that means N_k contains one edge from each extended clique with odd indices from 1 to $2k-1$ and one edge from each extended clique with even indices from $2k+2$ to $2i$. Every N_k forms an RSA alignment and its incidence vector χ_k satisfies $c^T\chi_k = c_0$ and hence $a^T\chi_k = a_0$. Subtracting $a^T\chi_{k+1} = a_0$ from $a^T\chi_k = a_0$ yields $a_{d_{2k+2}} = a_{d_{2k+1}}$ for $k = 0, \dots, i-1$.

Putting the above arguments together we have $a_{d_1} = a_{d_2} = \cdots = a_{d_{2i+1}}$. Since the argument holds for any alignment edge $e \in C_i$ and any base pair edge $b \in C_i$ the coefficients in the odd cycle C are equal. Choosing $\lambda = \frac{a_0}{c_0}$ we have $a_0 = \lambda \cdot c_0$ and $a^T = \lambda \cdot c^T$. Therefore $c^T x \leq c_0$ is a facet-defining inequality for $P_{\mathcal{R}}(G)$. $\square$

In the multiple sequence case we show that the *mixed cycle* inequalities are facet-defining for $P_{\mathcal{R}}(G)$ if and only if they contain no chord.

Lemma 28 *Let $G = (V, E, H, B)$ be an RSA graph and C be a critical mixed cycle with $|C \cap E| = \ell$. Then the inequality*

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

defines a facet of $P_{\mathcal{R}}(G)$ if and only if C has no chord.

Proof. Assume that C is a critical mixed cycle with $|C \cap E| = \ell$ and without chord. Let $e_1, \dots, e_\ell$ be ℓ edges on C. We obtain ℓ different feasible solutions by removing the edge e_i, $1 \leq i \leq \ell$ from C. The incidence vectors of these solutions are linearly independent and satisfy $x(C \cap E) = \ell - 1$. Since C has no chord and is a critical mixed cycle we can add either a base pair edge $b \in B$ together with its generating alignment edges e_l and e_r or an alignment edge $e \in E \setminus C$ to one of the above solutions without introducing a mixed cycle in G. This yields another $m+n-\ell$ vectors that fulfill $x(C \cap E) = \ell - 1$. Moreover, the incidence vectors of all 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^A of a solution $A \subseteq E \cup B$ satisfying $x(C \cap E) = \ell - 1$ has to satisfy $\chi^A_e = 0$, so $dim\{x \in P_{\mathcal{R}}(G) | x(C \cap E) = \ell - 1\} \leq |E \cup B| - 2$. Thus $x(C \cap E) \leq \ell - 1$ is not a facet-defining inequality. $\square$

4 The branch-and-cut algorithms

Branch-and-cut algorithms have been first applied successfully to the linear-ordering problem [9], and then for the traveling-salesman problem [32]. In the meantime they are applied in many fields of Operations Research and the

Natural Sciences. This is the first time that branch-and-cut algorithms are used in the field of Computational Molecular Biology.

In order to apply branch-and-cut algorithms successfully for combinatorial optimization problems, the following tasks need to be solved:

(1) Definition of a polytope P (or polyhedron) associated to the problem.
(2) Investigation of the structure of P in form of facet-defining inequalities; this gives a partial description defining a polytope $R \supseteq P$.
(3) Solving the separation problems over the polytope R. (the separation problem for a class of inequalities takes a point in ^{n} and returns a violated inequality from the class if there is one).

We have solved the tasks (1) and (2) in Section 2 and in Section 3, respectively. In this section we will show how to solve the separation problems for some of the classes of facet-defining inequalities given in Section 3.

Although the number of inequalities in the complete description of the MT polytope is exponential, we will show how to solve the separation problem for it in polynomial time. According to Grötschel, Lovász and Schrijver [10] this implies that the associated relaxed optimization problem can be solved in polynomial time. This equivalence of *optimization and separation* leads to a polynomial time algorithm for the MT problem for two sequences.

In the following we will describe how branch-and-cut algorithms work. First we relax the given integer linear program by dropping the integer condition and solve the resulting linear program. If the solution $\bar{x}$ of the linear program is integral we have the optimal solution. Otherwise we search for a valid inequality $fx \leq f_0$ that "cuts off" the solution $\bar{x}$, i.e, $fy \leq f_0$ for all $y \in P$ (P is the convex hull of all feasible solutions) and $f\bar{x} > f_0$; the set $\{x \mid fx = f_0\}$ is called a *cutting plane*. The search for a cutting is done by solving the separation problem for all known classes of (facet-defining) inequalities. 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: We generate two subproblems by setting one fractional variable to 0 in the first subproblem and to 1 in the second subproblem and solve these subproblems recursively. This gives rise to an enumeration tree of subproblems.

4.1 A branch-and-cut algorithm for the GMT problem

In order to specialize the generic branch-and-cut algorithm we need to describe separation algorithms for our various classes of inequalities.

First we describe how to solve the separation problem for the mixed cycle inequalities. Assume the solution $\bar{x}$ of the linear program is fractional. Our problem is to find a critical mixed cycle C in the extended alignment graph $G = (V, E, H)$ which violates the mixed-cycle inequality $\sum_{d \in v(C)} x_d \le |v(C)| - 1$. First assign for each block d the cost $1 - \bar{x}_d$ to each edge $e \in d$ and 0 to all $a \in H$. Then compute for each arc $a = (u, v) = (s_{ij}, s_{ij+1})$, $1 \le i \le k$, $1 \le j < n_i$ the shortest path from v to u. Together with the arc a this path forms a mixed cycle. During this computation we have to take care that we compute the shortest path with the fewest edges. This can be done by ordering paths lexicographically according to their costs and then according to the number of edges. Then the lexicographically shortest mixed cycle is also critical.

If a shortest path P from v to u is found it must contain $l \ge 2$ edges $e_1, \dots, e_l$ from different blocks. If the cost of P is less than 1, i.e. $\sum_{d \in v(P)} (1 - \bar{x}_d) < 1$, a violated inequality is found, namely $\sum_{d \in v(P)} \bar{x}_d > |v(P)| - 1$.

Theorem 29 *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.*

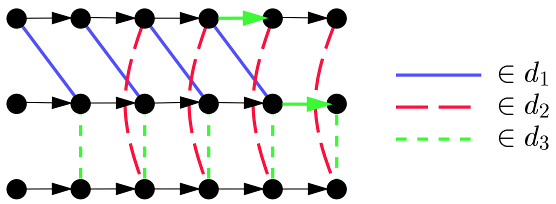


Fig. 13. Alignment graph with 13 edges partitioned into 3 blocks d_1, d_2, d_3. Only the dotted purple arcs need to be checked.

Unfortunately this might still result in a big number of shortest path computations. This is particularly annoying for partitions with large blocks, because there is a lot of different paths resulting in the same inequality. For example Figure 13 shows an EAG with a partition into three blocks. The only mixed cycle inequality that can be found is $x_{d_1} + x_{d_2} + x_{d_3} \le 2$. With the naive approach we would have to make 15 shortest path computations. We will show that in this example it is safe to make only two such computations (the dotted purple arcs).

We call two paths P and P' *equivalent* if and only if $v(P) = v(P')$. The set of all paths forms equivalence classes under the above relation. We will now show how to pick a subset $A \subseteq H$ of arcs such that we only have to compute a shortest path from v to u for each $a = (u, v) \in A$. We do that by excluding certain arcs from consideration.

In a block $d \in D$ we say that an alignment edge e is *right of* an alignment edge f if both edges run between nodes of the same sequence and *start*$(e) >$ *start*(f) and *end*$(e) >$ *end*(f). Let $D(u, i)$ be the set of all blocks that have an edge incident to u and to a node in sequence i, i.e., $D(u,i) := \{d \in D \mid \exists e = \{u, s_{ix}\} \in d$ for some $1 \leq x \leq n_i\}$. Then the following lemma holds:

Lemma 30 *Let $a = (u,v)$ be an arc in sequence S_i with $D(u,j) \subseteq D(v,j)$ for some $1 \leq j \neq i \leq k$. Then for any critical mixed cycle C that enters S_i from S_j through an edge e incident to u and that contains a, there is an equivalent critical mixed cycle C' using an edge $f \in v(e)$ which is right of e.*

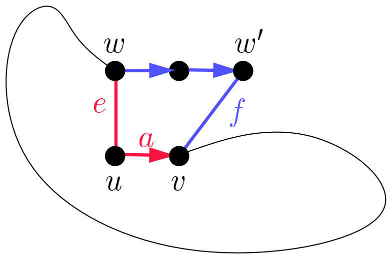


Fig. 14. Two equivalent critical mixed cycles

Proof. Since $a = (u,v)$ is an arc and $D(u,j) \subseteq D(v,j)$ both nodes u and v must be incident to edges in $v(e)$, namely to e and f. Since the block $v(e)$ is a trace f must be right of e. Let w be the node in S_j that is incident to e and w' be the node in S_j incident to f. We can construct C' from C by deleting e and a from C and replacing it by the path consisting of the arcs running from w to w' followed by f (see Figure 14). $\square$

We can therefore discard an arc $a = (u,v)$ in sequence S_i if for all $j = 1, \ldots, k$, $j \neq i$ holds that $D(u,j) \subseteq D(v,j)$, because for any critical mixed cycle C that enters at node u there is an equivalent critical mixed cycle C' and any critical mixed cycle that contains a and enters S_i before u must contain an additional arc in S_i.

Theorem 31 *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 $|A|$ shortest paths in G, where $A \subseteq H$ is defined as $\{(u,v) \in H \mid \exists i \in \{1, \ldots, k\}$ such that $D(u,i) \subsetneq D(v,i)\}$.*

In the separation algorithm for the class of clique inequalities we make use of the *pairgraph* $P_{\mathcal{T}}(G_{ij})$ between sequence S_i and S_j for $1 \leq i < j \leq k$. 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 $\sum_{d \in v(C)} \bar{x}_d \leq 1$. For each edge $e \in d$ assign the cost $\bar{x}_d$ to the node v_e in $P_{\mathcal{T}}(G_{ij})$. Recall that no two edges in the same block can lie on a source-to-sink path and that all maximal cliques in the independence system are represented by some source-to-sink path.

We compute the longest source-to-sink path C in $P_{\mathcal{T}}(G_{ij})$. If the cost of C is greater than 1, i.e. $\sum_{d \in v(C)} \bar{x}_d > 1$ we have found a violated clique inequality. Since $P_{\mathcal{T}}(G_{ij})$ is acyclic, such a path can be found in time polynomial in the size of the EAG.

Theorem 32 *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 $P_{\mathcal{T}}(G_{ij})$ for $1 \leq i < j \leq k$, where k is the number of sequences.*

In the branch-and-cut algorithm we first separate the clique inequalities as described above. If we cannot find a violated clique inequality we check whether the EAG contains a mixed cycle by computing a shortest path from v to u for each arc $a = (u, v)$ in a set A which is defined in Theorem 31. If we find one or more such paths we add the corresponding mixed cycle inequalities to the LP and resolve it. Finally, if we do not find any violated inequalities or if the solution value of the LP does not improve significantly over a number of iterations, we branch.

In the branching phase we choose the fractional base pair variable which is closest to 0.5 and has the highest objective function coefficient. After the branching we iterate the process on the two subproblems.

4.2 A branch-and-cut algorithm for the RSA problem

In order to specialize the generic branch-and-cut algorithm for the RSA problem we need to describe separation algorithms for our classes of inequalities. Since each base pair inequality is only a stronger version of exactly one inequality in the ILP formulation, we replace each such inequality in the ILP formulation by the corresponding base pair inequality.

All maximal extended cliques C that do not contain a base pair edge are contributing and therefore the extended clique inequality of C is facet-defining for the RSA polytope. Above we showed that those inequalities can be separated in polynomial time by computing a longest path in a pairgraph. The other maximal extended cliques C contain at least one base pair edge and are by Theorem 25 facet-defining if and only if they are contributing. Unfortunately we have not found an efficient way of separating this class of inequalities. By checking the ILP inequality $x_i + x_j \leq 1, \forall i, j \in E$ with i is in conflict with j, we can determine the feasibility of an (integer) solution of the LP. If we find two conflicting edges $i, j \in E$ with $x_i + x_j > 1$ we chose two ways of handling this situation:

(1) If the RSA graph is not too dense, we enumerate all maximal extended

cliques containing i (or j). With the use of bit vectors and an adaption of an algorithm by Tsukiyama et al. [38] this can be done in reasonable time for up to 20000-30000 cliques. This cuts off the current infeasible solution and considerably shrinks the enumeration tree in the branching phase.

(2) If the RSA graph is too dense and therefore the number of cliques in the above mentioned enumeration explodes, we refrain from enumerating the extended cliques (which we do in most cases).

If we cannot find a violated extended clique inequality and our solution is still fractional we apply a heuristic for finding violated odd cycle inequalities. The heuristic chooses a fractional base pair variable x_{ij} and tries to find an even number of "connecting" alignment edges between the left and right generating edge of maximal value. If it finds such a set C of $2i$ alignment edges with the property $x_{ij} + \sum_{e_j \in C} x_j > i$ it adds the odd cycle inequality $x_{ij} + \sum_{e_j \in C} x_j \leq i$ to the LP. If this also fails we branch.

In the branching phase we choose the fractional base pair variable which is closest to 0.5 and has the highest objective function coefficient. After the branching we iterate the process on the two subproblems.

5 Computational results

In this section we report on the results generated by our program. The implementation is coded in C++ using the library of efficient data types and algorithms LEDA [25] and the branch-and-cut framework ABACUS [17].

5.1 The GMT problem

We tested three different ways to generate the extended alignment graph.

- As an example of a scoring scheme based on the comparison of two residues (MT) we adapted the PRIMAL [20] package by John Kececioglu. The value of the approximate solution of this program is used as a lower bound in our branch-and-cut algorithm.
- As an example for a scoring scheme based on the comparison of segment pairs we adopted two ways to generate the input for the branch-and-cut algorithm. The first takes the set of blocks that are computed by Burkhard Morgenstern's DIALIGN package [27]. The weight of DIALIGN's greedy heuristic is used as a lower bound for the branch-and-cut algorithm.
- In the second approach we compute (sub)optimal local alignments between

two sequences that do not share (mis)matches. We call the procedure that produces the blocks LOCAL. Here we employed a simple greedy strategy to compute lower bounds for the branch-and-cut algorithm.

```
  0 M---------LVLFVATWSDLGLCKKRPKP-GGWNTGGSRYPGQGSPGGNRYPPQGGGGWGQPHGGGWGQPHGGGWGQPHGGGWGQPHG
  0 M--ANLGCWMLVLFVATWSDLGLCKKRPKP-GGWNTGGSRYPGQGSPGGNRYPPQGGGGWGQPHGGGWGPHGGGWGQPHGGGWGQPHGG
  0 M--ANLGCWMLVLFVATWSDLGLCKKRPKP-GGWNTGGSRYPGQGSPGGNRYPPQGGGGWGQPHGGGWGQPHGGGWGQPHGGGWGQPHG
  0 M--ANLGCWMLVVFVATWSDLGLCKKRPKP-GGWNTGGSRYPGQGSPGGNRYPPQGGGGWGQPHGGGWGQPHGGGWGQPHGGGWGQ---
  0 M--ANLGCWMLVLFVATWSDLGLCKKRPKP-GGWNTGGSRYPGQGSPGGNLYPPQGGG-WGQPHGGGWGQPHGGGWGQPHGGSWGQPHG
  0 M--ANLGCWMLVLFVATWSDLGLCKKRPKP-GGWNTGGSRYPGQGSPGGNRYPPQGGGGWGQPHGGGWGQPHGGGWGQPHGGGWGQPHG
  0 M--ANLGCWMLVLFVATWSNLGLCKKRPKP-GGWNTGGSRYPGQGSPGGNRYPPQGGGGWGQPHGGGWGQPHGGGWGQPHGGGWGQPHG
  0 M--ANLGCWMLVLFVATWSDLGLCKKRPKP-GGWNTGGSRYPGQGSPGGNRYPPQGGGGWGQPHGGGWGQPHGGGWGQPHGGGWGQPHG
  0 M--ANLGCWMLVLFVATWSDLGLCKKRPKP-GGWNTGGSRYPGQGSPGGNRYPPQGGGGWGQPHGGGWGQPHGGGWGQPHGGGWGQPHG
  0 MVKSHIGSWILVLFVAMWSDVGLCKKRPKPGGGWNTGGSRYPGQGSPGGNRYPPQGGGGWGQPHGGGWGQPHGGGWGQPHGGGWGQPHG
  0 MVKSHIGSWILVLFVAMWSDVGLCKKRPKPGGGWNTGGSRYPGQGSPGGNRYPPQGGGGWGQPHGGGWGQPHGGGWGQPHGGGWGQPHG
  0 MVKSHIGSWILVLFVAMWSDVGLCKKRPKPGGGWNTGGSRYPGQGSPGGNRYPPQGGGGWGQPHGGGWGQPHGGGWGQPHGGGWGQPHG
  0 -------------------------------GGWNTGGSRYPGQGSPGGNRYPPQSGGTWGQPHGGGWGQPHGGGWGQPHGGGWGQPHG
  0 M--ANLSYWLLALFVAMWTDVGLCKKRPKP-GGWNTGGSRYPGQGSPGGNRYPPQGGGTWGQPHGGGWGQPHGGGWGQPHGGGWGQPHG
  0 M--ANLGYWLLALFVTMWTDVGLCKKRPKP-GGWNTGGSRYPGQGSPGGNRYPPQGG-TWGQPHGGGWGQPHGGSWGQPHGGSWGQPHG

 90 GGWGQ--------GGGTHNQWHKPNKPKTSMKHMAGAAAAGAVVGGLGGYMLGSAMSRPLIHFGNDYEDRYYRENMYRYPNQVYYRPVD
 90 GWGQ---------GGGTHSQWNKPSKPKSNMKHMAGAAAAGAVVGGLGGYMLGSAMSRPLIHFGNDYEDRYYRENMYRYPNQVYYRPVD
 90 GGWGQ--------GGGTHNQWHKPSKPKTSMKHMAGAAAAGAVVGGLGGYMLGSAMSRPLIHFGNDYEDRYYRENMYRYPNQVYYRPVD
 90 -------------GGGTHNQWHKPSKPKTSMKHMAGAAAAGAVVGGLGGYMLGSAMSRPLIHFGNDYEDRYYRENMYRYPNQVYYRPVD
 90 GGWGQ--------GGGTHNQWNKPSKPKTSMKHVAGAAAAGAVVGGLGGYMLGSAMSRPLIHFGNDYEDRYYRENMYRYPNQVYYRPVD
 90 GGWGQ--------GGGTHSQWNKPSKPKTNMKHMAGAAAAGAVVGGLGGYMLGSAMSRPIIHFGSDYEDRYYRENMHRYPNQVYYRPMD
 90 GGWGQ--------GGGTHSQWNKPSKPKTNMKHMAGAAAAGAVVGGLGGYMLGSAMSRPIIHFGNDYEDRYYRENMYRYPNQVYYRPVD
 90 GGWGQ--------GGGTHSQWNKPSKPKTNMKHMAGAAAAGAVVGGLGGYMLGSAMSRPIIHFGSDYEDRYYRENMHRYPNQVYYRPMD
 90 GGWGQ--------GGGTHSQWNKPSKPKTNMKHMAGAAAAGAVVGGLGGYMLGSAMSRPIIHFGSDYEDRYYRENMHRYPNQVYYRPMD
 90 GGWGQPHGGGGWGQGGTHGQWNKPSKPKTNMKHVAGAAAAGAVVGGLGGYMLGSAMSRPLIHFGSDYEDRYYRENMHRYPNQVYYRPVD
 90 GGGWG--------QGGSHSQWNKPSKPKTNMKHVAGAAAAGAVVGGLGGYMLGSAMSRPLIHFGNDYEDRYYRENMYRYPNQVYYRPVD
 90 GGGWG--------QGGTHSQWNKPSKPKTNMKHVAGAAAAGAVVGGLGGYMLGSAMNRPLIHFGNDYEDRYYRENMYRYPNQVYYRPVD
 90 GGWSQ--------GGGTHNQWNKPSKPKTNLKHVAGAAAAGAVVGGLGGYMLGSAMSRPMLHFGNDWEDRYYRENMYRYPNQVYYRPVD
 90 GGWGQ--------GGGTHNQWNKPSKPKTNMKHMAGAAAAGAVVGGLGGYMLGSAMSRPMMHFGNDWEDRYYRENMNRYPNQVYYRPVD
 90 GGWGQ--------GGGTHNQWNKPSKPKTNLKHVAGAAAAGAVVGGLGGYMLGSAMSRPMIHFGNDWEDRYYRENMYRYPNQVYYRPVD

180 QYSNQNNFVHDCVNITIKQHTVTTTTKGENFTETDVKMMERVVEQMCITQYEKESQAYYQ--RGSSMVLFSSPPVILLISFLI-----
180 QYSNQNNFVHDCVNITIKQHTVTTTTKGENFTETDVKMMERVVEQMCITQYEKESQAYYQ--RGSSMVFFSSPPVILLISFLIFLIVG
180 QYSNQNNFVHDCVNITIKQHTVTTTTKGENFTETDVKMMERVVEQMCITQYEKESQAYYQ--RGSSMVLFSSPPVILLISFLIFLIVG
180 QYSNQNNFVHDCVNITIKQHTVTTTTKGENFTETDVKMMERVVEQMCITQYEKESQAYYQ--RGSSMVLFSSPPVILLISFLIFLIVG
180 QYSNQNNFVHDCVNITIKQHTVTTTTKGENFTETDVKMMERVVEQMCITQYERESQAYYQ--RGSSMVLFSSPPVILLISFLIFLIVG
180 QYSSQNNFVHDCVNITIKQHTVTTTTKGENFTETDVKMMERVVEQMCITQYERESQAYYQ--RGSSMVLFSSPPVILLISFLIFLIVG
180 QYSNQNNFVHDCVNITIKQHTVTTTTKGENFTETDVKMMERVVEQMCITQYERESQAYYQ--RGSSMVLFSSPPVILLISFLIFLIVG
180 QYSNQNNFVHDCVNITIKQHTVTTTTKGENFTETDVKMMERVVEQMCITQYERESQAYYQ--RGSSMVLFSSPPVILLISFLIFLIVG
180 EYSNQNNFVHDCVNITIKQHTVTTTTKGENFTETDVKMMERVVEQMCITQYERESQAYYQ--RGSSMVLFSSPPVILLISFLIFLIVG
180 QYSNQNNFVHDCVNITVKEHTVTTTTKGENFTETDIKMMERVVEQMCITQYQRESQAYYQ--RGASVILFSSPPVILLISFLIFLIVG
180 RYSNQNNFVHDCVNITVKQHTVTTTTKGENFTETDIKIMERVVEQMCITQYQRESQAYYQ--RGASVILFSSPPVILLISFLIFLIVG
180 QYNNQNTFVHDCVNITVKQHTVTTTTKGENFTETDIKMMERVVEQMCITQYQRESQAYYQ--RGASVILFSSPPVILLISFLIFLIVG
180 QYSNQNNFVHDCVNITIKQHTVTTTTKGENFTETDVKMMERVVEQMCVTQYQKESQAYYDG-RRSSAVLFSSPPVILLISFLIFLIVG
180 QYNNQNNFVHDCVNITIKQHTVTTTTKGENFTETDIKIMERVVEQMCTTQYQKESQAYYDG-RRSSAVLFSSPPVILLISFLIFLMVG
180 QYSNQNNFVHDCVNITIKQHTVTTTTKGENFTETDVKMMERVVEQMCVTQYQKESQAYYDGRRSSSTVLFSSPPVILLISFLIFLIVG
```

Fig. 15. Optimal alignment of 15 prion protein sequences. Input was generated using PRIMAL

In the following we describe the three approaches in more detail.

Blocks computed by PRIMAL:
To generate an extended 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

```
Exact       value: 2067
Approximate value: 1278

HUMA : V-LSPADKTNVKAAWGKVGAHAGEYGAEALERMFLSFPTTKTYFPHF-
HAOR : M-LTDAEKKEVTALWGKAAGHGEEYGAEALERLFQAFPTTKTYFSHF-
HADK : V-LSAADKTNVKGVFSKIGGHAEEYGAETLERMFIAYPQTKTYFPHF-
HBHU : VHLTPEEKSAVTALWGKV--NVDEVGGEALGRLLVVYPWTQRFFESFG
HBOR : VHLSGGEKSAVTNLWGKV--NINELGGEALGRLLVVYPWTQRFFEAFG
HBDK : VHWTAEEKQLITGLWGKV--NVADCGAEALARLLIVYPWTQRFFASFG

HUMA : DLS-----HGSAQVKGHGKKVADALTNAVAHVDDMPNALSALSDLHAH
HAOR : DLS-----HGSAQIKAHGKKVADALSTAAGHFDDMDSALSALSDLHAH
HADK : DLS-----HGSAQIKAHGKKVAAALVEAVNHVDDIAGALSKLSDLHAQ
HBHU : DLSTPDAVMGNPKVKAHGKKVLGAFSDGLAHLDNLKGTFATLSELHCD
HBOR : DLSSAGAVMGNPKVKAHGAKVLTSFGDALKNLDDLKGTFAKLSELHCD
HBDK : NLSSPTAILGNPMVRAHGKKVLTSFGDAVKNLDNIKNTFAQLSELHCD

HUMA : KLRVDPVNFKLLSHCLLVTLAAHLPAEFTPAVHASLDKFLASVSTVLT
HAOR : KLRVDPVNFKLLAHCILVVLARHCPGEFTPSAHAAMDKFLSKVATVLT
HADK : KLRVDPVNFKFLGHCFLVVVAIHHPAALTPEVHASLDKFMCAVGAVLT
HBHU : KLHVDPENFRLLGNVLVCVLAHHFGKEFTPPVQAAYQKVVAGVANALA
HBOR : KLHVDPENFNRLGNVLIVVLARHFSKDFSPEVQAAWQKLVSGVAHALG
HBDK : KLHVDPENFRLLGDILIIVLAAHFTKDFTPECQAAWQKLVRVVAHALA

HUMA : SKYR
HAOR : SKYR
HADK : AKYR
HBHU : HKYH
HBOR : HKYH
HBDK : ARYH
```

Fig. 16. Optimal alignment of 6 globin sequences. Input was generated using LOCAL.

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.

Blocks computed by DIALIGN:
In the DIALIGN program the blocks are called *diagonals* because a block represents a gapless alignment which is a diagonal run in the corresponding dynamic programming matrix. The algorithm greedily picks the best diagonal from *all* possible diagonals which is consistent with previously chosen diagonals. Although this input could be modeled in the GMT formulation it is far too big. Therefore we input solely diagonals stemming from optimal pairwise alignments, that are not in conflict.

```
Exact       value: 59862.5
Approximate value: 53475.1

----------MLVLFVATWSDLGLCKKRPKP-GGWNTGGSRYPGQSSPGGNRYPPQs--GGWGQPHGGGWGQPHGGGWGQPHGGGWGQPHG
MVK-SHIGSWLLVLFVATWSDIGFCKKRPKPGGGWNTGGSRYPGQGSPGGNRYPPQ-GGGGWGQPHGGGWGQPHGGGWGQPHGGGWGQPHG
----------MLVLFVATWSDLGLCKKRPKP-GGWNTGGSRYPGQGSPGGNRYP----------PQGGGWGQPHGGGWGQPHGGGWGQPHG
----------MLVLFVATWSDLGLCKKRPKP-GGWNTGGSRYPGQGSPGGNRYPPQ-GGGGWGQPHGGGWGQPHGGGWGQPHGGGWGQPHG
---MANLGCWMLVLFVATWSDLGLCKKRPKP-GGWNTGGSRYPGQGSPGGNRYPPQ-GGGGWGQPHGGGWGQPHGGGWGQPHGGGWGQPHG
---MANLGCWMLVLFVATWSDLGLCKKRPKP-GGWNTGGSRYPGQGSPGGNRYPPQ-GGGGWGQPHGGGWGQPHGGGWGQPHGGGWGQPHG
---MANLGCWMLVVFVATWSDLGLCKKRPKP-GGWNTGGSRYPGQGSPGGNRYPPQ-GGGGWGQPHGGGWGQPHGGGWGQPHGGGWGQ---
---MANLGCWMLVLFVATWSDLGLCKKRPKP-GGWNTGGSRYPGQGSPGGNLYPPQ-GGG-WGQPHGGGWGQPHGGGWGQPHGGSWGQPHG
---MANLGCWMLVLFVATWSDLGLCKKRPKP-GGWNTGGSRYPGQGSPGGNRYPPQ-GGGGWGQPHGGGWGQPHGGGWGQPHGGGWGQPHG
---MANLGCWMLVLFVATWSNLGLCKKRPKP-GGWNTGGSRYPGQGSPGGNRYPPQ-GGGGWGQPHGGGWGQPHGGGWGQPHGGGWGQPHG
---MANLGCWMLVLFVATWSDLGLCKKRPKP-GGWNTGGSRYPGQGSPGGNRYPPQ-GGGGWGQPHGGGWGQPHGGGWGQPHGGGWGQPHG
---MANLGCWMLVLFVATWSDLGLCKKRPKP-GGWNTGGSRYPGQGSPGGNRYPPQ-GGGGWGQPHGGGWGQPHGGGWGQPHGGGWGQPHG
MVK-SHIGSWILVLFVAMWSDVGLCKKRPKPGGGWNTGGSRYPGQGSPGGNRYPPQ-GGGGWGQPHGGGWGQPHGGGWGQPHGGGWGQPHG
MVK-SHIGSWILVLFVAMWSDVGLCKKRPKPGGGWNTGGSRYPGQGSPGGNRYPPQ-GGGGWGQPHGGGWGQPHGGGWGQPHGGGWGQPHG
MVK-SHIGSWILVLFVAMWSDVGLCKKRPKPGGGWNTGGSRYPGQGSPGGNRYPPQ-GGGGWGQPHGGGWGQPHGGGWGQPHGGGWGQPHG
--------------------------------GGWNTGGSRYPGQGSPGGNRYPPQ-SGGTWGQPHGGGWGQPHGGGWGQPHGGGWGQPHG
---MANLSYWLLALFVAMWTDVGLCKKRPKP-GGWNTGGSRYPGQGSPGGNRYPPQ-GGGTWGQPHGGGWGQPHGGGWGQPHGGGWGQPHG
---MANLGYWLLALFVTMWTDVGLCKKRPKP-GGWNTGGSRYPGQGSPGGNRYPPQ-GGT-WGQPHGGGWGQPHGGSWGQPHGGSWGQPHG

GG--WGQ---------GGGTHNQWNKPSKPKTNMKHMAGAAAAGAVVGGLGGYMLGSAMSRPLIHFGNDYEDRYYRENMYRYPNQVYYRPV
GGg-WGQ---------GGGSHGQWGKPSKPKTNMKHVAGAAAAGAVVGGLGGYMLGSAMSRPLIHFGNDYEDRYYRENMYRYPNQVYYKPV
GG--WGQ---------GGGTHNQWNKPSKPKTNMKHMAGAAAAGAVVGGLGGYMLGSAMSRPLIHFGNDYEDRYYRENMYRYPNQVYYRPV
GG--WGQ---------GGGTHNQWHKPNKPKTSMKHMAGAAAAGAVVGGLGGYMLGSAMSRPLIHFGNDYEDRYYRENMYRYPNQVYYRPV
GG--WGQ---------GGGTHSQWNKPSKPKSNMKHMAGAAAAGAVVGGLGGYMLGSAMSRPLIHFGNDYEDRYYRENMYRYPNQVYYRPV
GG--WGQ---------GGGTHNQWHKPSKPKTSMKHMAGAAAAGAVVGGLGGYMLGSAMSRPLIHFGNDYEDRYYRENMYRYPNQVYYRPV
----------------GGGTHNQWHKPSKPKTSMKHMAGAAAAGAVVGGLGGYMLGSAMSRPLIHFGNDYEDRYYRENMYRYPNQVYYRPV
GG--WGQ---------GGGTHNQWNKPSKPKTSMKHVAGAAAAGAVVGGLGGYMLGSAMSRPLIHFGNDYEDRYYRENMYRYPNQVYYRPV
GG--WGQ---------GGGTHSQWNKPSKPKTNMKHMAGAAAAGAVVGGLGGYMLGSAMSRPIIHFGSDYEDRYYRENMHRYPNQVYYRPM
GG--WGQ---------GGGTHSQWNKPSKPKTNMKHMAGAAAAGAVVGGLGGYMLGSAMSRPIIHFGNDYEDRYYRENMYRYPNQVYYRPV
GG--WGQ---------GGGTHSQWNKPSKPKTNMKHMAGAAAAGAVVGGLGGYMLGSAMSRPIIHFGSDYEDRYYRENMHRYPNQVYYRPM
GG--WGQ---------GGGTHSQWNKPSKPKTNMKHMAGAAAAGAVVGGLGGYMLGSAMSRPIIHFGSDYEDRYYRENMHRYPNQVYYRPM
GG--WGQphggggwgq-GGTHGQWNKPSKPKTNMKHVAGAAAAGAVVGGLGGYMLGSAMSRPLIHFGSDYEDRYYRENMHRYPNQVYYRPV
GG-GWGQ----------GGSHSQWNKPSKPKTNMKHVAGAAAAGAVVGGLGGYMLGSAMSRPLIHFGNDYEDRYYRENMYRYPNQVYYRPV
GG-GWGQ----------GGTHSQWNKPSKPKTNMKHVAGAAAAGAVVGGLGGYMLGSAMNRPLIHFGNDYEDRYYRENMYRYPNQVYYRPV
GG--WSQ---------GGGTHNQWNKPSKPKTNLKHVAGAAAAGAVVGGLGGYMLGSAMSRPMLHFGNDWEDRYYRENMYRYPNQVYYRPV
GG--WGQ---------GGGTHNQWNKPSKPKTNMKHMAGAAAAGAVVGGLGGYMLGSAMSRPMMHFGNDWEDRYYRENMNRYPNQVYYRPV
GG--WGQ---------GGGTHNQWNKPSKPKTNLKHVAGAAAAGAVVGGLGGYMLGSAMSRPMIHFGNDWEDRYYRENMYRYPNQVYYRPV

DQYSNQNNFVHDCVNITIKQHTVTTTTKGENFTETDVKIMERVVEQMCITQYEKESQAYY----QRGSSMVLFSSPPVILLISFL------
DQYNNQNNFVHDCVNITVKQHTVTTTTKGENFTETDMKIMERVVEQMCVTQYQRESEAYY----QRGASAILFSPPPVILLISLLILLIVG
DQYSNQNNFVHDCVNITIKQHTVTTTTKGENFTETDVKMMERVVEQMCITQYERESQAYY----QRGSSMVLFSSPPVILLISFLI-----
DQYSNQNNFVHDCVNITIKQHTVTTTTKGENFTETDVKMMERVVEQMCITQYEKESQAYY----QRGSSMVLFSSPPVILLISFLI-----
DQYSNQNNFVHDCVNITIKQHTVTTTTKGENFTETDVKMMERVVEQMCITQYEKESQAYY----QRGSSMVFFSSPPVILLISFLIFLIVG
DQYSNQNNFVHDCVNITIKQHTVTTTTKGENFTETDVKMMERVVEQMCITQYEKESQAYY----QRGSSMVLFSSPPVILLISFLIFLIVG
DQYSNQNNFVHDCVNITIKQHTVTTTTKGENFTETDVKMMERVVEQMCITQYEKESQAYY----QRGSSMVLFSSPPVILLISFLIFLIVG
DQYSNQNNFVHDCVNITIKQHTVTTTTKGENFTETDVKMMERVVEQMCITQYERESQAYY----QRGSSMVLFSSPPVILLISFLIFLIVG
DQYSNQNNFVHDCVNITIKQHTVTTTTKGENFTETDVKMMERVVEQMCITQYERESQAYY----QRGSSMVLFSSPPVILLISFLIFLIVG
DQYSNQNNFVHDCVNITIKQHTVTTTTKGENFTETDVKMMERVVEQMCITQYERESQAYY----QRGSSMVLFSSPPVILLISFLIFLIVG
DQYSNQNNFVHDCVNITIKQHTVTTTTKGENFTETDVKMMERVVEQMCITQYERESQAYY----QRGSSMVLFSSPPVILLISFLIFLIVG
DEYSNQNNFVHDCVNITIKQHTVTTTTKGENFTETDVKMMERVVEQMCITQYERESQAYY----QRGSSMVLFSSPPVILLISFLIFLIVG
DQYSNQNNFVHDCVNITVKEHTVTTTTKGENFTETDIKMMERVVEQMCITQYQRESQAYY----QRGASVILFSSPPVILLISFLIFLIVG
DRYSNQNNFVHDCVNITVKQHTVTTTTKGENFTETDIKIMERVVEQMCITQYQRESQAYY----QRGASVILFSSPPVILLISFLIFLIVG
DQYNNQNTFVHDCVNITVKQHTVTTTTKGENFTETDIKMMERVVEQMCITQYQRESQAYY----QRGASVILFSSPPVILLISFLIFLIVG
DQYSNQNNFVHDCVNITIKQHTVTTTTKGENFTETDVKMMERVVEQMCVTQYQKESQAYY--DG-RRSSAVLFSSPPVILLISFLIFLIVG
DQYNNQNNFVHDCVNITIKQHTVTTTTKGENFTETDIKIMERVVEQMCTTQYQKESQAYY--DG-RRSSAVLFSSPPVILLISFLIFLMVG
DQYSNQNNFVHDCVNITIKQHTVTTTTKGENFTETDVKMMERVVEQMCVTQYQKESQAYYdg--RRSSSTVLFSSPPVILLISFLIFLIVG
```

Fig. 17. Optimal alignment of 18 prion protein sequences. Input was generated using DIALIGN.

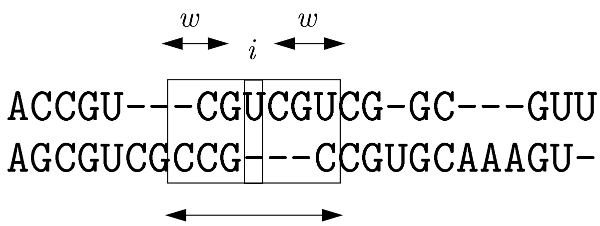
The weight w_d of a diagonal d is defined as follows: Let l_d be the length of the diagonal and s_d be the sum of the individual similarity values of residue pairs within this diagonal. Let $P(l_d, s_d)$ be the probability that a random diagonal of the same length l_d has at least the same sum s_d of similarity values. Then w_d is defined to be $-\log P(l_d, s_d)$. For a more in depth treatment see [26].

Blocks computed by LOCAL:
In this approach we first proceed as follows for all pairs of sequences. First we compute an optimal local alignment with affine gap costs. This naturally gives rise to a number of blocks by cutting the alignment at the gapped positions and taking the consecutive runs of (mis)matches as a block. Then we continue to compute the next best local alignment between these two sequences that shares no matches or mismatches with alignments already output. We stop this procedure when the length of the local alignments falls below a given value. For a pair of sequences we now have a collection of diagonals stemming from "good" local alignments not sharing a common (mis)match. The score w_d of a diagonal d is defined as follows: Let l_d be the length of the diagonal and m_d be the number of matching residue pairs within this diagonal. Let $P'(l_d, m_d)$ be the probability that a random diagonal of the same length l_d has at least the m_d matches. Then w_d is defined to be $-\log P'(l_d, m_d)$.

As input sequences we used a subset of the dataset of McClure et al. [24] and two samples of 15 respectively 18 prion proteins from the SWISSPROT database. All tests were conducted on a single processor of a Sun Enterprise 10000. The prion 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, however, normally is 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. On the other hand, the branch-and-cut algorithm produced the alignment shown in Figure 15 in 24 minutes. Figures 16 and 17 show the result of two runs of our algorithm that indicate the quality of the two greedy heuristics used in DIALIGN and LOCAL. The first figure shows an alignment of a set of six globin sequences, where the input was generated using the LOCAL procedure. The second figure shows an alignment of eighteen prion sequences where the input was generated using DIALIGN. In the first case all five motifs that are used by McClure et al. for evaluating the quality of an alignment are perfectly aligned. Small letters indicate that the respective residue is not contained in any diagonal. Figure 16 shows that the value of an optimal solution (2067) is much higher than the value of the heuristic solution which yields only a score of 1278. This shows that the pure greedy approach we use in LOCAL yields poor results. The alignment was computed in 36 seconds.

On the other hand, in Figure 17 one can see that improving the greedy approach with further heuristics yields good lower bounds that allow to tackle problems bigger size. The alignment of 18 sequences was computed in 144 minutes. It shows that our optimal solution (59862.5) is significantly better than the lower bound obtained by the heuristic alignment (53475).

The generation of the RSA graph
One input for our algorithm is a set of reasonable alignment edges. In principle, we use for this purpose alignment edges realized by some suboptimal alignment, i.e. an alignment with a score close to optimal. In contrast to [21] we do not take all the edges realized by any suboptimal alignments scoring better than a fixed threshold s below the optimal. Rather we employ a windowing technique to make the alignment graph denser in certain regions and thinner in others. The reason for applying the windowing technique described



Match=2, Mismatch=-1, Quotient = 3/9

Fig. 18. Calculation of quotient at position i with window of width 4

below is due to the fact that conventional suboptimal alignments have frequently shown insufficient deviation from the optimal alignment to cover the alignment edges necessary to build the structurally correct alignment.

On the other hand, upon inclusion of a sufficient number of suboptimal alignments the number of edges to consider became too large. As a remedy we designed a windowing technique that adjusts the suboptimality cutoff according to the local quality of an alignment. Where the alignment appears to be very good no suboptimal alternatives are considered. In alignment regions showing little sequence conservation more suboptimal alternatives are taken into account.

We proceed as follows: For a given conventional optimal alignment we compute for each position i in the first sequence, say, an index $q(i)$. Let $a(i)$ be the position of character i in the alignment and l be the length of the first sequence. Then, for a given window size w, we sum the substitution matrix values of the aligned characters from alignment position $\max\{0, a(i) - w\}$ to alignment position $\min\{a(l), a(i) + w\}$ and divide it by the length of the window. The index $q(i)$ is a measure for the local quality of the optimal alignment at sequence position i. See Figure 18 for an example with window width 4. Now we compute a coefficient $c(i)$ as follows: First we normalize $q(i)$ to a value $p(i)$ between 0 and 1 and then define $c(i) = (1 - p(i))^2$. The coefficient $c(i)$ is near 0 in regions where the alignment is reliable and near 1 in regions where it is not reliable. Figure 19 shows a plot of $c(i)$ for the optimal conventional

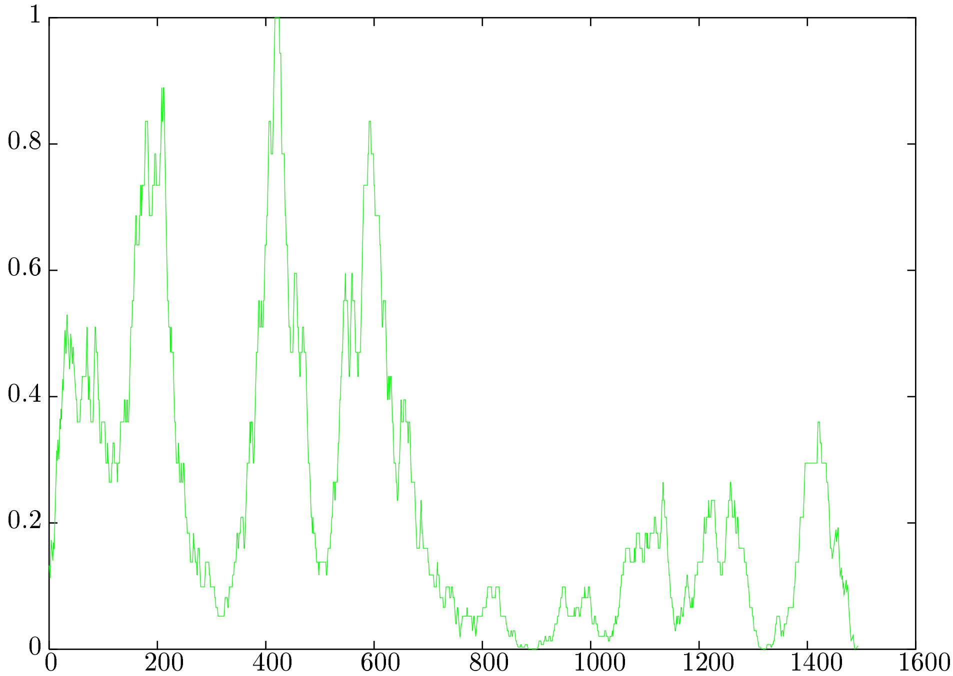


Fig. 19. Plot of $c(i)$ for an optimal conventional alignment.

alignment between Desulfurococcus Mobilis (1495 nucleotides) and Halobacterium Halobium (1472 nucleotides). One can see three prominent peaks where indeed our experiments show that the conventional alignment is wrong.

Finally, we collect all edges at position i that are realized by some alignment that is at most $c(i) \cdot s$ worse than the optimal conventional alignment where s is the (maximal) suboptimality. The alignment edges induce base pair edges for each of the pairs A-U, C-G and G-U as described in Section 2 and together with the base pair edges they form the input RSA graph.

Assessing the quality of the results

The given secondary structure should direct the optimal alignment towards the detection of conserved structural patterns. We used two ways of assessing the quality of a structural alignment.

- The first is the number of realized base pairs. We compare it to the number of standard base pairs (A-U, C-G, G-U) in the correct structure of the second sequence. The more base pairs we realize, the better the alignment.
- The second is the comparison of the sequence alignment from the database (which we assume to be the "correct" alignment) with the computed alignment. We use a dot plot representation to overlay the correct alignment with either the optimal conventional alignment or our structural alignment. A mark at position (i, j) in the plot indicates that the i-th character of the first sequence is aligned with the j-th character of the second sequence. The more (mis)matches of an alignment coincide with the (mis)matches from the correct alignment, the better the alignment.

Results
We initially tested the algorithm on small problem instances like tRNA alignments and alignments of 5S RNA sequences. For the cases studied the algorithm reproduced the correct alignments. Here we want to present some more challenging examples of 23S ribosomal RNA sequences from the Antwerpen rRNA database [6]. The base pairs for the first sequence were taken from the common secondary structure given in the database.

We present results for three sequences taken from the database:

(1) **Desulfurococcus Mobilis** with 1496 nucleotides. The common structure in the database realizes 469 base pairs of which 464 are standard base pairs.
(2) **Halobacterium Halobium** with 1474 nucleotides. The common structure in the database realizes 459 base pairs of which 452 are standard base pairs.
(3) **Methanobacterium Formicicum** with 1477 nucleotides. The common structure in the database realizes 455 base pairs of which 447 are standard base pairs.

From these three sequences we build three test sets, where the structure is always given for the first sequence. The first alignment we compute is between Desulfurococcus Mobilis and Halobacterium Halobium. The second between Halobacterium Halobium and Methanobacterium Formicicum and finally the third between Methanobacterium Formicicum and Desulfurococcus Mobilis. Figure 5.2 contains the results of different runs of our algorithm on these examples. We computed the conventional alignment with affine gap costs. The gap initiation penalty was 6 and the gap prolongation penalty 3. The substitution matrix we use assigns a score of 4 to a match and 1 to a mismatch In the computation of $c(i)$ we additionally assign the score of -1 to an indel. The first column shows the number of the test set. The second column gives the degree of suboptimality. If this number is too high there are too many alignment and base pair edges in the RSA graph and the problem becomes hard to solve. If this number is too small we are in danger of loosing too much information. The third column contains the run-time in minutes and seconds on an UltraSparc 2/200. Note that the numbers include the time of computing the conventional alignment which in our case is around 30s. The fourth column gives the total number of edges (or variables) and the last column shows the number of base pairs that are realized by the optimal RSA alignment.

As expected the number of realized base pairs increases with a higher suboptimality parameter except for test set 3. Here the conventional alignment seems to be reasonable and is not improved by our method. In the appendix we present a region of an alignment between Desulfurococcus Mobilis and Halobacterium Halobium computed in two minutes with suboptimality 10. In

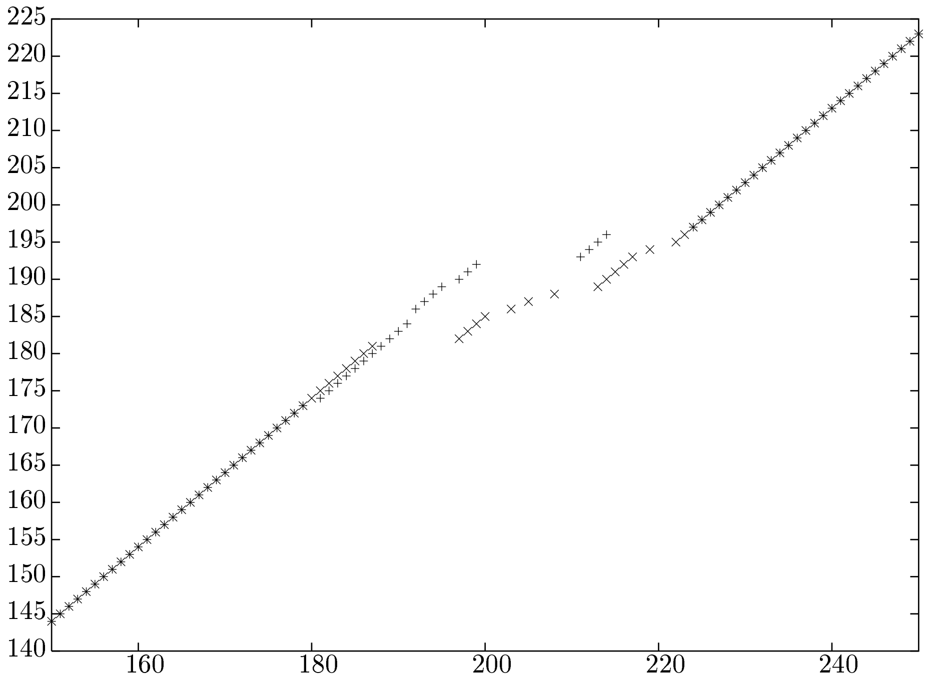
Test set	Suboptimality	Time	# Variables	# Realized base pairs
1	0	00:32	2051	436
1	5	00:45	2399	440
1	10	02:02	3091	446
1	13	20:26	3659	446
2	0	00:28	1925	436
2	10	00:34	2078	438
2	20	01:01	2857	441
2	30	02:11	4039	445
2	40	16:23	5643	445
3	0	00:30	2002	445
3	10	00:59	2855	445
3	20	02:04	3664	445
3	30	04:24	5087	445

Fig. 20. Results of the algorithm with different levels of suboptimality

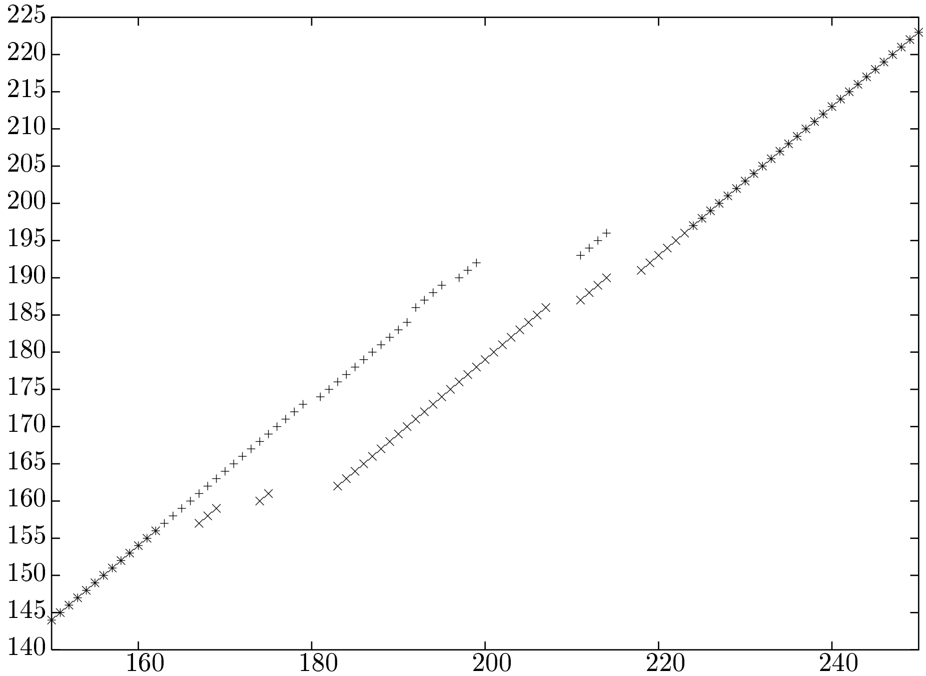
Figure 21 you see two dot plots. The correct alignment taken from the database is denoted with crosses. Using an 'x' as mark, we plotted in Figure 21 a) the (mis)matches from the structural alignment obtained by our algorithm and in Figure 21 b) the (mis)matches from all optimal conventional alignments. Our alignment coincides to a much larger degree with the handmade alignment than the optimal conventional one. Figure 22 depicts the alignment itself in this region. We show the optimal handmade, the optimal conventional and the optimal structural alignment. The numbers in the first and fourth row indicate structural elements (helices) where x and x' are complementary strands of a helix numbered x. Capital letters show bases that form a base pair with a base in the complementary strand. Small letters are either not part of a helix or form a bulge within a helix. We inserted square brackets into the alignment to indicate the beginning and end of a helix. They are not part of the alignment.

The first two rows in all three alignments show part of Desulfurococcus Mobilis sequence with three helices $9, 9'$, $10, 10'$ and $11, 11'$. It is easy to check that the capital letters read from the beginning from x build a base pair with the capital letters read from the end of x'.

The last two rows in all three alignments always show a part of Halobacterium Halobium with the same three helices $9, 9'$, $10, 10'$ and $11, 11'$. The optimal



a) Plot of optimal alignment against computed structural alignment



b) Plot of optimal alignment against computed conventional alignment

Fig. 21. Dot plots of optimal alignments

handmade alignment shows that helix 9 has the same length in both sequences, helix 10 is a bit shorter and helix 11 is considerably shorter in Halobacterium Halobium. The optimal conventional alignment identifies helix 9, however, it

```
                 Correct alignment

-----9------------------9'------------------------10----------
[GGGGgauaaCACCGG]gaaa[CUGGUGcuaaucCCCC]aua[GG GGAGGAGgCC]uggaag
[CGGGaauacUCUCGG]gaaa[CUGAGGcuaaucCCCG]aua ac[GCUUUGCUCC]uggaa-
-----9------------------9'------------------------10----------

-------10'----------11-----------------------------11'--
[GGUUCCUCCC C]-gaaag[GGU GUGGCaggGGU]uaac[GCUGCUACA CC]g
[GGGGCAAAGC]c ggaaa-[CGC]----------  uccg ---------[GC G]
-------10'----------11-----------------------------11'--
```

```
             Optimal conventional alignment

-----9------------------9'------------------------10----------
[GGGGgauaaCACCGG]gaaa[CUGGUGcuaaucCCCC]aua[GGGGAG GAgGCC]uggaag
[CGGGaauacUCUCGG]gaaa[CUGAGGcuaaucCCCG]aua ----ac[g----c u-----
-----9------------------9'------------------------10----------

-------10'--------------11----------------------11'-------
[GGUUCCUCC CC]gaa ag[GGUGUGGC aggGGU]uaa c[GCUGC UACA CC]g
--uugcucc]ug gaa[gg ggcaaagc]cgg--- aaa[c ---gc]uccg[gc g]
---------------------10'-----------------11-----------11'-
```

```
              Optimal structural alignment

------9------------------9'--------------------10--------------
[GGGGgauaaCACCGG]gaaa[CUGGUGcuaaucCCCC]aua[GG GGAGGAgGCC]uggaa g
[CGGGaauacUCUCGG]gaaa[CUGAGGcuaaucCCCG]aua ac[GCUUUGCUCC]uggaa[G
------9------------------9'--------------------10--------------

-------10'-----------11----------------------11'---------
[GGUUCCUCCCC]gaaag[GG UGUGGCaggGGU]uaa c[GC UGCUACA CC]g
 GGGCAAA---- ----- gc]cg--g-a--a-- --a[c gc]U-C--CG[gc g]
-------10'------------------------------11----------11'--
```

Fig. 22. Alignments corresponding to the dotplot in Figure 21

completely fails to recognize helices 10 and 11. A closer look at helix 10 in the optimal handmade alignment reveals that there is indeed a very poor sequence similarity at this position. Inspecting the optimal structural alignment we can observe that helix 10 is almost completely identified.

6 Conclusion

In this paper we formulated two multiple sequence alignment problems using polyhedral combinatorics. We described several classes of facet-defining inequalities for both the RSA and GMT polytope. Additionally we gave a complete description of the MT polytope which implies a polynomial time algorithm for sequence alignment not based on dynamic programming. We implemented branch-and-cut algorithms for the GMT and RSA problem. Our computational results show that we are able to solve problem instances to optimality, the size of which is not tractable for dynamic programming based approaches. Sophisticated implementations of such approaches such as MSA [12] or GSA [22] cannot possibly solve non-trivial problem instances of 18 sequences. This is due to the exponential space consumption of dynamic programming. Although both programs can compute an alignment of guaranteed optimality, by default they artifically reduce the explored part of the k-dimensional dynamic programming matrix, thereby loosing the guarantee of optimality. In addition the second program uses only linear gap costs and goal-directed search to speed up the computation. But even then the above statement stays true except for trivial examples.

We view as one of the contributions of our work the introduction of the polyhedral approach to the area of sequence alignment. With a polyhedral approach, variations of a basic problem can often be conveniently modeled through the addition of further constraints to the basic linear program. We demonstrated this with the formulation of the RNA secondary structure alignment problem and showed empirically that the RSA problem models the "biological truth" better than conventional sequence alignment with affine gap costs.

We do not compare our approach to semi-automatic methods or algorithms that do not optimally solve the problems. Although we are aware that such methods exist and do work well, we think a thorough discussion would not be in the scope of this paper.

Further research is needed in order to develop separation algorithms for classes of facet-defining inequalities for the GMT and RSA polytope. This would immediately imply faster algorithms for both the GMT and RSA problem. It should be mentioned, however, that this is no mean feat. It involves many steps from identifying facet-defining inequalities over devising efficient separation algorithms for them to finally incorporating such algorithms in a branch-and-cut framework. Also it is not clear whether the polyhedral approach will be equally successful for sequence alignment as it is for other combinatorial problems, such as the TSP. Nevertheless we see a lot of potential in our method compared to the standard dynamic programming approach which has already been studied thoroughly and is hard to improve.

Acknowledgments

The authors would like to thank Pavel Pevzner for pointing out an alternative proof to Lemma 15.

References

[1] S. F. Altschul and B. W. Erickson. Locally optimal subalignments using nonlinear similarity functions. *Bull. Math. Biol.*, 48:633–660, 1986.

[2] D. Applegate, R. Bixby, V. Chvátal, and B. Cook. Finding cuts in the TSP. DIMACS Technical Report 95-05, DIMACS, April 1995.

[3] V. Bafna, S. Muthukrishnan, and R. Ravi. Computing similarity between RNA strings. In Z. Galil and E. Ukkonen, editors, *Proceedings of the Sixth Annual Symposium on Combinatorial Pattern Matching (CPM-95)*, Lecture Notes in Computer Science 937, pages 1–16, Berlin, 1995. Springer.

[4] F. Corpet and B. Michot. RNAlign program: alignment of RNA sequences using both primary and secondary structures. *CABIOS*, 10(4):389–399, 1994.

[5] M. O. Dayhoff, R. M. Schwartz, and B. C. Orcut. A model of evolutionary change in proteins. In M. O. Dayhoff, editor, *Atlas of Proteins Sequence and Structure*, number 5, pages 345–352. National Biomedical Research Foundation, Washington, D.C., 1979.

[6] P. de Rijk, Y. Van de Peer, S. Chapelle, and R. de Wachter. Database on the structure of the small ribosomal subunit RNA. *Nucleic Acids Research*, 22:3495–3501, 1994.

[7] S. R. Eddy and R. Durbin. RNA sequence analysis using covariance models. *Nucleic Acids Research*, 22(11):2079–2088, 1994.

[8] J. Gorodkin, L. J. Heyer, and G. D. Stormo. Finding the most significant common sequence and structure motifs in a set of RNA sequences. *Nucleic Acids Research*, 25:3724–3732, 1997.

[9] M. Grötschel, M. Jünger, and G. Reinelt. A cutting plane algorithm for the linear ordering problem. *Operations Research*, 32:1195–1220, 1984.

[10] M. Grötschel, L. Lovász, and A. Schrijver. The ellipsoid method and its consequences in combinatorial optimization. *Combinatorica*, 1:169–197, 1981.

[11] M. Grötschel and M.W. Padberg. *Polyhedral Theory*. Wiley-interscience series in discrete mathematics and optimization. Wiley, Chichester, 1985.

[12] S.K. Gupta, J.D. Kececioglu, and A.A. Schaeffer. Improving the practical space and time efficiency of the shortest-paths approach to sum-of-pairs multiple sequence alignment. *J. Comput. Biol.*, 2:459–472, 1995.

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

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

[15] M. Jünger, G. Reinelt, and G. Rinaldi. The traveling salesman problem. In M. Ball, T. Magnanti, C. L. Monma, and G. L. Nemhauser, editors, *Handbook on Operations Research and Management Science*, pages 225–330. Elsevier, North Holland, 1995.

[16] M. Jünger, G. Reinelt, and S. Thienel. Practical problem solving with cutting plane algorithms in combinatorial optimization. In W. Cook, L. Lovász, and P. Seymour, editors, *Combinatorial Optimization: Papers from the DIMACS Special Year*, number 20 in DIMACS Series in Discrete Mathematics and Theoretical Computer Science, pages 111–152. AMS, Providence, RI, 1995.

[17] M. Jünger and S. Thienel. The design of the branch and cut system ABACUS. Technical Report 97.260, Institut für Informatik, Universität zu Köln, 1997.

[18] R. M. Karp and C. H. Papadimitriou. On linear characterizations of combinatorial optimization problems. *SIAM J. Computing*, 11:620–632, 1982.

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

[20] J. D. Kececioglu. PRIMAL: Practical rigorous multiple alignment. Computer software, 1996.

[21] H.-P. Lenhof, K. Reinert, and M. Vingron. A polyhedral approach to RNA sequence structure alignment. In *Proceedings of the Second Annual International Conference on Computational Molecular Biology (RECOMB-98)*, pages 153–162, New York, USA, 1998. ACM Press.

[22] M. Lermen and K. Reinert. The practical use of the $\mathcal{A}^*$ algorithm for exact multiple sequence alignment. Research report MPI-I-97-1-028, Max-Planck-Institut für Informatik, Saarbrücken, 1997.

[23] M. Levitt. Detailed molecular model for transfer ribonucleic acid. *Nature*, 224:759–763, 1969.

[24] M. McClure, T. K. Vasi, and W. M. Fitch. Comparative analysis of multiple protein-sequence alignment methods. *Mol. Biol. Evol.*, 4(11):571–592, 1994.

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

[26] B. Morgenstern, W. R. Atchley, K. Hahn, and A. Dress. Segment-based scores for pairwise and multiple sequence alignments. In *Proceedings of the Sixth International Conference on Intelligent Systems for Molecular Biology (ISMB-98), (in press)*, 1998.

[27] B. Morgenstern, A. Dress, and T. Werner. Multiple DNA and protein sequence alignment based on segment-to-segment comparison. *Proc. Natl. Acad. Sci.*, 93:12098–12103, 1996.

[28] S. B. Needleman and C. D. Wunsch. A general method applicable to the search for similarities in the amino-acid sequence of two proteins. *J. Mol. Biol.*, 48:443–453, 1970.

[29] G. L. Nemhauser, A. Kan, H. G. Rinnooy, and M. J. Todd, editors. *Optimization*, volume 1 of *Handbooks in Operations Research and Management Science*. North-Holland, Amsterdam, 1989.

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

[31] C. Notredame, E. A. O'Brien, and D. G. Higgins. RAGA: RNA sequence alignment by genetic algorithm. *Nucleic Acids Research*, 25:4570–4580, 1997.

[32] M. W. Padberg and G. Rinaldi. Optimization of a 532 city symmetric traveling salesman problem by branch and cut. *Operations Research Letters*, 6:1–7, 1987.

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

[34] K. Reinert. *A Polyhedral Approach to Sequence Alignment Problems.* PhD thesis, submitted to the Universität des Saarlandes, Im Stadtwald, 66123 Saarbrücken, Germany, 1999.

[35] D. Sankoff. Simultaneous solution of the RNA folding, alignment and protosequence problems. *SIAM J. Applied Mathematics*, 45(5):810–825, 1985.

[36] D. Sankoff and J. B. Kruskal. *Time Warps, String Edits and Macromolecules: the Theory and Practice of Sequence Comparison.* Addison Wesley, 1983.

[37] A. Schrijver. *Theory of Linear and Integer Programming.* Wiley-interscience series in discrete mathematics and optimization. Wiley, Chichester, 1986.

[38] S. Tsukiyama, M. Ide, H. Ariyoshi, and I. Shirawaka. A new algorithm for generating all the maximal independent sets. *SIAM J. Computing*, 6:505–517, 1977.

[39] M. Vingron and P. Pevzner. Multiple sequence comparison and consistency on multipartite graphs. *Adv. Appl. Math.*, 16:1–22, 1995.

[40] M. S. Waterman. Consensus methods for folding single-stranded nucleic acids. In *Mathematical Methods for DNA Sequences*, pages 185–224. CRC Press, Boca Raton, FL, 1989.

[41] W. J. Wilbur and D. J. Lipman. The context dependent comparison of biological sequences. *SIAM J. Applied Mathematics*, 44(3):557–567, 1984.