Math. Program., Ser. B 105, 387–425 (2006)

Digital Object Identifier (DOI) 10.1007/s10107-005-0659-3

Ernst Althaus · Alberto Caprara · Hans-Peter Lenhof · Knut Reinert

A branch-and-cut algorithm for multiple sequence alignment

Received: April, 2004 / Revised version: June 6, 2005
Published online: 10 November 2005 – © Springer-Verlag 2005

Abstract. We consider a branch-and-cut approach for solving the *multiple sequence alignment problem*, which is a central problem in computational biology. We propose a general model for this problem in which arbitrary gap costs are allowed. An interesting aspect of our approach is that the three (exponentially large) classes of natural valid inequalities that we consider turn out to be both facet-defining for the convex hull of integer solutions and separable in polynomial time. Both the proofs that these classes of valid inequalities are facet-defining and the description of the separation algorithms are far from trivial. Experimental results on several benchmark instances show that our method outperforms the best tools developed so far, in that it produces alignments that are better from a biological point of view. A noteworthy outcome of the results is the effectiveness of using branch-and-cut with only a carefully-selected subset of the variables as a heuristic.

1. Introduction

Aligning DNA or protein sequences is one of the most important and predominant problems in computational molecular biology. The associated *multiple sequence alignment problem* can be formally defined as follows. Let $S = \{s^1, s^2, \dots, s^k\}$ be a set of k strings over an alphabet Σ and let $\bar{\Sigma} = \Sigma \cup \{-\}$, where "$-$" (dash) is a symbol used to represent "gaps" in strings. Given a string s, we let $|s|$ denote the number of characters in the string and s_l the lth character of the string for $l = 1, \dots, |s|$. We will assume that $|s^i| \geq 4$ for all strings s^i and let $n := \sum_{i=1}^{k} |s^i|$ be the overall number of characters in the strings.

An *alignment* of S is a set $\bar{S} = \{\bar{s}^1, \bar{s}^2, \cdots, \bar{s}^k\}$ of strings over the alphabet $\bar{\Sigma}$, where each string can be interpreted as a row of a two dimensional *alignment matrix* (for an illustration, see Figures 1, 2, and 3). The set $\bar{S}$ of strings has to satisfy the following properties: (1) the strings in $\bar{S}$ all have the same length, (2) ignoring dashes, string $\bar{s}^i$ is identical to string s^i, and (3) none of the columns of the alignment matrix is allowed to contain only dashes.

If $\bar{s}^i_l$ and $\bar{s}^j_l$ are both different from "$-$", the corresponding characters in s^i and s^j are *aligned*, and this has an associated cost or benefit depending on whether we minimize cost or maximize similarity. Moreover, a *gap* of s^i with respect to s^j is a maximal sequence $s^i_l\ s^i_{l+1}\ \dots\ s^i_m$ of characters in s^i that are aligned with dashes "$-$" in row j. Associated with each of these gaps is a cost or a negative benefit. The problem calls for an alignment whose overall cost is minimized or whose overall benefit is maximized.

E. Althaus: Universität des Saarlandes, Im Stadtwald, 66123 Saarbrücken, Germany

A. Caprara: DEIS, Universitá di Bologna, Viale Risorgimento 2, 40136 Bologna, Italy

H.-P. Lenhof: Universität des Saarlandes, Im Stadtwald, 66123 Saarbrücken, Germany

K. Reinert: Institute of Computer Science, Freie Universität Berlin, Takustr. 9, 14195 Berlin, Germany

```
C T A C G T G G A T C        C T A C G T G G A T C
C T A - G - G - A T C        C T A - - - G G A T C
```

Fig. 1. Two possible pairwise alignments of DNA fragments

For example, consider Figure 1, in which two alignments are shown for two DNA fragments. In the case of DNA, the strings are drawn from a four-letter alphabet representing the four basic building blocks of DNA, the nucleic acids adenine (A), cytosine (C), guanine (G), and thymine (T). Typically, a simple scoring scheme is chosen to align DNA sequences, say $+2$ for a match and -1 for a mismatch. If, in addition, each gap position is penalized by a constant, the two alignments in Figure 1 have the same score since they both have three characters in the first string aligned with gap characters. However, a single, longer gap is much more likely to arise in reality because it might be caused by a single mutational event resulting in a polymorphism in the genome. Hence, the left alignment should be scored worse than the right one with one single gap. Additionally, the length of the gap should have a smaller impact on the score than the number of gaps. We can achieve this by using convex gap cost functions like $a + b \log l$, where l is the length of the gap. In practice, affine gap costs of the form $a + bl$ are often used since the resulting alignment problems are easier to solve.

Alignment programs are still among the most important bioinformatics tools, with a large number of applications. Pairwise alignments, for example, are mostly used to find strings in a database that share certain commonalities with a query sequence, but which might not be known to be biologically related. Multiple alignments serve a different purpose, as they are used to solve problems that are *inverse* with respect to the ones addressed by pairwise string comparisons [15]. The inverse problem is to infer certain shared patterns from known biological relationships. An example is given in Figure 2, which shows a multiple alignment of 14 proteins that are known to bind sugar. A pairwise inspection of the sequences will reveal many commonalities and some differences between the sequences. But what features do they *all* share that allow them to bind sugar? The multiple alignment shown in Figure 2 reveals that all the molecules can form four disulphid bridges using the eight sulphur-containing cysteins indicated in yellow. This is a basic feature of the three-dimensional structure of all those molecules (the structure of the hevein molecule is shown).

A key question is how a multiple alignment should be scored. The model that is used most frequently is the so called *weighted sum of pairs* score, given by the sum of the scores of the pairwise alignments induced by the multiple alignment [6]. In the example in Figure 2, the alphabet consists of the 20 amino acids, the building blocks of proteins. The scoring functions for protein alignments are usually more complicated than the ones for DNA. Amino acid substitution matrices give the log odds of replacing one amino acid by another [7, 16]. For example the score of a D (aspartic acid) staying a D is 4, only slightly higher than that of a D turning into an E (glutamic acid), while it is unlikely that a D turns into an W (tryptophan), yielding a score of -7.

If the number k of strings is fixed, the multiple sequence alignment problem for strings of length n can be solved in time and space $O(n^k)$ with (quasi)-affine gap costs [14, 21, 24, 25]. More complex gap cost functions add a polylog factor to this complexity [9, 18]. However, if the number k of strings is not fixed, [28] proved that multiple

```
AATAHAQRCG EQGSNMECPN NLCCSQYGYC GMGGDYCGKG ..CQNGACYT
VAATNAQTCG KQNDGMICPH NLCCSQFGYC GLGRDYCGTG ..CQSGACCS
VGLVSAQRCG SQGGGGTCPA LWCCSIWGWC GDSEPYCGRT ..CENK.CWS
AATAQAQRCG EQGSNMECPN NLCCSQYGYC GMGGDYCGKG ..CQNGACWT
AATAQAQRCG EQGSNMECPN NLCCSQYGYC GMGGDYCGKG ..CQNGACWT
......QRCG EQGSGMECPN NLCCSQYGYC GMGGDYCGKG ..CQNGACWT
SETVKSQNCG .......CAP NLCCSQFGYC GSTDAYCGTG ..CRSGPCRS
RGSAE..QCG RQAGDALCPG GLCCSSYGWC GTTVDYCGIG ..CQSQ.CDG
AGPAAAQNCG .......CQP NFCCSKFGYC GTTDAYCGDG ..CQSGPCRS
AGPAAAQNCG .......CQP NVCCSKFGYC GTTDEYCGDG ..CQSGPCRS
RGSAE..QCG QQAGDALCPG GLCCSSYGWC GTTADYCGDG ..CQSQ.CDG
RGSAE..QCG RQAGDALCPG GLCCSFYGWC GTTVDYCGDG ..CQSQ.CDG
TGVAIAEQCG RQAGGKLCPN NLCCSQWGWC GSTDEYCSPD HNCQSN.CK.
......EQCG RQAGGKLCPN NLCCSQYGWC GSSDDYCSPS KNCQSN.CK.
```

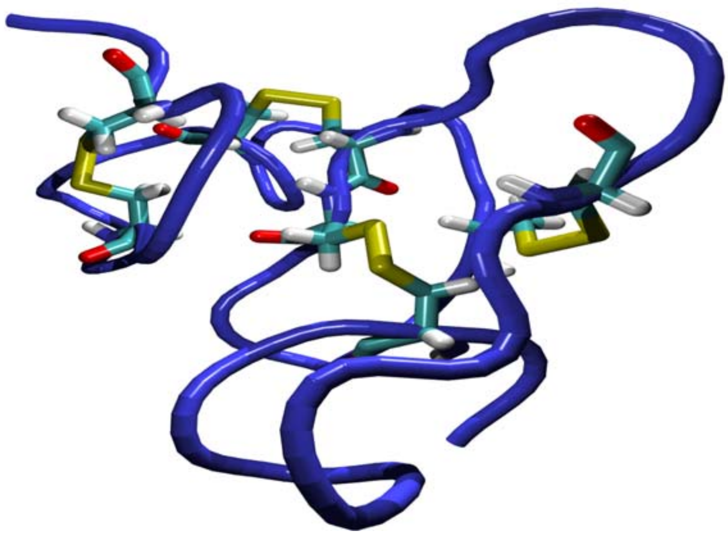

Fig. 2. Multiple alignment of N-acetylglucosamine-binding proteins and tertiary structure of one of those proteins, the hevein

sequence alignment with the sum of pair score is NP-complete by a reduction to *shortest common supersequence* [11]. Hence, it is unlikely that polynomial time algorithms exist and, depending on the problem size, various heuristics are generally applied to solve the problem approximately (see, e.g., [2, 8]).

In most alignment models proposed in the literature, the objective function depends only on the set of pairwise aligned characters. If gap functions are used, they are normally affine since this allows a faster computation with respect to more complicated convex functions [13]. In this paper we consider a general formulation of the multiple sequence alignment problem that can deal with arbitrary gap costs, which include truly affine, convex and position-dependent gap costs that were proposed by several authors in the field of computational biology (see, e.g., [9, 18]). We will see that the application of truly convex gap costs does indeed yield good results.

The new formulation presented in this paper is an extension of the *gapped trace problem* proposed by [23], which is naturally cast in graph-theoretic terms. The

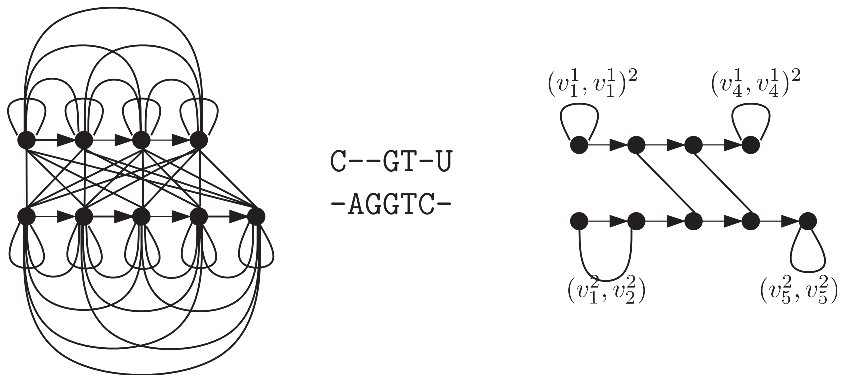

Fig. 3. Gapped alignment graph for two strings (left). Possible alignment matrix (middle) and corresponding gapped trace (right)

representation on a graph leads to an *integer linear programming* (ILP) formulation with variables associated with edges and arcs of the graph, which we solve with a branch-and-cut approach.

A nice aspect of our approach is that the three (exponentially large) classes of natural valid inequalities that we consider turn out to be both facet-defining for the convex hull of the integer solutions and separable in polynomial time. Both the proofs that the classes of valid inequalities are facet-defining and the descriptions of the separation algorithms are far from trivial.

Our method, evaluated using several benchmark libraries of alignments (e.g.,[27]), turns out to outperform the classical methods used for multiple alignment. For medium-size instances, it generally finds the best solutions from a biological viewpoint according to the benchmark evaluation programs. This provides a positive answer to a repeatedly asked question, namely whether it is worthwhile to compute near-optimal alignments using scoring functions that unavoidably only approximate the true biological phenomena. Another interesting outcome of our results is the effectiveness of using, as a heuristic, branch-and-cut with only a carefully-selected subset of the variables, mainly for the cases in which the problem, even after preprocessing, is still too large to be solved to proven optimality.

2. Formulation and valid inequalities

In this section, we describe a graph representation of the multiple sequence alignment problem and a related ILP formulation whose *linear programming* (LP) relaxation is strengthened by introducing three natural classes of valid inequalities, which are proved to be facet-defining.

2.1. A graph-theoretic model

The problem is represented using a *mixed multigraph* $G = (V, E, A)$, called the *gapped alignment graph*, where E is a set of (undirected) edges and A is a set of (directed) arcs.

The nodes in V represent the characters of the input strings, namely, for each string s^i and character s^i_l of s^i there is a corresponding node $v^i_l \in V$. Accordingly, node set V is partitioned into $V^1, \dots, V^k$, where $V^i := \{v^i_l : 1 \le l \le |s^i|\}$ is the set of nodes corresponding to characters in s^i.

The edges in E represent *alignments* of pairs of characters in different strings, namely, for each pair of strings s^i, s^j and characters s^i_l of s^i and s^j_m of s^j, an edge joining v^i_l and v^j_m, denoted by $e = \{s^i_l, s^j_m\}$, represents the fact that s^i_l and s^j_m are aligned. If this is the case, we say that the edge is *realized* by the alignment, and the corresponding cost is denoted by w_e. Accordingly, edge set E is partitioned into sets E^{ij} for $i, j = 1, \dots, k; i < j$, where $E^{ij} := \{\{v^i_l, v^j_m\} : 1 \le l \le |s^i|, 1 \le m \le |s^j|\}$ denotes the set of all edges with one endpoint in V^i and the other in V^j. Note that the undirected graph (V, E) is a complete k-partite graph with color classes $V^1, \dots, V^k$.

The arc set A is partitioned into A_p and A_g. Arcs in A_p represent *consecutivity* of characters within a string and run from each node to its "right" neighbor, namely $A_p = \{(v^i_l, v^i_{l+1}) : 1 \le i \le k, 1 \le l < |s^i|\}$. Note that the arcs in A_p are independent of the alignment. The arcs in A_g represent *gaps* in the alignment, namely for each pair of strings s^i, s^j and substring $s^i_l\ s^i_{l+1}\ \dots\ s^i_m$ of s^i, an arc from v^i_l to v^i_m, denoted by $a = (v^i_l, v^i_m)^j$, represents the fact that no character in the substring is aligned with any character in s^j, whereas both s^i_{l-1} (if $l > 1$) and s^i_{m+1} (if $m < |s^i|$) are aligned with some character in s^j. If this is the case, we say that gap arc a is *realized* and the corresponding cost is w_a. We also say that nodes $v^i_l, \dots, v^i_m$ are *spanned* by gap arc a. Accordingly, the gap arc set A_g is partitioned into sets A^{ij} for $i, j = 1, \dots, k; i \ne j$, with $A^{ij} := \{(v^i_l, v^i_m)^j : 1 \le l \le m \le |s^i|\}$. Note that for each pair of nodes v^i_l, v^i_m, $l \le m$, there are $k-1$ gap arcs in A_g from v^i_l to v^i_m, each associated with gaps of s^i with respect to different strings.

Given two gap arcs $(v^i_l, v^i_m)^j, (v^i_p, v^i_q)^j \in A^{ij}$, we say that they *conflict* if the substrings spanned by the arcs overlap or even touch, that is if $\{l, \dots, m+1\} \cap \{p, \dots, q+1\} \ne \emptyset$. The fact that the two arcs conflict even if $p = m + 1$ (or $l = q + 1$) is due to the fact that there must be at least one aligned character between consecutive gap arcs. We let $\mathcal{I}$ denote the collection of all maximal sets of pairwise conflicting gap arcs (with polynomially many elements, as also discussed in detail in the next section). Finally, we let

$$A^{ij}(l \leftrightarrow m) := \{(v^i_p, v^i_q)^j : p \le l, q \ge m\}$$

denote the set of gap arcs in A^{ij} spanning $v^i_l, \dots, v^i_m$. We also allow $l = m + 1$, i.e., use $A^{ij}(p+1 \leftrightarrow p)$ to denote the set of gap arcs that span either v^i_p or v^i_{p+1}. The latter is motivated by the necessity of representing sets of conflicting gap arcs.

We identify paths and cycles in G by their set of edges and arcs. A cycle in G is called a *mixed cycle* if it contains only arcs in A_p, and at least one arc and one edge. We stress that the arcs in the mixed cycle should be oriented so that it is possible to walk along the cycle traversing each arc according to its orientation (see Figure 6 for an illustration). A mixed cycle represents a contradictory ordering of the characters in the alignment, the most trivial mixed cycle corresponding to two alignment edges "crossing" (which is clearly not allowed). A mixed cycle is called *critical* if, for $i = 1, \dots, k$, all vertices in V^i visited by the cycle correspond to consecutive characters of s^i. It is easy to verify that a set $F \subseteq E$ contains the edges in a mixed cycle if and only if it contains the edges in a critical mixed cycle.

A pair F, B_g with $F \subseteq E$ and $B_g \subseteq A_g$ is called a *gapped trace* of G if it satisfies the following requirements:

1. For $i, j = 1, \ldots, k, i \neq j$ and $l = 1, \ldots, |s^i|$, node v_l^i is either incident to exactly one alignment edge in $F \cap E^{ij}$ or spanned by exactly one gap arc in $B_g \cap A^{ij}$.
2. There is no critical mixed cycle M such that $M \cap E \subseteq F$.
3. For $i, j = 1, \ldots, k, i \neq j$, there is no pair of conflicting gap arcs in $B^g \cap A^{ij}$.
4. If F contains two edges incident with the same node, say $\{v_{l_1}^{i_1}, v_{l_2}^{i_2}\}$ and $\{v_{l_1}^{i_1}, v_{l_3}^{i_3}\}$, then F also contains edge $\{v_{l_2}^{i_2}, v_{l_3}^{i_3}\}$.

The proof of the following proposition is essentially from [23]. However, due to its importance, we repeat it here.

Proposition 1. *There is a correspondence between gapped traces of G and alignments of S.*

Proof. Given an alignment of S, it is easy to check that a gapped trace F, B_g is obtained by letting F be the set of edges $\{v_l^i, v_m^j\}$ corresponding to aligned characters s_l^i, s_m^j and B_g be the set of gap arcs $(v_l^i, v_m^i)^j$ corresponding to maximal sequences of characters $s_l^i \ldots s_m^i$ that are not aligned with any character of s^j.

Conversely, given a gapped trace F, B_g, consider the connected components $\mathcal{C}_1, \ldots, \mathcal{C}_m$ in the graph (V, F) along with the auxiliary directed graph D with node set $\{\mathcal{C}_1, \ldots, \mathcal{C}_m\}$ and arc set $\{(\mathcal{C}_h, \mathcal{C}_i) : v_l^j \in \mathcal{C}_h \text{ and } v_{l+1}^j \in \mathcal{C}_i \text{ for some } j \in \{1, \ldots, k\} \text{ and } l \in \{1, \ldots, |s^i|\}\}$. Requirement 4 guarantees that each connected component is a clique, and requirement 2 that D is acyclic, and therefore has a topological ordering. Assume $\mathcal{C}_1, \ldots, \mathcal{C}_m$ are the nodes of D in topological order. An alignment matrix is then obtained by assigning to column i the characters corresponding to $\mathcal{C}_i$ (putting a "$-$" for the strings s^j such that $V^j \cap \mathcal{C}_i = \emptyset$). In particular, requirements 1 and 3 guarantee that the gaps in the alignment correspond to the gap arcs of B_g. □

2.2. An initial ILP model

The goal is to identify the gapped trace F, B_g such that $\sum_{e \in F} w_e + \sum_{a \in B_g} w_a$ is maximized. See Figure 3 for an example of a gapped alignment graph and an associated gapped trace. A natural model is obtained by introducing a binary *alignment variable* x_e (also denoted by $x_{\{v_l^i, v_m^j\}}$) for every edge $e = \{v_l^i, v_m^j\} \in E$, taking the value 1 if edge e is realized (i.e., if it belongs to F in the gapped trace), and a binary *gap variable* y_a (also denoted by $y_{(v_l^i, v_m^i)^j}$) for every gap arc $a = (v_l^i, v_m^i)^j \in A_g$, taking the value 1 if edge e is realized (i.e., if it belongs to set B_g in the gapped trace).

Our initial ILP formulation is given in Figure 4, where $\mathcal{M}$ denotes the (exponentially large) collection of all critical mixed cycles in G. Constraints (2), (3), (4) and (5) correspond to requirements 1, 2, 3 and 4, respectively, in the definition of gapped trace. Note that inequalities (4) are not necessary in the case of *convex* gap arc weights,

$$\max \sum_{e \in E} w_e \cdot x_e + \sum_{a \in A_g} w_a \cdot y_a \tag{1}$$

subject to

$$\sum_{m=1}^{|s^j|} x_{\{v_l^i, v_m^j\}} + \sum_{a \in A^{ij}(l \leftrightarrow l)} y_a = 1, \quad i, j = 1, \ldots, k; i \neq j; l = 1, \ldots, |s^i|, \tag{2}$$

$$\sum_{e \in M \cap E} x_e \leq |M \cap E| - 1, \quad M \in \mathcal{M}, \tag{3}$$

$$\sum_{a \in I} y_a \leq 1, \quad I \in \mathcal{I}, \tag{4}$$

$$x_{\{v_{l_1}^{i_1}, v_{l_2}^{i_2}\}} + x_{\{v_{l_1}^{i_1}, v_{l_3}^{i_3}\}} - x_{\{v_{l_2}^{i_2}, v_{l_3}^{i_3}\}} \leq 1, \quad i_r = 1, \ldots, k; i_r \neq i_{r+1};$$

$$l_r = 1, \ldots, |s^{i_r}| \; (r = 1, 2, 3), \tag{5}$$

$$x_e, y_a \in \{0, 1\}, \quad e \in E, a \in A_g. \tag{6}$$

Fig. 4. Initial ILP formulation

i.e., if $w_{(v_l^i, v_m^i)^j} \geq w_{(v_l^i, v_p^i)^j} + w_{(v_{p+1}^i, v_m^i)^j}$ for all $i, j = 1, \ldots, k, i \neq j; l, p, m = 1, \ldots, |s^i|, l \leq p < m$. Note also that transitivity inequalities involving four or more strings, e.g.,

$$x_{\{v_{l_1}^{i_1}, v_{l_2}^{i_2}\}} + x_{\{v_{l_2}^{i_2}, v_{l_3}^{i_3}\}} + x_{\{v_{l_3}^{i_3}, v_{l_4}^{i_4}\}} - x_{\{v_{l_1}^{i_1}, v_{l_4}^{i_4}\}} \leq 2,$$

are implied by (5).

The *gapped trace polytope* $\mathcal{P} \subseteq \mathbb{R}^{|E|+|A_g|}$ is the convex hull of the feasible solutions of (1–6), whereas the LP relaxation of (1–6) is obtained by replacing (6) with

$$x_e, y_a \geq 0, \quad e \in E, a \in A_g, \tag{7}$$

noting that $x_e, y_a \leq 1$ is implied by equations (2).

2.3. Valid inequalities

In order to define valid inequalities for $\mathcal{P}$, given an ordered pair of strings s^i, s^j, we characterize pairs of edges in E^{ij} and/or gap arcs in A^{ij} that are *incompatible*, i.e., the associated variables cannot both take the value 1 in a feasible solution. All these incompatibilities follow immediately from requirements 1, 2 and 3 and the corresponding constraints (2), (3) and (4) in the ILP formulation.

By requirement 2, two alignment edges in E^{ij} are incompatible if and only if they are crossing or share an endpoint, i.e., each edge $\{v_l^i, v_m^j\}$ is incompatible with all edges $\{v_p^i, v_q^j\}$ such that either $p \leq l$ and $q \geq m$ or $p \geq l$ and $q \leq m$, since realizing both edges would create a (critical) mixed cycle.

By requirement 3, two gap arcs in A^{ij} are incompatible if and only if they conflict, i.e., each gap arc $(v_l^i, v_m^i)^j$ is incompatible with all gap arcs $(v_p^i, v_q^i)^j$ such that $p \leq m + 1$ and $q \geq l - 1$.

By requirement 1, an alignment edge in E^{ij} and a gap arc in A^{ij} are incompatible if and only if one endpoint of the former is spanned by the latter, i.e., each edge $\{v^i_l, v^j_m\}$ is incompatible with all gap arcs $(v^i_p, v^i_q)^j$ such that $p \leq l \leq q$.

Note that there are additional incompatibilities between gap arcs in A^{ij} and gap arcs in A^{ji}, namely arc $(v^j_1, v^j_{|s^j|})^i \in A^{ji}$ is incompatible with all arcs in A^{ij} except $(v^i_1, v^i_{|s^i|})^j$, and arcs $(v^j_2, v^j_{|s^j|})^i$ and $(v^j_1, v^j_{|s^j|-1})^i \in A^{ji}$ are incompatible with all arcs $(v^i_l, v^i_m)^j \in A^{ij}$ such that $l > 1$ and $m < |s^i|$. However, we will focus our attention on sets of incompatible gap arcs that also conflict, i.e., belong to the same set A^{ij}.

Let H^{ij} be the *incompatibility graph* with node set $E^{ij} \cup A^{ij}$ and an edge between each pair of incompatible edges and arcs. The subgraph $H^{ij}(E^{ij})$ induced by the edge nodes is a *co-comparability* graph associated with the partial ordering $\{v^i_l, v^j_m\} \prec \{v^i_p, v^j_q\}$ if $l < p$ and $m < q$. The subgraph $H^{ij}(A^{ij})$ induced by the gap arc nodes is an *interval* graph in which every node $(v^i_l, v^i_m)^j$ corresponds to the interval $[l, m+1]$. Accordingly, both $H^{ij}(E^{ij})$ and $H^{ij}(A^{ij})$ are *perfect*. On the other hand, it is easy to show that H^{ij} itself is not perfect (the reader is referred to [12] for an introduction to perfect graphs).

The following characterization of the maximal sets of pairwise incompatible edges will be used throughout the rest of the paper. For $i, j = 1, \ldots, k, i \neq j$; $l_b, l_e = 1, \ldots, |s^i|, l_b \leq l_e$; $m_b, m_e = 1, \ldots, |s^j|, m_b \leq m_e$, we let

$$\mathcal{E}^{ij}(l_b \leftrightarrow l_e, m_b \leftrightarrow m_e)$$

denote the collection of all sets of edges in $S \subseteq E^{ij}$ such that

(a) all edges in S are pairwise incompatible;
(b) for each edge $\{v^i_l, v^j_m\} \in S$, $l_b \leq l \leq l_e$ and $m_b \leq m \leq m_e$;
(c) S is maximal with respect to properties (a) and (b).

The next lemma shows formally that each set $S \in \mathcal{E}^{ij}(l_b \leftrightarrow l_e, m_b \leftrightarrow m_e)$ is obtained by the procedure whose informal intuitive description is the following. Imagine node sets V^i and V^j laid down on two horizontal lines with V^i above V^j and nodes from left to right in increasing order of indices (see Figure 3). Start by inserting into S the current edge $\{v^i_{l_b}, v^j_{m_e}\}$ and then let each subsequent edge inserted into S be obtained from the current one by either moving its upper endpoint one node to the right or its lower endpoint one node to the left, ending with edge $\{v^i_{l_e}, v^j_{m_b}\}$.

Lemma 1. *Given a set $S \subseteq E^{ij}$ and l_b, l_e, m_b, m_e such that $1 \leq l_b \leq l_e \leq |s^i|$, $1 \leq m_b \leq m_e \leq |s^j|$, we have that $S \in \mathcal{E}^{ij}(l_b \leftrightarrow l_e, m_b \leftrightarrow m_e)$ if and only if S has the form*

$$S = \{\{v^i_{l_1}, v^j_{m_1}\}, \ldots, \{v^i_{l_p}, v^j_{m_p}\}\},$$

where

(i) $l_1 = l_b, l_p = l_e, m_1 = m_e, m_p = m_b$;
(ii) for $q = 1, \ldots, p-1$, either $l_{q+1} = l_q + 1$ and $m_{q+1} = m_q$ or $l_{q+1} = l_q$ and $m_{q+1} = m_q - 1$.

Proof. To prove sufficiency, given a set S satisfying (i) and (ii), it is immediate to verify that S satisfies (a) and (b). As to (c), assume that there exists an edge $e = \{v^i_l, v^j_m\} \notin S$ such that also $S \cup \{e\}$ satisfies (a) and (b). Let q be the minimum index $q \in \{1, \dots, p\}$ such that $l = l_q$, noting that such an index exists due to (ii). If $m_q < m$, (ii) implies that edge $\{v^i_{l-1}, v^j_{m_q}\}$, which is compatible with e, is in S. Otherwise, $m_q > m$ and (ii) implies that edge $\{v^i_{l+1}, v^j_{m_q}\}$, which is compatible with e, is in S. In both cases, $S \cup \{e\}$ does not satisfy (a), yielding a contradiction.

To prove necessity, consider a set $S \in \mathcal{E}^{ij}(l_b \leftrightarrow l_e, m_b \leftrightarrow m_e)$. By requirement (a), all edges in S are incompatible, meaning that, if we order them by increasing node of V^i to which they are incident (with appropriate tie breaking), they will also be ordered by decreasing node of V^j to which they are incident. Therefore, we assume that the edges in S according to this order are $\{v^i_{l_1}, v^j_{m_1}\}, \dots, \{v^i_{l_p}, v^j_{m_p}\}$. Requirement (b) imposes $l_1 \geq l_b, l_p \leq l_e, m_1 \leq m_e, m_p \geq m_b$. We next show that the maximality requirement (c) implies that all these inequalities hold at equality. Indeed, if $l_1 > l_b$, edge $\{v^i_{l_b}, v^j_{m_e}\}$ could be added to S, preserving (a) and (b). This implies $l_1 = l_b$, and an analogous reasoning implies $l_p = l_e, m_1 = m_e, m_p = m_b$, showing (i). Finally, we show that the maximality requirement (c) implies (ii). Indeed, for $q = 1, \dots, p-1$, (ii) is not satisfied if and only if $l_{q+1} \geq l_q + 1$ and $m_{q+1} \leq m_q - 1$, and in this case edge $\{v^i_{l_{q+1}}, v^j_{m_q}\}$ could be added to S. This implies (ii). □

The following characterization of the collection $\mathcal{I}$ of maximal sets of pairwise incompatible gap arcs in A^{ij} shows that the number of constraints (4) is $O(nk)$.

Lemma 2. *Given a set $I \subseteq A^{ij}$, we have that $I \in \mathcal{I}$ if and only if I has the form $A^{ij}(m+1 \leftrightarrow m)$, where $1 \leq m < |s^i|$.*

Proof. Recalling the interval graph $H^{ij}(A^{ij})$ defined above, we have that $I \in \mathcal{I}$ if and only if I corresponds to a maximal clique of $H^{ij}(A^{ij})$. It is well known that each maximal clique in an interval graph corresponds to a set of intervals containing a value $m+1$ that is the endpoint of some interval. For graph $H^{ij}(A^{ij})$, this set is given by $A^{ij}(m+1 \leftrightarrow m)$, proving necessity. Moreover, for each integer $m \in \{1, \dots, |s^i| - 1\}$, the corresponding clique $A^{ij}(m+1 \leftrightarrow m)$ in $H^{ij}(A^{ij})$ is maximal since every gap arc $a \notin A^{ij}(m+1 \leftrightarrow m)$ is compatible with either $(v^i_m, v^i_m)^j$ or $(v^i_{m+1}, v^i_{m+1})^j$, proving sufficiency. □

Maximal clique inequalities In the following, a *clique* will denote a set $K = K_E \cup K_A$ of pairwise incompatible edges and gap arcs with $K_E \subseteq E^{ij}$ and $K_A \subseteq A^{ij}$ for some $1 \leq i, j \leq k, i \neq j$. These are precisely the cliques of the incompatibility graph H^{ij}. A clique is *maximal* if the corresponding set K is maximal, i.e., if there is no edge in E^{ij} nor arc in A^{ij} that is incompatible with all edges and arc in K. The corresponding *maximal clique inequality* has the form

$$\sum_{e \in K_E} x_e + \sum_{a \in K_A} y_a \leq 1, \tag{8}$$

and is clearly valid for $\mathcal{P}$. Note that (4) and (8) (with "=" is replaced by "≤") are examples of maximal clique inequalities.

We already noted that H^{ij} is not perfect. Nevertheless, the following characterization of its maximal cliques, which is an extension of the one in [24] (where only alignment variables are considered and hence H^{ij} is perfect), will lead to an efficient algorithm for the separation of (8).

Proposition 2. *An inequality of the form (8) associated with sets $K_E \subseteq E^{ij}$ and $K_A \subseteq A^{ij}$ is a maximal clique inequality if and only if either*

$$K_E = \emptyset, K_A = A^{ij}(m+1 \leftrightarrow m)$$

for some $1 \le l < |s^i|$, or

$$K_E \in \mathcal{E}^{ij}(l_b \leftrightarrow l_e, 1 \leftrightarrow |s^j|), K_A = A^{ij}(l_b \leftrightarrow l_e)$$

for some $1 \le l_b \le l_e \le |s^i|$.

Proof. To prove sufficiency, noting that each set $K = K_E \cup K_A$ with one of the forms given in the statement is clearly a clique of H^{ij}, we have to show that it is maximal. If $K_E = \emptyset$ and $K_A = A^{ij}(m+1 \leftrightarrow m)$ for some $1 \le l < |s^i|$, the clique is maximal by Lemma 2 and the fact that there is no edge that is incompatible with both gap arcs $(v^i_m, v^i_m)^j, (v^i_{m+1}, v^i_{m+1})^j \in K_A$. If $K_E \in \mathcal{E}^{ij}(l_b \leftrightarrow l_e, 1 \leftrightarrow |s^j|)$ and $K_A = A^{ij}(l_b \leftrightarrow l_e)$ for some $1 \le l_b \le l_e \le |s^i|$, the clique is maximal by Lemma 1 and the fact that there is no gap arc outside K_A that is incompatible with both edges $\{v^i_{l_b}, v^j_1\}, \{v^i_{l_e}, v^j_{|s^j|}\} \in K_E$.

To prove necessity, we have to show that each maximal clique $K = K_E \cup K_A$ of H^{ij} has one of the forms given in the statement. To this end, if $K_E = \emptyset$, K is a maximal set of incompatible gap arcs in A^{ij}, and the proof follows from Lemma 2, the resulting maximal clique inequality being one of (4). Otherwise, let $v^i_{l_b}$ and $v^i_{l_e}$ be the first and last node of s^i to which some edge in K_E is adjacent. The unique maximal set of gap arcs in A^{ij} that are pairwise incompatible with all edges in K_E is $A^{ij}(l_b \leftrightarrow l_e)$. Moreover, maximality of K_E, jointly with Lemma 1 and the fact that there is no gap arc in A^{ji} in the clique, imply that $K_E \in \mathcal{E}^{ij}(l_b \leftrightarrow l_e, m_b \leftrightarrow m_e)$ with $m_b = 1$ and $m_e = |s^j|$. In particular, if $m_b > 1$ (resp., $m_e < |s^j|$) then edge $\{v^i_{l_e}, v^j_1\}$ (resp., $\{v^i_{l_b}, v^j_{|s^j|}\}$) would be incompatible with all members of K. □

An illustration of possible sets K_E and K_A in a maximal clique inequality is given in Figure 5.

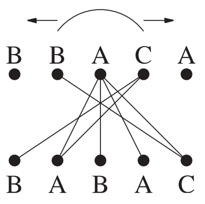

Fig. 5. Illustration of possible edges and gap arcs in a maximal clique inequality

Lifted mixed cycle inequalities In this section, we introduce a class of inequalities that dominate (3). As is the case for (3), these inequalities involve only alignment variables. Consider a sequence of strings $s^{i_1}, \dots, s^{i_t}$, $t \geq 3$, along with edge set $C \subseteq E$, which is partitioned into edge sets $C^{i_r i_{r+1}} \subseteq E^{i_r i_{r+1}}$, $r = 1, \dots, t$ (letting $i_{t+1} := i_1$). If C meets the following requirements

(a) for $r = 1, \dots, t$, all edges in $C^{i_r i_{r+1}}$ are pairwise incompatible;
(b) every set $\{e_1, \dots, e_t\}$, where e_r is chosen arbitrarily from $C^{i_r i_{r+1}}$ for $r = 1, \dots, t$, is the set of edges in a critical mixed cycle;

then the inequality

$$\sum_{e \in C} x_e \leq t - 1 \tag{9}$$

is valid for $\mathcal{P}$. Note that (3) are a special case of (9) in which each set $C^{i_r i_{r+1}}$ contains only one edge. If in addition

(c) C is *maximal* with respect to properties (a) and (b);

we call (9) a *lifted mixed cycle inequality.*

Proposition 3. *An inequality of the form* (9) *associated with C, where $C = \bigcup_{r=1}^{t} C^{i_r i_{r+1}}$ with $C^{i_r i_{r+1}} \subseteq E^{i_r i_{r+1}}$, $r = 1, \dots, t$, is a lifted mixed cycle inequality if and only if there exists a path*

$$P = \{v_{l_1}^{i_1}, v_{l_2}^{i_2}\}, \{v_{l_2}^{i_2}, v_{l_3}^{i_3}\}, \dots, \{v_{l_t}^{i_t}, v_{l_1 - 1}^{i_1}\}$$

containing t edges (and no arc) such that

(i) for $r = 1, \dots, t$, $\{v_{l_r}^{i_r}, v_{l_{r+1}}^{i_{r+1}}\} \in C^{i_r i_{r+1}}$;
(ii) for $r = 1, \dots, t$, $C^{i_r i_{r+1}} \in \mathcal{E}^{i_r i_{r+1}}(l_r \leftrightarrow |s^{i_r}|, 1 \leftrightarrow l_{r+1})$;

where $i_{t+1} := i_1$ and $l_{t+1} := l_1 - 1$.

Proof. To prove sufficiency, consider a set C with the form given in the statement. By definition of $\mathcal{E}^{i_r i_{r+1}}(\cdot)$, C satisfies (a). Moreover, observe that path P, together with arc $(v_{l_1 - 1}^{i_1}, v_{l_1}^{i_1}) \in A_p$, defines a mixed cycle M. We have that C satisfies (b), as any set of edges obtained by selecting one edge from each $C^{i_r i_{r+1}}$ defines a mixed cycle, which is obtained from M by replacing two consecutive edges by two edges with a path in A_p in between. Finally, suppose C does not satisfy (c), and consider an edge $e = \{v_l^{i_r}, v_m^{i_{r+1}}\} \in E^{i_r i_{r+1}} \setminus C$ such that $C \cup \{e\}$ also satisfies (a) and (b). By (ii) and Lemma 1, we must have $l < l_r$ or $m > l_{r+1}$. In this case, (b) is violated, since $P \setminus \{v_{l_r}^{i_r}, v_{l_{r+1}}^{i_{r+1}}\} \cup \{e\}$ is not the set of edges in a critical mixed cycle. This is a contradiction.

To prove necessity, we show that every set C that satisfies (a), (b) and (c) has the form given in the statement. For $r = 1, \dots, t$, order the edges in $C^{i_r i_{r+1}}$ according to increasing character of s^{i_r} to which they are incident, breaking ties by decreasing character of $s^{i_{r+1}}$ to which they are incident, and let e_r be the first edge according to this ordering. Consider the critical mixed cycle M with edges $e_1, \dots, e_t$ (which is a critical mixed cycle by (b)) and note that, after possible shifting of the indices $i_1, \dots, i_k$, we

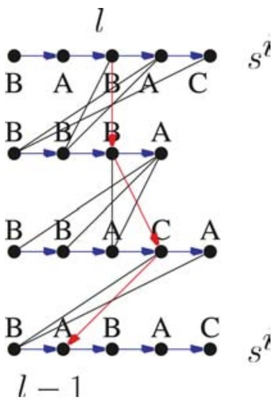

Fig. 6. Illustration of possible edges in a lifted mixed cycle inequality (edges represented by an arrow are those in the reference mixed cycle)

can assume without loss of generality that M contains at least one arc in A_p associated with s^{i_1}. We show that M contains only one arc. Indeed, if M contains also an arc in A_p associated with string s^{i_r} for some $1 < r \leq t$, we have a contradiction to the maximality of C, as we may add to $C^{i_{r-1}i_r}$ the edge in $E^{i_{r-1}i_r}$ with the same endpoint in $s^{i_{r-1}}$ as e_{r-1} and the same endpoint in s^{i_r} as e_r. An analogous reasoning shows that C contains exactly one arc in A_p associated with string s^{i_1}. Therefore, P exists and has the form $e_1, \ldots, e_t$. Moreover, (i) is verified by definition, and (ii) follows from (a) and (c). □

Figure 6 shows a possible set of edges in a lifted mixed cycle inequality. Note that P together with arc $(v^{i_1}_{l_1-1}, v^{i_1}_{l_1}) \in A_p$ defines a mixed cycle, which we will call the *reference mixed cycle* in the sequel.

Generalized transitivity inequalities The last class of inequalities that we present is a generalization of the transitivity inequalities (5). Consider three different strings $s^{i_1}, s^{i_2}, s^{i_3}$, along with a node $v^{i_1}_{l_1} \in V^{i_1}$ and subsets $S^2 \subseteq V^{i_2}$ and $S^3 \subseteq V^{i_3}$. Constraints (5) imply that, if we realize an edge with one endpoint in $v^{i_1}_{l_1}$ and the other in S^2, and an edge with one endpoint in $v^{i_1}_{l_1}$ and the other in S^3, we must also realize an edge with one endpoint in S^2 and the other in S^3. This yields the valid inequality

$$\sum_{v^{i_2}_{l_2} \in S^2} x_{\{v^{i_1}_{l_1}, v^{i_2}_{l_2}\}} + \sum_{v^{i_3}_{l_3} \in S^3} x_{\{v^{i_1}_{l_1}, v^{i_3}_{l_3}\}} - \sum_{v^{i_2}_{l_2} \in S^2} \sum_{v^{i_3}_{l_3} \in S^3} x_{\{v^{i_2}_{l_2}, v^{i_3}_{l_3}\}} \leq 1, \tag{10}$$

which coincides with one of (5) if $|S^2| = |S^3| = 1$.

2.4. *Polyhedral results*

In this section, we show that the inequalities defined in the previous section are (under appropriate technical conditions) facet-defining for the gapped trace polytope $\mathcal{P}$.

Preliminaries Our proofs will be based on a general result that is easy to derive but also non-standard, and therefore stated and proved in detail below. The result concerns a polyhedron $\mathcal{Q} \subseteq \mathbb{R}^p$ for which there exists a set of m valid equations $Ax = b$ that are linearly independent, i.e., the dimension of $\mathcal{Q}$ is at most $p - m$. By possible renumbering of the coordinates, we assume without loss of generality that the first m columns of A are linearly independent. An inequality $\alpha x \leq \beta$ or equation $\alpha x = \beta$ is said to be in *normal form* (with respect to $Ax = b$) if $\alpha_j = 0$ for $j = 1, \ldots, m$. As customary, we say that an inequality $\alpha x \leq \beta$ valid for $\mathcal{Q}$ defines a proper face $\mathcal{F} = \{x \in \mathcal{Q} : \alpha x = \beta\}$ if $\emptyset \neq \mathcal{F} \neq \mathcal{Q}$.

Proposition 4. *Consider an inequality $\alpha x \leq \beta$ valid for $\mathcal{Q}$ in normal form defining a proper face $\mathcal{F}$. If for every inequality $\gamma x \leq \delta$ valid for $\mathcal{Q}$ in normal form defining a proper face $\mathcal{G} \supseteq \mathcal{F}$ we have $(\gamma, \delta) = \lambda(\alpha, \beta)$ for some scalar $\lambda > 0$, then (i) the dimension of $\mathcal{Q}$ is exactly $n - m$ and (ii) $\mathcal{F}$ is a facet of $\mathcal{Q}$.*

Proof. In order to prove (i), suppose that the dimension of $\mathcal{Q}$ is smaller than $p - m$, i.e., there exists an equation $cx = d$ that is valid for $\mathcal{Q}$ and not implied by $Ax = b$. By redefining $(c, d) := (c, d) - (c_1, \ldots, c_m)B^{-1}(A, b)$, where B is the square matrix defined by the first m columns of A, we have that $cx = d$ is in normal form, and that $(c, d) \neq (0, 0)$ since the original $cx = d$ is not a linear combination of the equations in $Ax = b$.

The proof of (i) follows from the fact that the inequality $(\alpha + c)x \leq (\beta + d)$ violates the hypothesis in the statement of the proposition. Indeed, this inequality is in normal form, valid for $\mathcal{Q}$, and defines the proper face $\mathcal{G} = \mathcal{F}$. On the other hand, $(\alpha + c, \beta + d) = \lambda(\alpha, \beta)$ for some scalar $\lambda > 0$ is impossible, since this would imply either $(c, d) = (0, 0)$ or $(\alpha, \beta) = \frac{1}{\lambda - 1}(c, d)$, the latter implying in turn $\mathcal{F} = \mathcal{Q}$, i.e., $\mathcal{F}$ is not proper.

In order to show (ii), suppose $\mathcal{F}$ is not a facet of $\mathcal{Q}$, i.e., there exists a valid inequality for $\mathcal{Q} \gamma x \leq \delta$ that defines a proper face $\mathcal{G} \supseteq \mathcal{F}$ such that $\mathcal{G} \setminus \mathcal{F} \neq \emptyset$. By redefining $(\gamma, \delta) := (\gamma, \delta) - (\gamma_1, \ldots, \gamma_m)B^{-1}(A, b)$, we have that $\gamma x \leq \delta$ is in normal form, valid for $\mathcal{Q}$, and defines the same proper face $\mathcal{G}$, which implies that $(\gamma, \delta) = \lambda(\alpha, \beta)$ for some scalar $\lambda > 0$ is impossible, violating the hypothesis in the statement of the proposition. $\square$

Implicit equations and normal form We first determine a lower bound on the dimension of $\mathcal{P}$, by finding equations that are implicit in the formulation.

Lemma 3. *The equations*

$$y_{(v_1^j, v_{|s^j|}^j)^i} = y_{(v_1^i, v_{|s^i|}^i)^j}, \quad i, j = 1, \ldots, k; i < j, \tag{11}$$

are valid for $\mathcal{P}$.

Proof. Due to constraints (2) and (4), gap arc $(v_1^i, v_{|s^i|}^i)^j$ is realized if and only if no edge in E^{ij} is realized. In this case, also gap arc $(v_1^j, v_{|s^j|}^j)^i$ is realized. $\square$

Recall that n is the overall number of characters in the k strings.

Lemma 4. *The dimension of $\mathcal{P}$ is at most $|E| + |A_g| - (k-1)n - \binom{k}{2}$.*

Proof. Note that $\mathcal{P}$ lies in the $(|E|+|A_g|)$-dimensional space. We have $(k-1)n$ equations (2) in the initial formulation as well as $\binom{k}{2}$ equations (11). Furthermore, these equations are linearly independent. Indeed, each variable $y_{(v_l^j, v_l^j)^i}$ $(i, j = 1, \ldots, k; i \neq j; l = 1, \ldots, |s^j|)$ has a nonzero coefficient (equal to 1) in exactly one equation (2), whereas each variable $y_{(v_1^j, v_{|s^j|}^j)^i}$ $(i, j = 1, \ldots, k; i < j)$ has a nonzero coefficient (equal to 1) in exactly one equation (2). Hence, the equation system (2), (11), after permuting rows so as to have these $(k-1)n + \binom{k}{2}$ variables at the beginning, looks like

$$\begin{bmatrix} I & A & B \\ 0 & I & C \end{bmatrix} \cdot \begin{bmatrix} y \\ x \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \tag{12}$$

where I denotes an identity matrix of suitable order, and clearly has full row rank. Therefore, the dimension of the polytope is at most the dimension of the underlying space minus the rank of this equation system, which completes the proof. $\square$

Jointly with Proposition 4, our proofs in the following will show that equations (2) and (11) are the only implicit equations (and so the lower bound on the dimension of $\mathcal{P}$ is tight).

In accordance with Proposition 4, we call an inequality $\alpha x + \beta y \leq \gamma$ in *normal form* if

$$\beta_{(v_l^j, v_l^j)^i} = 0, \quad i, j = 1, \ldots, k; i \neq j; l = 1, \ldots, |s^j| \tag{13}$$

and

$$\beta_{(v_1^j, v_{|s^j|}^j)^i} = 0, \quad i, j = 1, \ldots, k; i < j. \tag{14}$$

Note that the lifted mixed cycle inequalities (9) and generalized transitivity inequalities (10) are in normal form, as all arc variables have coefficient 0 in these inequalities. Moreover, a maximal clique inequality (8) is in normal form if $K_E \in \mathcal{E}^{ij}(l_b \leftrightarrow l_e, 1 \leftrightarrow |s^j|)$ with $l_e > l_b$ and $i < j$. Indeed, since only some arcs in A^{ij} correspond to variables with nonzero coefficient in the inequality, condition $i < j$ ensures (14), and condition $l_e > l_b$ ensures (13). Note that condition $i < j$ is satisfied by each clique inequality (8) subject to a possible renumbering of the k strings, and that condition $l_e > l_b$ is satisfied by each clique inequality that is not one of (4) nor the inequality version of (2).

Following Proposition 4, our approach to prove that a given valid inequality in normal form $\alpha x + \beta y \leq \gamma$ is facet-defining is the following. Consider an arbitrary valid inequality in normal form $\pi x + \sigma y \leq \rho$ that is *tight* (i.e., satisfied at equality) for all points for which $\alpha x + \beta y \leq \gamma$ is tight. Proving that $\pi x + \sigma y \leq \rho$ is a *multiple* of $\alpha x + \beta y \leq \gamma$ implies that the latter is facet-defining.

Throughout the section, given a valid inequality in normal form $\alpha x + \beta y \leq \gamma$, a *tight alignment* is an alignment corresponding to a feasible solution of (1–6) for which the inequality is tight. The crucial issue within our proofs is often to define tight alignments so that they are feasible but do not transitively imply any "undesired" additional edge.

We will describe alignments by listing the edges that they realize (possibly mentioning that some additional edges are transitively implied by those explicitly given). Note that the arcs realized by an alignment are uniquely determined by the edges realized. On the other hand, in some cases we will point out explicitly that some arcs are realized if this is helpful within the description.

Three technical lemmas The following three key lemmas, whose proof is fairly technical, form the basis of our proofs. The structure of their proofs is essentially the same; however, we prefer to give them separately since the details are quite different for each of them.

Within the proofs, we will often work with two alignments A_1 and A_2, pointing out all their differences explicitly and saying that they have no further difference, without giving the (straightforward) details showing that, apart from those mentioned explicitly, the arcs implied by the edges in the two alignments are the same.

Lemma 5. *Consider a maximal clique inequality (8) and a corresponding valid inequality in normal form $\pi x + \sigma y \leq \rho$ that is tight for all points for which the former is tight. If $K_E \in \mathcal{E}^{ij}(l_b \leftrightarrow l_e, 1 \leftrightarrow |s^j|)$, with $l_b > 1$, $l_e < |s^i|$, $l_e \geq l_b + 2$, $\{v^i_{l_b}, v^j_{|s^j|-2}\} \in K_E$, and $\{v^i_{l_e}, v^j_3\} \in K_E$, then, for each gap arc $a \in A_g \setminus K_A$, we have $\sigma_a = 0$.*

Proof. Consider a maximal clique inequality (8) that satisfies the technical conditions in the statement of the lemma, assuming the strings are numbered so that it is in normal form, i.e., $i < j$, where i and j are the indices of the two strings associated with the inequality and will be used as such throughout the proof. Moreover, let $\pi x + \sigma y \leq \rho$ be in normal form and tight for all points for which the former is tight.

Claim. Consider two edges $e = \{v^i_l, v^j_m\}$, $f = \{v^i_l, v^j_{m+1}\} \in K_E$ such that $1 < l < |s^i|$ and $1 \leq m < |s^j|$. Then, $\pi_e = \pi_f$.

Proof of Claim 2.4. We show a tight alignment A_1 that realizes e and gap arc $(v^j_{m+1}, v^j_{m+1})^i \notin K_A$ and a tight alignment A_2 that realizes f and gap arc $(v^j_m, v^j_m)^i \notin K_A$. These are the only differences between A_1 and A_2, which realize also edge $\{v^i_{l-1}, v^j_{m-1}\} \notin K_E$ if $m > 1$; edge $\{v^i_{l+1}, v^j_{m+2}\} \notin K_E$ if $m + 1 < |s^j|$; and no other edge. The fact that $\pi x + \sigma y \leq \rho$ is tight for these alignments implies

$$\pi_e + \sigma_{(v^j_{m+1}, v^j_{m+1})^i} = \pi_f + \sigma_{(v^j_m, v^j_m)^i},$$

yielding the claim since $\sigma_{(v^j_{m+1}, v^j_{m+1})^i} = \sigma_{(v^j_m, v^j_m)^i} = 0$ by definition of normal form. □

Claim. Consider two edges $e = \{v^{i'}_l, v^{j'}_m\}$, $f = \{v^{i'}_l, v^{j'}_{m+1}\} \in E \setminus K_E$ such that $1 \leq i', j' \leq k$, $i' \neq j'$, $1 < l < |s^{i'}|$, and $1 \leq m < |s^{j'}|$. Then, $\pi_e = \pi_f$.

Proof of Claim 2.4. We show a tight alignment A_1 that realizes e and gap arc $(v^{j'}_{m+1}, v^{j'}_{m+1})^{i'}$ and a tight alignment A_2 that realizes f and gap arc $(v^{j'}_m, v^{j'}_m)^{i'}$. These are the only differences between A_1 and A_2, which realize also edge $\{v^{i'}_{l-1}, v^{j'}_{m-1}\}$ if $m > 1$; edge $\{v^{i'}_{l+1}, v^{j'}_{m+2}\}$ if $m + 1 < |s^{j'}|$; and the possible other edges specified below.

The proof is subdivided into cases, and, for each case, reasoning as in Claim 2.4 yields the proof. The specification of the additional edges realized by A_1, A_2 (so as to guarantee that they are tight) will be provided case by case in the case analysis below. We have to ensure that these common edges do not lead, together with e and f, to different transitively implied edges in A_1 and A_2, i.e., that no other edge with one endpoint in common with e or f is realized by A_1, A_2.
(1) If $|\{i, j\} \cap \{i', j'\}| \leq 1$, A_1, A_2 realize also an edge in K_E that has no common endpoint with e, f, noting that such an edge exists as $l_e \geq l_b + 2$.
(2) If $i = i'$ and $j = j'$, we consider subcases.
(2.1) If $l - 1 > l_e$, A_1, A_2 realize also $\{v^i_{l_e}, v^j_1\} \in K_E$.
(2.2) If $l + 1 < l_b$, A_1, A_2 realize also $\{v^i_{l_b}, v^j_{|s^j|}\} \in K_E$.
(2.3) If $l - 1 = l_e$, it is easy to verify that either $\{v^i_{l-1}, v^j_{m-1}\} \in K_E$, in which case A_1, A_2 do not realize any further edge, or there exists $g = \{v^i_{l'}, v^j_{m'}\} \in K_E$ with $l' < l - 1$ and $m' < m - 1$, in which case A_1, A_2 realize also g.
(2.4) If $l + 1 = l_b$, the case is symmetrical to (2.3).
(2.5) If $l_b \leq l \leq l_e$, since $\{v^i_l, v^j_m\}, \{v^i_l, v^j_{m+1}\} \notin K_E$, there exists $\{v^i_l, v^j_{m'}\} \in K_E$ with either $m' < m$ or $m' > m + 1$. If $m' < m$, then $l_b \leq l - 1 \leq l_e$, and, as in (2.3), either $\{v^i_{l-1}, v^j_{m-1}\} \in K_E$ or there exists $g = \{v^i_{l'}, v^j_{m'}\} \in K_E$ with $l' < l - 1$ and $m' < m - 1$. The case $m' > m + 1$ is symmetrical.
(3) If $i = j'$ and $j = i'$, we consider subcases.
(3.1) If $m - 1 > l_e$, A_1, A_2 realize also $\{v^i_{l_e}, v^j_1\} \in K_E$.
(3.2) If $m + 2 < l_b$, A_1, A_2 realize also $\{v^i_{l_b}, v^j_{|s^j|}\} \in K_E$.
(3.3) If $m - 1 = l_e$, either $\{v^i_{m-1}, v^j_{l-1}\} \in K_E$, in which case A_1, A_2 do not realize any further edge, or there exists $g = \{v^i_{m'}, v^j_{l'}\} \in K_E$ with $m' < m - 1$ and $l' < l - 1$, in which case A_1, A_2 realize also g.
(3.4)] If $m + 2 = l_b$, the case is symmetrical to (3.3).
(3.5)] If $l_b \leq m \leq l_e$, there exists an edge $\{v^i_m, v^j_{l'}\} \in K_E$ with either $l' < l$ or $l' > l$. If $l' < l$, then $l_b \leq m - 1 \leq l_e$, and, as in (3.3), either $\{v^i_{m-1}, v^j_{l-1}\} \in K_E$ or there exists $g = \{v^i_{m'}, v^j_{l'}\} \in K_E$ with $m' < m - 1$ and $l' < l - 1$. If $l' > l$, it is easy to verify that $l_b \leq m + 2 \leq l_e$, as otherwise $m = l_e$ or $m + 1 = l_e$, and either $\{v^i_m, v^j_l\} \in K_E$ or $\{v^i_{m+1}, v^j_l\} \in K_E$. Then, the situation is symmetrical to the case $l_b \leq m - 1 \leq l_e$.
(3.6) If $l_b \leq m + 1 \leq l_e$, the case is symmetrical to (3.5). □

Claim. Consider two gap arcs $a = (v^{j'}_l, v^{j'}_{m-1})^{i'}$, $b = (v^{j'}_l, v^{j'}_m)^{i'} \in A_g \setminus K_A$ such that $1 \leq i', j' \leq k$, $i' \neq j'$, and $m < |s^{j'}|$. Then, $\sigma_a = \sigma_b$.

Proof of Claim 2.4. We show a tight alignment A_1 that realizes edge $e = \{v^{i'}_p, v^{j'}_m\}$ and gap arcs a and $c = (v^{j'}_{m+1}, v^{j'}_{m+1})^{i'}$, and a tight alignment A_2 that realizes edge $f = \{v^{i'}_p, v^{j'}_{m+1}\}$ and gap arc b, with $1 < p < |s^{i'}|$ and either $e, f \in K_E$ or $e, f \notin K_E$, which implies $\pi_e = \pi_f$ by Claims 2.4 and 2.4. These are the only differences between A_1 and A_2, which realize also edge $\{v^{i'}_{p-1}, v^{j'}_{l-1}\}$ if $l > 1$; edge $\{v^{i'}_{p+1}, v^{j'}_{m+2}\}$ if $m+1 < |s^j|$; and the possible other edges specified below. The specification of p and the additional edges realized by the two alignments (so as to guarantee that they are tight) is provided below for different cases, as for Claim 2.4. The fact that $\pi x + \sigma y \leq \rho$ is tight for these alignments implies $\pi_e + \sigma_a + \sigma_c = \pi_f + \sigma_b$, i.e., $\sigma_a = \sigma_b$ since $\sigma_c = 0$ by definition of normal form.

(1) If $|\{i, j\} \cap \{i', j'\}| \leq 1$, p is arbitrary and A_1, A_2 realize also an edge in K_E that has no common endpoint with e, f, noting that such an edge exists as $l_e \geq l_b + 2$.

(2) If $i = i'$ and $j = j'$, since $l_b > 1$ and $l_e < |s^i|$, let p be such that $l_b \leq p \leq l_e$ and both e and f are in K_E, and A_1, A_2 do not realize any further edge.

(3) If $i = j'$ and $j = i'$, note that $\{l_b, \dots, l_e\} \nsubseteq \{l, \dots, m\}$, as this would imply $b \in K_A$. Hence, either $l_b < l$ or $l_e > m$. We consider subcases.

(3.1) If $l - 1 > l_e$, let $p > 2$ and A_1, A_2 realize also $\{v^i_{l_e}, v^j_1\} \in K_E$.

(3.2) If $m + 2 < l_b$, let $p < |s^j| - 1$ and A_1, A_2 realize also $\{v^i_{l_b}, v^j_{|s^j|}\} \in K_E$.

(3.3) If $l - 1 = l_e$, let $p = 2$ and A_1, A_2 do not realize any further edge.

(3.4) If $m + 2 = l_b$, the case is symmetrical to (3.3).

(3.5) If $l_b < l \leq l_e$, since $\{v^i_{l_b}, v^j_{|s^j|-2}\} \in K_E$, we have that $g = \{v^i_{l-1}, v^j_q\} \in K_E$ for some $q \leq |s^{i'}| - 2$. Let $p = q + 1 < |s^j|$ and A_1, A_2 realize also g.

(3.6) If $l_b \leq m + 1 < l_e$, the case is symmetrical to (3.5), using the fact that $\{v^i_{l_e}, v^j_3\} \in K_E$.

(3.7) If $l \leq l_b$ and $m+1 \geq l_e$, we have $m+1 = l_e$, otherwise $\{l_b, \dots, l_e\} \subseteq \{l, \dots, m\}$. In this case, noting $m > l_b$, let p be such that $e, f \in K_E$ and A_1, A_2 do not realize any further edge. □

Claim. Consider two gap arcs $a = (v^{j'}_{l+1}, v^{j'}_m)^{i'}, b = (v^{j'}_l, v^{j'}_m)^{i'} \in A_g \setminus K_A$ such that $1 \leq i', j' \leq k, i' \neq j'$, and $l > 1$. Then, $\sigma_a = \sigma_b$.

Proof of Claim 2.4. Analogous to the proof of Claim 2.4. □

We are now ready to prove the statement of the lemma. Consider a gap arc $a = (v^{j'}_l, v^{j'}_m)^{i'} \notin K_A$. For the case $m < |s^{j'}|$, note that $\sigma_{(v^{j'}_l, v^{j'}_l)^{i'}} = 0$ and $(v^{j'}_l, v^{j'}_p)^{i'} \notin K_A$ for $p = l, \dots, m$, together with Claim 2.4, imply $\sigma_{(v^{j'}_l, v^{j'}_p)^{i'}} = 0$ for $p = l, \dots, m$, i.e., $\sigma_a = 0$. The case $l > 1$ is analogous to the case $m < |s^{j'}|$, using Claim 2.4. The case remaining is $a = (v^{j'}_1, v^{j'}_{|s^{j'}|})^{i'}$. In this case, if $i' < j'$, we have $\sigma_a = 0$ by definition of normal form. Finally, if $i' > j'$, let $b = (v^{i'}_1, v^{i'}_{|s^{i'}|})^{j'}$, noting $\sigma_b = 0$. Note that $a \notin K_A$ implies $j' \neq i$ or $i' \neq j$. Consider the following tight alignments that realize a: A_1, which realizes no edge; A_2, which realizes edge $e = \{v^{i'}_1, v^{j'}_1\}$; A_3, which realizes edge $f = \{v^{i'}_{|s^{i'}|}, v^{j'}_{|s^{j'}|}\}$; and A_4, which realizes edges e, f. All these alignments are tight since they realize gap arc $(v^i_1, v^i_{|s^i|})^j \in K_A$. Recalling

$$\begin{aligned}\sigma_b &= \sigma_{(v^{i'}_2, v^{i'}_{|s^{i'}|})^{j'}} = \sigma_{(v^{j'}_2, v^{j'}_{|s^{j'}|})^{i'}} = \sigma_{(v^{i'}_1, v^{i'}_{|s^{i'}|-1})^{j'}} \\ &= \sigma_{(v^{j'}_1, v^{j'}_{|s^{j'}|-1})^{i'}} = \sigma_{(v^{i'}_2, v^{i'}_{|s^{i'}|-1})^{j'}} = \sigma_{(v^{j'}_2, v^{j'}_{|s^{j'}|-1})^{i'}} = 0,\end{aligned}$$

we have $\sigma_a = \pi_e = \pi_f = \pi_e + \pi_f$, whose unique solution is $\sigma_a = \pi_e = \pi_f = 0$. □

Lemma 6. *Consider a lifted mixed cycle inequality* (9) *and a corresponding valid inequality in normal form $\pi x + \sigma y \leq \rho$ that is tight for all points for which the former is tight. If $C = \bigcup_{r=1}^{t} C^{i_r i_{r+1}}$ and $C^{i_r i_{r+1}} \in \mathcal{E}^{i_r i_{r+1}}(l_r \leftrightarrow |s^{i_r}|, 1 \leftrightarrow l_{r+1})$ for $r = 1, \ldots, t$, with $3 < l_r < |s^{i_r}| - 2$ for $r = 2, \ldots, t$, and $4 < l_1 < |s^{i_1}| - 2$, then, for each gap arc $a \in A_g$, we have $\sigma_a = 0$.*

Proof. Consider a lifted mixed cycle inequality (9) that satisfies the technical conditions in the statement of the lemma. Recall the definitions of e_r and f_r in the proof of Theorem 2, and let $E_0 \subseteq E$ denote the set of edges having coefficient 0 in the inequality. Moreover, let $\pi x + \sigma y \leq \rho$ be in normal form and tight for all points for which the former is tight.

Claim. Consider two edges $e = \{v_l^i, v_m^j\}$, $f = \{v_l^i, v_{m+1}^j\} \in E_0$ such that $1 \leq i, j \leq k$, $i \neq j$, $1 < l < |s^i|$, and one of the following holds:

(a) $|\{i, j\} \cap \{i_1, \ldots, i_t\}| \leq 1$;
(b) $i = i_r$, $j = i_q$ for some indices $r, q \in \{1, \ldots, t\}$ that are not cyclically consecutive; $l - 1 > l_r$; $m > 1$;
(c) $i = i_r$, $j = i_{r+1}$ for some index $r \in \{1, \ldots, t\}$; $l + 1 < l_r$ if $r \neq 1$, $l + 1 < l_r - 1$ if $r = 1$;
(d) $i = i_{r+1}$, $j = i_r$ for some index $r \in \{1, \ldots, t\}$; $l - 1 > l_{r+1}$; $m > 1$.

Then, $\pi_e = \pi_f$.

Proof of Claim 2.4. We show a tight alignment A_1 that realizes e and gap arc $(v_{m+1}^j, v_{m+1}^j)^i$ and a tight alignment A_2 that realizes f and gap arc $(v_m^j, v_m^j)^i$. These are the only differences between A_1 and A_2, which realize also edge $\{v_{l-1}^i, v_{m-1}^j\}$ if $m > 1$; edge $\{v_{l+1}^i, v_{m+2}^j\}$ if $m + 1 < |s^j|$; and the other edges specified below. The proof is subdivided into cases, and, for each case, reasoning as in Claim 2.4 in the proof of Lemma 5 yields the proof.
(1) If $j \notin \{i_1, \ldots, i_t\}$, recalling (a), A_1, A_2 realize also edges $\{e_1, \ldots, e_t\} \setminus \{e_r\}$ for an arbitrary $r \in \{1, \ldots, t\}$.
(2) If $j = i_r$ and $i \notin \{i_1, \ldots, i_t\}$, recalling (a), we consider subcases.
(2.1) If $m > 1$, A_1, A_2 realize also edges $\{e_1, \ldots, e_t\} \setminus \{e_r\}$.
(2.2) If $m = 1$, A_1, A_2 realize also edges $\{e_1, \ldots, e_t\} \setminus \{e_{r-1}\}$.
(3) If $i = i_r$ and $j = i_q$ with r, q not cyclically consecutive, recalling (b), A_1, A_2 realize also edges $\{e_1, \ldots, e_t\} \setminus \{e_r, e_q\} \cup \{f_r\}$.
(4) If $i = i_r$ and $j = i_{r+1}$, recalling (c), we consider subcases.
(4.1) If $m + 1 < |s^{i_{r+1}}|$, A_1, A_2 realize also edges $\{e_1, \ldots, e_t\} \setminus \{e_{r-1}, e_r\} \cup \{f_{r-1}\}$.
(4.2) If $m + 1 = |s^{i_{r+1}}|$, noting that $m > l_{r+1}$, A_1, A_2 realize also edges $\{e_1, \ldots, e_t\} \setminus \{e_{r-1}, e_r, e_{r+1}\} \cup \{f_{r-1}, f_{r+1}\}$.
(5)] If $i = i_{r+1}$ and $j = i_r$, recalling (d), A_1, A_2 realize also edges $\{e_1, \ldots, e_t\} \setminus \{e_r, e_{r+1}\} \cup \{f_{r+1}\}$. □

Claim. Consider two gap arcs $a = (v_l^j, v_{m-1}^j)^i$, $b = (v_l^j, v_m^j)^i$ such that $1 \leq i, j \leq k$, $i \neq j$, and $m < |s^j|$. Then, $\sigma_a = \sigma_b$.

Proof of Claim 2.4. We show a tight alignment A_1 that realizes edge $e = \{v^i_p, v^j_m\}$ and gap arcs a and $c = (v^j_{m+1}, v^j_{m+1})^i$, and a tight alignment A_2 that realizes edge $f = \{v^i_p, v^j_{m+1}\}$ and gap arc b, with e, f satisfying one of the technical requirements in the statement of Claim 2.4, which implies $\pi_e = \pi_f$. These are the only differences between A_1 and A_2, which realize also edge $\{v^i_{p-1}, v^j_{l-1}\}$ if $l > 1$; edge $\{v^i_{p+1}, v^j_{m+2}\}$ if $m + 1 < |s^j|$; and the other edges specified below. The proof is subdivided into cases, and, for each case, reasoning as in Claim 2.4 in the proof of Lemma 5 yields the proof.
(1) If $j \notin \{i_1, \ldots, i_t\}$, p is arbitrary and A_1, A_2 realize also edges $\{e_1, \ldots, e_t\} \setminus \{e_r\}$ for an arbitrary $r \in \{1, \ldots, t\}$.
(2) If $j = i_r$ and $i \notin \{i_1, \ldots, i_t\}$, noting that $m > 1$, p is arbitrary and A_1, A_2 realize also edges $\{e_1, \ldots, e_t\} \setminus \{e_r\}$.
(3) If $i = i_r$ and $j = i_q$ with r, q not cyclically consecutive, noting that $m > 1$, let p be such that $p - 1 > l_r$ and A_1, A_2 realize also edges $\{e_1, \ldots, e_t\} \setminus \{e_r, e_q\} \cup \{f_r\}$.
(4) If $i = i_r$ and $j = i_{r+1}$, we consider subcases.
(4.1) If $m + 1 < |s^{i_{r+1}}|$, let p be such that $p + 1 < l_r$ if $r \neq 1$ and $p + 1 < l_r - 1$ if $r = 1$, and A_1, A_2 realize also edges $\{e_1, \ldots, e_t\} \setminus \{e_{r-1}, e_r\} \cup \{f_{r-1}\}$.
(4.2) If $m + 1 = |s^{i_{r+1}}|$, noting that $m > l_{r+1}$, let p be such that $p + 1 < l_r$ if $r \neq 1$ and $p + 1 < l_r - 1$ if $r = 1$, and A_1, A_2 realize also edges $\{e_1, \ldots, e_t\} \setminus \{e_{r-1}, e_r, e_{r+1}\} \cup \{f_{r-1}, f_{r+1}\}$.
(5)] If $i = i_{r+1}$ and $j = i_r$, noting that $m > 1$, let p be such that $p - 1 > l_{r+1}$ and A_1, A_2 realize also edges $\{e_1, \ldots, e_t\} \setminus \{e_r, e_{r+1}\} \cup \{f_{r+1}\}$. □

Claim. Consider two gap arcs $a = (v^j_{l+1}, v^j_m)^i$, $b = (v^j_l, v^j_m)^i$ such that $1 \leq i, j \leq k$, $i \neq j$, and $l > 1$. Then, $\sigma_a = \sigma_b$.

Proof of Claim 2.4. Analogous to the proof of Claim 2.4. □

The proof of the lemma follows from Claims 2.4 and 2.4 by the same reasoning as in the conclusion of the proof of Lemma 5. In particular, in order to show $\sigma_a = 0$ for $a = (v^j_1, v^j_{|s^j|})^i$ and $i > j$, consider the following tight alignments that realize a: A_1, which realizes edges $\{e_1, \ldots, e_t\} \setminus \{e_r\}$, where e_r is defined so that no edge realized has one endpoint in string s^i and the other in s^j; A_2, which realizes $\{e_1, \ldots, e_t\} \setminus \{e_r\} \cup \{e\}$; A_3, which realizes $\{e_1, \ldots, e_t\} \setminus \{e_r\} \cup \{f\}$; and A_4, which realizes $\{e_1, \ldots, e_t\} \setminus \{e_r\} \cup \{e, f\}$, where e and f have both one endpoint in string s^i and the other in s^j, are compatible, and have no common endpoint with $\{e_1, \ldots, e_t\}$. As in the proof of Lemma 5, we have $\sigma_a = \pi_e = \pi_f = \pi_e + \pi_f$, whose unique solution is $\sigma_a = \pi_e = \pi_f = 0$. □

Lemma 7. *Consider a generalized transitivity inequality (10) and a corresponding valid inequality in normal form $\pi x + \sigma y \leq \rho$ that is tight for all points for which the former is tight. If there exist three consecutive nodes $v^{i_2}_{l-1}, v^{i_2}_l, v^{i_2}_{l+1} \in V^{i_2} \setminus S^2$ and three consecutive nodes $v^{i_3}_{m-1}, v^{i_3}_m, v^{i_3}_{m+1} \in V^{i_3} \setminus S^3$, then, for each gap arc $a \in A_g$, we have $\sigma_a = 0$.*

Proof. Consider a generalized transitivity inequality (10) that satisfies the technical conditions in the statement of the lemma, letting $E_1 \subseteq E$ and $E_0 \subseteq E$ denote the set of edges having coefficient 1 and 0 in the inequality, respectively, and $\pi x + \sigma y \leq \rho$ be in normal form and tight for all points for which the former is tight.

Claim. Consider two edges $e = \{v_l^i, v_m^j\}$, $f = \{v_l^i, v_{m+1}^j\} \in E_0$ such that $1 \le i, j \le k$, $i \ne j$, $1 < l < |s^i|$, $\{v_{l-1}^i, v_{m-1}^j\} \in E_0$ if $m > 1$, and $\{v_{l+1}^i, v_{m+2}^j\} \in E_0$ if $m+1 < |s^j|$. Then, $\pi_e = \pi_f$.

Proof of Claim 2.4. We show a tight alignment A_1 that realizes e and gap arc $(v_{m+1}^j, v_{m+1}^j)^i$ and a tight alignment A_2 that realizes f and gap arc $(v_m^j, v_m^j)^i$. Alignments A_1 and A_2 realize also edge $\{v_{l-1}^i, v_{m-1}^j\}$ if $m > 1$; edge $\{v_{l+1}^i, v_{m+2}^j\}$ if $m + 1 < |s^j|$; and the other edges specified below. The proof is subdivided into cases, and, for each case, reasoning as in Claim 2.4 in the proof of Lemma 5 yields the proof.
(1) If $i_1 \notin \{i, j\}$, A_1, A_2 realize an edge $g \in E_1$ not incident with v_l^i, v_m^j, v_{m+1}^j. Such an edge always exists. Indeed, if it did not exist, we would have $\{i, j\} = \{i_2, i_3\}$, say $i = i_2$ and $j = i_3$, $S_2 = \{v_l^i\}$ and $S_3 \subseteq \{v_m^j, v_{m+1}^j\}$, i.e., e or f would have coefficient -1 in (10), in contradiction to $e, f \in E_0$.
(2) If $i_1 = i$, we consider subcases.
(2.1) If $l_1 \ne l$, A_1, A_2 realize an edge $g \in E_1$ not incident with any node in V^j.
(2.2) If $l_1 = l$, A_1, A_2 realize an edge $\{v_{l_1}^{i_1}, v_h^{j'}\} \in E_1$ such that $j' \ne j$, as well as the transitively implied edges. Besides e, f, the only difference between A_1 and A_2 is that the first realizes $e' = \{v_m^j, v_h^{j'}\}$, and the second $f' = \{v_{m+1}^j, v_h^{j'}\}$. We get

$$\pi_e + \pi_{e'} + \sigma_{(v_{m+1}^j, v_{m+1}^j)^i} + \sigma_{(v_{m+1}^j, v_{m+1}^j)^{j'}} = \pi_f + \pi_{f'} + \sigma_{(v_m^j, v_m^j)^i} + \sigma_{(v_m^j, v_m^j)^{j'}},$$

yielding $\pi_e = \pi_f$ since $\pi_{e'} = \pi_{f'}$ from (1), noting that $e, f \in E_0$ implies $e', f' \in E_0$.
(3) If $i_1 = j$, we consider subcases.
(3.1) If $l_1 \ne m$, $l_1 \ne m + 1$, A_1, A_2 realize any edge in E_1 not incident with any node in V^i.
(3.2) If $l_1 = m$, A_1, A_2 realize edges $\{v_m^j, v_h^{i'}\} \in E_1$ and $\{v_{m+1}^j, v_{h+1}^{i'}\}$, such that $i' \ne i$ and $h < |s^{i'}|$, as well as the transitively implied edges. Besides e, f the only difference between A_1 and A_2 is that the first realizes also $e' = \{v_l^i, v_h^{i'}\}$ and the second also $f' = \{v_l^i, v_{h+1}^{i'}\}$. We get

$$\pi_e + \pi_{e'} + \sigma_{(v_{m+1}^j, v_{m+1}^j)^i} + \sigma_{(v_{h+1}^{i'}, v_{h+1}^{i'})^i} = \pi_f + \pi_{f'} + \sigma_{(v_m^j, v_m^j)^i} + \sigma_{(v_h^{i'}, v_h^{i'})^i},$$

yielding $\pi_e = \pi_f$ since $\pi_{e'} = \pi_{f'}$ from (1).
(3.3) If $l_1 = m + 1$, the case is symmetrical to (3.2). □

Claim. Consider two gap arcs $a = (v_l^j, v_{m-1}^j)^i$, $b = (v_l^j, v_m^j)^i$ such that $1 \le i, j \le k$, $i \ne j$, and $m < |s^j|$. Then, $\sigma_a = \sigma_b$.

Proof of Claim 2.4. We show a tight alignment A_1 that realizes edge $e = \{v_p^i, v_m^j\}$ and gap arcs a and $c = (v_{m+1}^j, v_{m+1}^j)^i$, and a tight alignment A_2 that realizes edge $f = \{v_p^i, v_{m+1}^j\}$ and gap arc b, with $1 < p < |s^i|$ and $e, f \in E_0$, which implies $\pi_e = \pi_f$ by Claim 2.4. Alignments A_1 and A_2 realize also $\{v_{p-1}^i, v_{l-1}^j\} \in E_0$ if $l > 1$; $\{v_{p+1}^i, v_{m+2}^j\} \in E_0$ if $m + 1 < |s^j|$; and the other edges specified below. The proof is subdivided into cases, and, for each case, reasoning as in Claim 2.4 in the proof of

Lemma 5 yields the proof.
(1) If $i_1 \notin \{i, j\}$, let p be such that $v^i_{p-1}, v^i_p, v^i_{p+1} \notin S_2 \cup S_3$ and A_1, A_2 realize an edge $g \in E_1$ not incident with any of v^i_p, v^j_m, v^j_{m+1}, noting that such a g exists.
(2) If $i_1 = i$, let $p \neq l_1$ and A_1, A_2 realize an edge $g \in E_1$ not incident with any node in V^j.
(3) If $i_1 = j$, let p be such that $v^i_{p-1}, v^i_p, v^i_{p+1} \notin S_2 \cup S_3$. We consider subcases.
(3.1) If $l_1 \neq m, l_1 \neq m+1$, A_1, A_1 realize an edge in E_1 not incident with any node in V^i.
(3.2) If $l_1 = m$, A_1, A_2 realize edges $\{v^j_m, v^{i'}_h\} \in E_1$ and $\{v^j_{m+1}, v^{i'}_{h+1}\}$, such that $i' \neq i$ and $h < |s^{i'}|$, as well as the transitively implied edges. Besides e, f the only difference between A_1 and A_2 is that the first realizes also $e' = \{v^i_p, v^{i'}_h\}$ and the second also $f' = \{v^i_p, v^{i'}_{h+1}\}$. We get

$$\pi_e + \pi_{e'} + \sigma_{(v^j_{m+1}, v^j_{m+1})^i} + \sigma_a + \sigma_{(v^{i'}_{h+1}, v^{i'}_{h+1})^i} = \pi_f + \pi_{f'} + +\sigma_b + \sigma_{(v^{i'}_h, v^{i'}_h)^i},$$

yielding $\sigma_a = \sigma_b$ since $\pi_e = \pi_f$ and $\pi_{e'} = \pi_{f'}$ by Claim 2.4.
(3.3) If $l_1 = m+1$, the case is symmetrical to (3.2). □

Claim. Consider two gap arcs $a = (v^j_{l+1}, v^j_m)^i$, $b = (v^j_l, v^j_m)^i$ such that $1 \leq i, j \leq k$, $i \neq j$, and $l > 1$. Then, $\sigma_a = \sigma_b$.

Proof of Claim 2.4. Analogous to the proof of Claim 2.4. □

The proof of the lemma follows from Claims 2.4 and 2.4 by the same reasoning as in the conclusion of the proof of Lemma 5. In particular, in order to show $\sigma_a = 0$ for $a = (v^j_1, v^j_{|s^j|})^i$ and $i > j$, consider the following tight alignments that realize a: A_1, which realizes only an edge $e = \{v^{i_1}_{l_1}, v^{j'}_{m'}\} \in E_1$ such that $\{i_1, j'\} \neq \{i, j\}$; A_2, which realizes e and $f = \{v^i_l, v^j_m\}$; A_3, which realizes e and $g = \{v^i_h, v^j_q\}$; and A_4, which realizes e, f, g, where f and g are compatible and have no common endpoint with e. As in the proof of Lemma 5, we have $\sigma_a + \pi_e = \pi_e + \pi_f = \pi_e + \pi_g = \pi_e + \pi_f + \pi_g$, whose unique solution is $\sigma_a = \pi_e = \pi_f = 0$. □

Note that the technical conditions in the statements above are fairly mild, namely they are satisfied by the vast majority of the inequalities. Moreover, they are generally not necessary, but on the other hand lead to a considerable simplification of the proofs with respect to the case in which weaker conditions are imposed.

The facet proofs

Theorem 1. *Every maximal clique inequality* (8) *that satisfies the technical conditions in the statement of Lemma 5 is facet-defining for $\mathcal{P}$.*

Proof. Consider a maximal clique inequality (8) that satisfies the technical conditions in the statement of Lemma 5, recalling that $K_E \in \mathcal{E}^{ij}(l_b \leftrightarrow l_e, 1 \leftrightarrow |s^j|)$, and let $\pi x + \sigma y \leq \rho$ be in normal form and tight for all points for which the former is tight. By Lemma 5, we have $\sigma_a = 0$ for all $a \in A_g \setminus K_A$.

We have $\pi_e = \rho$ for every $e \in K_E$, since the alignment in which e is the only edge realized is tight, and for each gap arc a realized we have $\sigma_a = 0$.

Moreover, for each $e \in E \setminus K_E$ that is not of the form $\{v_l^i, v_{|s^j|}^j\}$ with $l < l_b$ or $\{v_l^i, v_1^j\}$ with $l > l_e$, we next show an edge $f \in K_E$ for which it is easy to verify that the alignment in which e, f are the only edges realized is tight (and for each gap arc a realized we have $\sigma_a = 0$). This implies $\pi_e + \pi_f = \rho$, i.e., $\pi_e = 0$ since $\pi_f = \rho$. Specifically, if $e \notin E^{ij}$, let f be any edge in K_E with no common endpoint with e. Otherwise, let $e = \{v_l^i, v_m^j\}$. If $l_b \le l \le l_e$ let f be an edge in K_E compatible with e, noting that such an edge exists by the maximality of $\mathcal{E}^{ij}(l_b \leftrightarrow l_e, 1 \leftrightarrow |s^j|)$; if $l < l_b$ let $f := \{v_{l_b}^i, v_{|s^j|}^j\}$; if $l > l_e$ let $f := \{v_{l_e}^i, v_1^j\}$.

For each gap arc $a = (v_l^i, v_m^i)^j \in K_A$, we have $l \le l_b$ and $m \ge l_e$, and the tight alignment that realizes a, edge $e = \{v_{l-1}^i, v_1^j\}$ if $l > 1$, edge $f = \{v_{m+1}^i, v_{|s^j|}^j\}$ if $m < |s^i|$, and no other edge implies $\sigma_a + \pi_e + \pi_f = \rho$, i.e., $\sigma_a = \rho$ since $\pi_e = \pi_f = 0$ as shown above.

Finally, for each edge $e = \{v_l^i, v_{|s^j|}^j\}$ with $l < l_b$, the tight alignment that realizes e and no other edge realizes also gap arc $a = (v_{l+1}^i, v_{|s^i|}^i)^j \in K_A$, implying $\pi_e + \sigma_a = \rho$, i.e., $\pi_e = 0$ since $\sigma_a = \rho$, as shown above. An analogous reasoning shows $\pi_e = 0$ for $e = \{v_l^i, v_1^j\}$ with $l > l_e$.

Summarizing, $\pi x + \sigma y \le \rho$ has the form

$$\sum_{e \in K_E} \rho\, x_e + \sum_{a \in K_A} \rho\, y_a \le \rho.$$

Noting that ρ has to be positive for the inequality to be valid and distinct from $0 \le 0$ concludes the proof. □

Theorem 2. *Every lifted mixed cycle inequality* (9) *that satisfies the conditions in the statement of Lemma 6 is facet-defining for $\mathcal{P}$.*

Proof. Consider a lifted mixed cycle inequality (9) that satisfies the conditions in the statement of Lemma 6, recalling that $C = \bigcup_{r=1}^{t} C^{i_r i_{r+1}}$ and $C^{i_r i_{r+1}} \in \mathcal{E}^{i_r i_{r+1}}(l_r \leftrightarrow |s^{i_r}|, 1 \leftrightarrow l_{r+1})$ for $r = 1, \dots, t$, and let $\pi x + \sigma y \le \rho$ be in normal form and tight for all points for which the former is tight. By Lemma 6, we have $\sigma_a = 0$ for all $a \in A_g$.

For $r = 1, \dots, t$, define $e_r := \{v_{|s^{i_r}|}^{i_r}, v_1^{i_{r+1}}\}$ and note that $e_r \in C^{i_r i_{r+1}}$. Furthermore, let $f_1, \dots, f_t$, with $f_r := \{v_{l_r}^{i_r}, v_{l_{r+1}}^{i_{r+1}}\}$ for $r = 1, \dots, t$ be the edges in the reference mixed cycle associated with the mixed cycle inequality considered.

We first show $\pi_e = \frac{\rho}{t-1}$ for $e \in C^{i_r i_{r+1}}$ and $r = 1, \dots, t$. For $r = 1, \dots, t$, consider the tight alignment that realizes all edges in $\{e_1, \dots, e_t\} \setminus \{e_r\}$ and no other edge. (In particular, no edge is transitively implied since $|s^i| \ge 4$ for all i.) These t alignments show that $\pi_{e_1}, \dots, \pi_{e_t}$ satisfy the following system of equations

$$\begin{array}{ccccccc} & \pi_{e_2} & + \dots & +\pi_{e_{t-1}} & +\pi_{e_t} & = \rho \\ \pi_{e_1} & & + \dots & +\pi_{e_{t-1}} & +\pi_{e_t} & = \rho \\ & & \ddots & & & & , \\ \pi_{e_1} & +\pi_{e_2} & + \dots & & +\pi_{e_t} & = \rho \\ \pi_{e_1} & +\pi_{e_2} & + \dots & +\pi_{e_{t-1}} & & = \rho \end{array}$$

i.e., $\pi_{e_r} = \frac{\rho}{t-1}$ for $r = 1, \dots, t$. Now consider an edge $e = \{v_l^{i_r}, v_m^{i_{r+1}}\} \in C^{i_r i_{r+1}}$, recalling that $l \geq l_r > 1$. Considering the tight alignment that realizes edges e and $\{e_1, \dots, e_t\} \setminus \{e_r, e_{r+1}\}$ (and no other edge) we get $\pi_e = \frac{\rho}{t-1}$.

We complete the proof by showing $\pi_e = 0$ for every edge $e = \{v_l^i, v_m^j\} \notin C$. The proof is subdivided into cases. In the definition of tight alignments, one has to be careful not to define (infeasible) alignments that realize the set of edges in a mixed cycle.

(1) If $\{i, j\} \cap \{i_1, \dots, i_t\} = \emptyset$, $\pi_e = 0$ is shown by the tight alignment that realizes edges e and $\{e_1, \dots, e_t\} \setminus \{e_r\}$ for an arbitrary $r \in \{1, \dots, t\}$.

(2) If $i = i_r$ and $j \notin \{i_1, \dots, i_t\}$, $\pi_e = 0$ is shown by the tight alignment that realizes edges e and $\{e_1, \dots, e_t\} \setminus \{e_r\}$ if $l > 1$, and e and $\{e_1, \dots, e_t\} \setminus \{e_{r-1}\}$ if $l = 1$.

(3) If $i = i_r$, $j = i_q$ and i_r, i_q are not (cyclically) consecutive in $i_1, \dots, i_t$, we consider subcases. By possibly exchanging i and j, we can assume without loss of generality $r < q$. For the sake of exposition, we give the proof for $r \neq 1$ and then discuss how to adapt it if this is not the case.

(3.1) If $l < |s^{i_r}|$ and $m < l_q$, $\pi_e = 0$ is shown by the tight alignment that realizes edges e and $\{e_1, \dots, e_t\} \setminus \{e_{r-1}, e_{q-1}\} \cup \{f_{q-1}\}$.

(3.2) If $l < l_r$ and $m < |s^{i_q}|$, the case is analogous to (3.1).

(3.3) If $l > 1$ and $m > l_q$, $\pi_e = 0$ is shown by the tight alignment that realizes edges e and $\{e_1, \dots, e_t\} \setminus \{e_r, e_q\} \cup \{f_q\}$.

(3.4) If $l > l_r$ and $m > 1$, the case is analogous to (3.3).

(3.5) If $m = l_q$, we have $1 < m < |s^{i_q}|$, for which cases $l < l_r$ and $l > l_r$ have already been considered. Therefore, the only case left is $l = l_r$. Recalling that $1 \notin \{r+1, \dots, q-1\}$, the proof is by induction on $q - r$. If $q - r = 2$, $\pi_e = 0$ is shown by the tight alignment that realizes e and $\{e_1, \dots, e_t\} \setminus \{e_r, e_{r+1}, e_q\} \cup \{f_r, f_{r+1}\}$, noting that e is transitively implied by $f_r = \{v_l^{i_r}, v_{l_{r+1}}^{i_{r+1}}\}$ and $f_{r+1} = \{v_{l_{r+1}}^{i_{r+1}}, v_m^{i_q}\}$. This shows the basis of the induction. If $q - r > 2$, $\pi_e = 0$ is shown by the alignment that realizes e and $\{e_1, \dots, e_t\} \setminus \{e_r, \dots, e_q\} \cup \{f_r, \dots, f_{q-1}\}$ and the transitively implied edges, all of the form $f = \{v_{l_{r'}}^{i_{r'}}, v_{l_{q'}}^{i_{q'}}\}$ with $2 \leq q' - r' < q - r$, so that $\pi_f = 0$ by the inductive hypothesis.

(3.6) If $l = l_r$, the case has already been considered in (3.5).

(3.7) If $l = |s^{i_r}|$, we have $l > l_r$, for which case $m > 1$ has already been considered in (3.4). Therefore, the case left is $m = 1$. In this case, considering an edge $f = \{v_{l'}^{i_{q-1}}, v_{m'}^{i_q}\}$ with $f \in C^{i_{q-1} i_q}$, $l' > l_{q-1}$, $1 < m' \leq l_q$ (noting that such an edge always exists), $\pi_e = 0$ is shown by the alignment that realizes e, f and $\{e_1, \dots, e_t\} \setminus \{e_r, \dots, e_q\} \cup \{f_r, \dots, f_{q-2}\}$, along with the edges g transitively implied by $f_r, \dots, f_{q-2}$, for which $\pi_g = 0$ is shown by (3.5).

(3.8) If $m = |s^{i_q}|$, the case is analogous to (3.7).

For case (3), if $r = 1$, the analysis is identical by replacing $l < l_r$ by $l < l_r - 1$ in (3.2). This leads to case $l = l_r - 1$ still to be considered in (3.5). This case is handled analogously considering the interval from q to $t + 1 \equiv 1$ (instead of the interval from r to q) as well as induction on $(t + 1) - q$.
(4) If $i = i_r$ and $j = i_{r+1}$, we consider subcases. As before, assume first $r \neq 1$.
(4.1) If $l \geq l_r$ and $m \leq l_{r+1}$, there exists at least one edge $f \in C^{i_r i_{r+1}}$ that is compatible with e, otherwise $C^{i_r i_{r+1}}$ would not be maximal. Then, $\pi_e = 0$ is shown by the tight alignment that realizes edges $e, f, \{f_1, \dots, f_t\} \setminus \{f_r, f_{r+1}\}$, and the transitively implied edges g, with $\pi_g = 0$ by (3).
(4.2) If $l < l_r$, $\pi_e = 0$ is shown by the tight alignment that realizes e, $\{f_1, \dots, f_t\} \setminus \{f_r, f_{r+1}\} \cup \{e_{r+1}\}$, and, in case $m = |s^{i_{r+1}}|$, the edge g transitively implied by e and e_{r+1}, with $\pi_g = 0$ by (3).
(4.3) If $m > l_{r+1}$, the case is analogous to (4.2).

For case (4), if $r = 1$, we have to replace $l < l_r$ by $l < l_r - 1$ in (4.2) and consider the additional case $l = l_r - 1$ and $m \leq l_{r+1}$. In this case, if $m = l_{r+1}$, $\pi_e = 0$ is shown by the tight alignment that realizes e, $\{f_1, \dots, f_t\} \setminus \{f_r\}$, and the transitively implied edges g, with $\pi_g = 0$ by (3). If $m < l_{r+1}$, $\pi_e = 0$ is shown by the tight alignment that realizes e, $\{f_1, \dots, f_t\} \setminus \{f_{r+1}\}$, and the transitively implied edges g, with $\pi_g = 0$ by (3). □

Theorem 3. *Every generalized transitivity inequality (10) that satisfies the technical conditions in the statement of Lemma 7 is facet-defining for $\mathcal{P}$.*

Proof. Consider a generalized transitivity inequality (10) that satisfies the technical conditions in the statement of Lemma 7, and let $\pi x + \sigma y \leq \rho$ be in normal form and tight for all points for which the former is tight. By Lemma 7, we have $\sigma_a = 0$ for all $a \in A_g$.

For each $e = \{v_{l_1}^{i_1}, v_{l_2}^{i_2}\}$, with $v_{l_2}^{i_2} \in S^2$, or $e = \{v_{l_1}^{i_1}, v_{l_3}^{i_3}\}$, with $v_{l_3}^{i_3} \in S^3$, we have $\pi_e = \rho$ since the alignment in which e is the only edge realized is tight.

Analogously, for each $e = \{v_{l_2}^{i_2}, v_{l_3}^{i_3}\}$, with $v_{l_2}^{i_2} \in S^2$ and $v_{l_3}^{i_3} \in S^3$, we have $\pi_e = -\rho$ since the alignment in which the only edges realized are e, $f = \{v_{l_1}^{i_1}, v_{l_2}^{i_2}\}$, and $g = \{v_{l_1}^{i_1}, v_{l_3}^{i_3}\}$ is tight, which implies $\pi_e + \pi_f + \pi_g = \rho$, i.e., $\pi_e = -\rho$ since $\pi_f = \pi_g = \rho$.

Finally, consider an edge e with coefficient 0 in the generalized transitivity inequality considered. If e is not incident with $v_{l_1}^{i_1}$, then there exists a tight alignment that realizes only edges e and either $\{v_{l_1}^{i_1}, v_{l_2}^{i_2}\}$ with $v_{l_2}^{i_2} \in S^2$, or $\{v_{l_1}^{i_1}, v_{l_3}^{i_3}\}$ with $v_{l_3}^{i_3} \in S^3$. This implies $\pi_e = 0$. On the other hand, if e is incident with $v_{l_1}^{i_1}$, say $e = \{v_{l_1}^{i_1}, v_l^j\}$, with $j \neq i_2$, then there exists a tight alignment that realizes edges e, $f = \{v_{l_1}^{i_1}, v_{l_2}^{i_2}\}$ for some $v_{l_2}^{i_2} \in S^2$, and $g = \{v_l^j, v_{l_2}^{i_2}\}$. We have $\pi_e + \pi_f + \pi_g = \rho$, i.e., $\pi_e = 0$ since $\pi_f = \rho$ and $\pi_g = 0$, as shown above. (If $j = i_2$, the same reasoning applies letting $f = \{v_{l_1}^{i_1}, v_{l_3}^{i_3}\}$ and $g = \{v_l^j, v_{l_3}^{i_3}\}$ for some $v_{l_3}^{i_3} \in S^3$.)

Summarizing, $\pi x + \sigma y \leq \rho$ has the form

$$\sum_{v_{l_2}^{i_2} \in S^2} \rho\, x_{\{v_{l_1}^{i_1}, v_{l_2}^{i_2}\}} + \sum_{v_{l_3}^{i_3} \in S^3} \rho\, x_{\{v_{l_1}^{i_1}, v_{l_3}^{i_3}\}} - \sum_{v_{l_2}^{i_2} \in S^2} \sum_{v_{l_3}^{i_3} \in S^3} \rho\, x_{\{v_{l_2}^{i_2}, v_{l_3}^{i_3}\}} \leq \rho.$$

Noting that ρ has to be positive for the inequality to be valid and distinct from $0 \leq 0$ concludes the proof. $\square$

3. A branch-and-cut approach

In this section, we describe our branch-and-cut approach to the problem, mainly showing that the three classes of valid inequalities presented in the previous section are efficiently separable. At the end, we also describe our overall branch-and-cut implementation.

3.1. Separation procedures

In this section, we present efficient algorithms that solve the separation problem for the classes of inequalities mentioned. The fact that polynomial-time separation algorithms exist for the facet-defining inequalities that we present is somehow unusual (in the positive sense), in that often efficient separation procedures are known only for *weaker* versions of the facet-defining (i.e., strongest possible) inequalities that one can derive for the problem at hand.

We will not consider separation methods for the constraints in our initial ILP model, since there are only $O(nk)$ equations of type (2) and inequalities of type (4), transitivity inequalities (5) are a special case of generalized transitivity inequalities, and mixed cycle inequalities (3) are dominated by lifted mixed cycle inequalities.

Pairgraphs Given edge weights w_e for all edges $e \in E$, a fundamental step in our separation algorithms is the computation of $K_E \in \mathcal{E}^{ij}(l_b \leftrightarrow l_e, m_b \leftrightarrow m_e)$ that maximizes $\sum_{e \in K_E} w_e$. In accordance with Lemma 1, one may use dynamic programming to determine such a set. As is natural, we represent the dynamic programming procedure as a path computation in a directed acyclic graph. Specifically, we need the notion of *pairgraph*, introduced by [24] for the special case $l_b = 1, l_e = |s^i|, m_b = 1$, and $m_e = |s^j|$, which is easily extended to our case. The pairgraph P^{ij} for an ordered pair of strings s^i, s^j is a directed acyclic graph with a node n_e for every edge $e \in E^{ij}$ of the alignment graph. Every node $n_{\{v_l^i, v_m^j\}}$ has up to two outgoing arcs, namely $(n_{\{v_l^i, v_m^j\}}, n_{\{v_{l+1}^i, v_m^j\}})$ if $l < |s^i|$, and $(n_{\{v_l^i, v_m^j\}}, n_{\{v_l^i, v_{m-1}^j\}})$ if $m > 1$. See Figure 7 for an illustration.

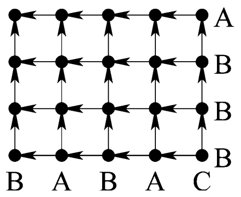

Fig. 7. Example of a pairgraph

Lemma 8. *There is a one-to-one correspondence between sets* $\{e_1, \ldots, e_l\} \in \mathcal{E}^{ij}(l_b \leftrightarrow l_e, m_b \leftrightarrow m_e)$ *and paths* $n_{e_1}, \ldots, n_{e_l}$ *in* P^{ij} *such that* $e_1 = \{v^i_{l_b}, v^j_{m_e}\}$ *and* $e_l = \{v^i_{l_e}, v^j_{m_b}\}$.

Proof. Follows directly from Lemma 1 and the definition of pairgraph. □

In accordance with Lemma 8, the set $K_E \in \mathcal{E}^{ij}(l_b \leftrightarrow l_e, m_b \leftrightarrow m_e)$ that maximizes $\sum_{e \in K_E} w_e$ corresponds to the longest node-weighed path in P^{ij} between $n_{\{v^i_{l_b}, v^j_{m_e}\}}$ and $n_{\{v^i_{l_e}, v^j_{m_b}\}}$ with respect to node weights $w_{n_e} := w_e$ for $e \in E$. Since P^{ij} is acyclic, we can compute such a path in time linear in the size of P^{ij}, i.e., in $O(|s^i||s^j|)$ time.

Maximal clique inequalities

Theorem 4. *Given a point* $(x^*, y^*) \in \mathbb{R}_+^{|E|+|A_g|}$*, determining whether there is a maximal clique inequality* (8) *violated by* (x^*, y^*) *is solvable in* $O(n^3)$ *time.*

Proof. We fix the strings s^i and s^j, $i \neq j$, considering all $k(k-1)$ pairs. In accordance with Proposition 2, for every $1 \leq l_b < l_e \leq |s^i|$, we (a) compute $K_E \in \mathcal{E}^{ij}(l_b \leftrightarrow l_e, 1 \leftrightarrow |s^j|)$ that maximizes $\sum_{e \in K_E} x^*_e$ by finding a longest path in the pairgraph from $n_{\{v^i_{l_b}, v^j_{|s^j|}\}}$ to $n_{\{v^i_{l_e}, v^j_1\}}$; (b) compute $\sum_{a \in A^{ij}(l_b \leftrightarrow l_e)} y^*_a$; and (c) test if the corresponding maximal clique inequality is violated, that is if $\sum_{e \in K_E} x^*_e + \sum_{a \in A^{ij}(l_b \leftrightarrow l_e)} y^*_a > 1$.

Given i and j, executing Step (a) for all l_b, l_e pairs takes $O(|s^i|^2|s^j|)$ time, since for each of the $|s^i| - 1$ values of l_b we find the longest path tree from $n_{\{v^i_{l_b}, v^j_{|s^j|}\}}$ in P^{ij}, in time $O(|s^i||s^j|)$. As shown below, executing Step (b) for all l_b, l_e pairs requires $O(|s^i|^2)$ time. The running time for Step (c) is constant for each l_b, l_e pair. Thus the total running time for all i, j pairs is $O(\sum_{i=1}^k \sum_{j=1}^k |s^i|^2|s^j| + \sum_{i=1}^k |s^i|^2) = O(n^3)$.

In order to execute Step (b) in $O(|s^i|^2)$ time for all l_b, l_e pairs, note that there are $|s^i|^2$ gap variables associated with substrings of s^i of length at most $|s^i|$. This running time can be achieved by first computing, for $q = 1, \ldots, |s^i|$, $\pi_q := \sum_{l=1}^q y^*_{(v^i_l, v^i_q)}$ and $\omega_q := \sum_{m=q}^{|s^i|} y^*_{(v^i_q, v^i_m)}$ in $O(|s^i|^2)$ time, and then computing $\sigma_{l_b, l_e} := \sum_{a \in A^{ij}(l_b \leftrightarrow l_e)} y^*_a$ in the following order: $\sigma_{1,1} := \omega_1$; for $q = 2, \ldots, |s^i|$, $\sigma_{1,q} := \sigma_{1,q-1} + \omega_q$; for $p = 2, \ldots, |s^i|$ and $q = p, \ldots, |s^i|$, $\sigma_{p,q} := \sigma_{p-1,q} - \pi_{p-1}$.

The method is trivially extended to the case $l_b = l_e$ (for which maximal clique inequalities are a relaxation of (2)), and $K_E = \emptyset$ (for which they coincide with (4)). □

Note that if all strings have approximately the same length n/k the actual complexity is proportional to n^3/k.

Lifted mixed cycle inequalities

Theorem 5. *Given a point* $(x^*, y^*) \in \mathbb{R}_+^{|E|+|A_g|}$ *that satisfies all maximal clique inequalities* (8)*, determining whether there is a lifted mixed cycle inequality violated by* (x^*, y^*) *is solvable in* $O(n^3)$ *time.*

Proof. By Proposition 3, if $\{v_{l_r}^{i_r}, v_{l_{r+1}}^{i_{r+1}}\}$ is an edge of the reference mixed cycle corresponding to a most violated inequality (9), then $C^{i_r,i_{r+1}}$ is given by $K_E \in \mathcal{E}^{i_r i_{r+1}}(l_r \leftrightarrow |s^{i_r}|, 1 \leftrightarrow l_{r+1})$ that maximizes $\sum_{e\in K_E} x_e^*$. Accordingly, let $H = (V, E')$ be the directed graph with the same nodes as G and two arcs (v_l^i, v_m^j), $(v_m^j, v_l^i) \in E'$ for every edge $\{v_l^i, v_m^j\}$ of G. Arc (v_l^i, v_m^j) in H represents the presence of edge $\{v_l^i, v_m^j\}$ in the reference mixed cycle, with $i = i_r$ and $j = i_{r+1}$ for some $1 \le r \le t$. Accordingly, if the weight w_a of an arc $a = (v_l^i, v_m^j) \in E'$ is defined as

$$w_a := 1 - \max_{K_E \in \mathcal{E}^{ij}(l\leftrightarrow|s^i|, 1\leftrightarrow m)} \sum_{e\in K_E} x_e^*,$$

then the most violated lifted mixed cycle inequality whose reference mixed cycle contains the arc $(v_{l_1-1}^{i_1}, v_{l_1}^{i_1}) \in A_p$ (if any such inequality exists) corresponds to a shortest arc-weighed path in H from $v_{l_1}^{i_1}$ to $v_{l_1-1}^{i_1}$. Indeed, if this path, say P, is given by arcs $(v_{l_1}^{i_1}, v_{l_2}^{i_2}), \ldots, (v_{l_t}^{i_t}, v_{l_1-1}^{i_1})$, the violation is given by

$$\sum_{r=1}^{t} \sum_{e\in C^{i_r,i_{r+1}}} x_e^* - t + 1 = 1 - \sum_{r=1}^{t}(1 - \sum_{e\in C^{i_r,i_{r+1}}} x_e^*) = 1 - \sum_{a\in P} w_a,$$

i.e., the inequality is violated if and only if the weight of P is smaller than 1.

Computing the weights for the arcs in E' can be done in $O(n^3)$ by finding the all pairs shortest paths in each of the $\binom{k}{2}$ pairgraphs (and is already done for the separation of maximal clique inequalities). If some of the weights turn out to be negative, then we have a violated maximal clique inequality. Otherwise, we can compute the shortest paths by Dijkstra's algorithm. The running time for each call to Dijkstra's algorithm is $O(n^2)$, since graph H has n nodes and $O(n^2)$ edges. Hence, the total running time is $O(n^3)$, since Dijkstra's algorithm is called $n - k$ times, once for each candidate pair $v_{l_1-1}^{i_1}, v_{l_1}^{i_1}$. □

Figure 8 illustrates the definition of H. To accelerate the separation, we sparsify graph H as follows (see also [23]). We add arcs (v_l^i, v_{l+1}^i) for $i = 1, \ldots, k$; $l = 1, \ldots, |s^i|-1$, assigning them weight 0, and delete all edges (v_l^i, v_m^j) whose weight is not strictly smaller than the weight of at least one of (v_{l+1}^i, v_m^j) or (v_l^i, v_{m-1}^j). For every pair v_l^i, v_m^j, $i \ne j$,

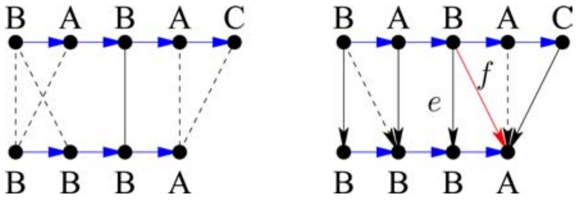

Fig. 8. Illustration of the definition and sparsification of graph H in the separation of lifted mixed cycle inequalities

since $w_{(v_l^i, v_m^j)} \leq w_{(v_{l+1}^i, v_m^j)}$ and $w_{(v_l^i, v_m^j)} \leq w_{(v_l^i, v_{m-1}^j)}$ for all j, m, the cost of the shortest path from v_l^i to v_m^j in the sparsified graph is equal to $w_{(v_l^i, v_m^j)}$. Therefore the shortest path from $v_{l_1-1}^{i_1}$ to $v_{l_1}^{i_1}$ in the sparsified graph H that does not contain arc $(v_{l_1-1}^{i_1}, v_{l_1}^{i_1})$ corresponds to a shortest path from $v_{l_1-1}^{i_1}$ to $v_{l_1}^{i_1}$ in the original graph H and viceversa. It is easy to check that all arcs (v_l^i, v_m^j) such that $x^*_{\{v_l^i, v_m^j\}} = 0$ are deleted from H in this way. For instance, edge f in figure 8 may be deleted.

Generalized transitivity inequalities

Theorem 6. *Given a point* $(x^*, y^*) \in \mathbb{R}_+^{|E|+|A_g|}$*, determining whether there is a generalized transitivity inequality (10) violated by* (x^*, y^*) *is solvable in* $O(n^4)$ *time.*

Proof. We fix the three strings $s^{i_1}, s^{i_2}, s^{i_3}$ along with $v_{l_1}^{i_1} \in V^{i_1}$. Then we need to check the existence of two sets $S^2 \subseteq V^{i_2}$ and $S^3 \subseteq V^{i_3}$ such that

$$\sum_{v_{l_2}^{i_2} \in S^2} x^*_{\{v_{l_1}^{i_1}, v_{l_2}^{i_2}\}} + \sum_{v_{l_3}^{i_3} \in S^3} x^*_{\{v_{l_1}^{i_1}, v_{l_3}^{i_3}\}} - \sum_{v_{l_2}^{i_2} \in S^2} \sum_{v_{l_3}^{i_3} \in S^3} x^*_{\{v_{l_2}^{i_2}, v_{l_3}^{i_3}\}} > 1.$$

Consider the complete bipartite graph $B = (V^{i_2} \cup V^{i_3}, F)$ and assign to each vertex $v_{l_2}^{i_2} \in V^{i_2}$ profit $p_{v_{l_2}^{i_2}} := x^*_{\{v_{l_1}^{i_1}, v_{l_2}^{i_2}\}}$, to each vertex $v_{l_3}^{i_3} \in V^{i_3}$ profit $p_{v_{l_3}^{i_3}} := x^*_{\{v_{l_1}^{i_1}, v_{l_3}^{i_3}\}}$, and to each edge $\{v_{l_2}^{i_2}, v_{l_3}^{i_3}\} \in F$ cost $c_{\{v_{l_2}^{i_2}, v_{l_3}^{i_3}\}} := x^*_{\{v_{l_2}^{i_2}, v_{l_3}^{i_3}\}}$. Clearly, finding the inequality (10) that is most violated is equivalent to finding subsets $S^2 \subseteq V^{i_2}$ and $S^3 \subseteq V^{i_3}$ such that

$$\sum_{v_{l_2}^{i_2} \in S^2} p_{v_{l_2}^{i_2}} + \sum_{v_{l_3}^{i_3} \in S^3} p_{v_{l_3}^{i_3}} - \sum_{v_{l_2}^{i_2} \in S^2} \sum_{v_{l_3}^{i_3} \in S^3} c_{\{v_{l_2}^{i_2}, v_{l_3}^{i_3}\}}$$

is maximized. Letting $\overline{S}^2$ and $\overline{S}^3$ represent, respectively, $V^{i_2} \setminus S^2$ and $V^{i_3} \setminus S^3$, this is equivalent to finding subsets $\overline{S}^2 \subseteq V^{i_2}$ and $\overline{S}^3 \subseteq V^{i_3}$ such that

$$\sum_{v_{l_2}^{i_2} \in \overline{S}^2} p_{v_{l_2}^{i_2}} + \sum_{v_{l_3}^{i_3} \in \overline{S}^3} p_{v_{l_3}^{i_3}} + \sum_{v_{l_2}^{i_2} \in V^{i_2} \setminus \overline{S}^2} \sum_{v_{l_3}^{i_3} \in V^{i_3} \setminus \overline{S}^3} c_{\{v_{l_2}^{i_2}, v_{l_3}^{i_3}\}} \tag{15}$$

is minimized. We show that the second version calls for an $\{s, t\}$-cut of minimum value in a suitable directed network. This network has node set $\{s, t\} \cup V^{i_2} \cup V^{i_3}$, where s plays the role of the *source* and t of the *sink*, an arc $(s, v_{l_2}^{i_2})$ of capacity $p_{v_{l_2}^{i_2}}$ for each $v_{l_2}^{i_2} \in V^{i_2}$, an arc $(v_{l_3}^{i_3}, t)$ of capacity $p_{v_{l_3}^{i_3}}$ for each $v_{l_3}^{i_3} \in V^{i_3}$, and an arc $(v_{l_2}^{i_2}, v_{l_3}^{i_3})$ of capacity $c_{\{v_{l_2}^{i_2}, v_{l_3}^{i_3}\}}$ for each $v_{l_2}^{i_2} \in V^{i_2}$ and $v_{l_3}^{i_3} \in V^{i_3}$. Recall that the value of an $\{s, t\}$-cut defined by node set S, with $s \in S$, $t \in \overline{S}$, is given by the sum of the capacities of all arcs with tail in S and head in $\overline{S}$, where $\overline{S}$ is the set of nodes outside S. Letting $\overline{S}^2 := V^{i_2} \cap \overline{S}$ and $\overline{S}^3 := V^{i_3} \cap S$, it is simple to verify that the value of this cut is given by (15). Note that it may well be the case that either $\overline{S}^2 = V^{i_2}$ or $\overline{S}^3 = V^{i_3}$ in the above problem (i.e., either $S^2 = \emptyset$ or $S^3 = \emptyset$), which means that no inequality (10) is violated since, by equations (2) both $\sum_{v_l^{i_2} \in V^{i_2}} x^*_{\{v_{l_1}^{i_1}, v_l^{i_2}\}} \leq 1$ and $\sum_{v_m^{i_3} \in V^{i_3}} x^*_{\{v_{l_1}^{i_1}, v_m^{i_3}\}} \leq 1$.

In the separation procedure above, for each triple $s^{i_1}, s^{i_2}, s^{i_3}$ and $l_1 \in \{1, \dots, |s^{i_1}|\}$, we have to compute a maximum flow in a network with $O(|s^{i_2}| + |s^{i_3}|)$ nodes and $O(|s^{i_2}||s^{i_3}|)$ arcs, which can be done in $O(|s^{i_2}|^3 + |s^{i_3}|^3)$ time. The overall complexity is $O(\sum_{i_1=1}^k \sum_{i_2=1}^k \sum_{i_3=1}^k |s^{i_1}|(|s^{i_2}|^3 + |s^{i_3}|^3)) = O(n^4)$. □

As before, if each string has approximately the same length n/k, the complexity is proportional to n^4/k. Note that the problem solved in the proof of Theorem 6 is a generalization of vertex cover in a bipartite graph (arising when the edge costs are are either 0 or ∞), and in fact the solution method above is a simple extension of the method for this problem.

The computation can be sped up by ignoring all edges $\{v_l^{i_2}, v_m^{i_3}\}$ with $x^*_{\{v_l^{i_2}, v_m^{i_3}\}} = 0$, sparsifying the network accordingly.

3.2. Further issues

Our method uses the branch-and-cut framework SCIL, which is described in detail in reference [26]. In this section we will not illustrate this general methodology but only the additional peculiarities of our approach.

A compact representation of maximal clique inequalities It is known [17, 4] that whenever an exponentially-large family of inequalities admits a separation algorithm corresponding to the solution of a polynomial-size LP, then "merging" this LP with the original LP relaxation (without any inequality in the family considered) is equivalent to imposing all inequalities in the family. This comes at the cost of introducing additional variables and constraints to the original LP, and often does not pay off in terms of running time, with some notable exceptions [5].

In our case, it is easy to see that all our separation procedures can be interpreted as the solution of a polynomial-size LP, i.e., it would be immediate to construct an LP of polynomial size that is equivalent to the LP relaxation of (1–6) with the addition of (8), (9), (10). However, extensive computational testing on the instances that we considered showed that the only class of inequalities among those considered that is better to replace with additional variables and constraints as mentioned above are the maximal clique inequalities "without gap variables", namely those among (8) for which $l_b = 1, l_e = |s^i|$, and $K_A = \{(v_1^i, v_{|s^i|}^i)^j\}$.

Recall that in the separation of these inequalities, a maximal clique corresponds to a path in pairgraph P^{ij} from node $n_{\{s_1^i, s_{|s^j|}^j\}}$ to node $n_{\{s_{|s^i|}^i, s_1^j\}}$. For a given solution (x^*, y^*), a maximal clique inequality (8) is violated if there is a path between these two nodes of length more than $1 - y^*_{(v_1^i, v_{|s^i|}^i)^j}$, i.e., if the value of the following LP having one variable $z_{\{v_l^i, v_m^j\}}$ associated with each node $n_{\{v_l^i, v_m^j\}}$ of P^{ij} (and corresponding to the dual of the customary LP formulation of longest path on a directed acyclic graph, using arc variables) is more than $1 - y^*_{(v^i_1, v^i_{|s^i|})^j}$:

$$\min z_{\{v^i_{|s^i|}, v^j_1\}}$$

$$\begin{aligned}
&\text{subject to}\\
&z_{\{v^i_1, v^j_{|s^j|}\}} = x^*_{\{v^i_1, v^j_{|s^j|}\}},\\
&z_{\{v^i_{l+1}, v^j_m\}} \geq z_{\{v^i_l, v^j_m\}} + x^*_{\{v^i_{l+1}, v^j_m\}}, \quad l = 1, \ldots, |s^i| - 1; m = 1, \ldots, |s^j|,\\
&z_{\{v^i_l, v^j_{m-1}\}} \geq z_{\{v^i_l, v^j_m\}} + x^*_{\{v^i_l, v^j_{m-1}\}}, \quad l = 1, \ldots, |s^i|; m = 2, \ldots, |s^j|.
\end{aligned}$$

Accordingly, in order to enforce that all maximal clique inequalities without gap variables are satisfied in the LP relaxation we consider, we add variables $z_{\{v^i_l, v^j_m\}}$ for $\{v^i_l, v^j_m\} \in E$ along with the following constraints:

$$\begin{aligned}
&z_{\{v^i_{|s^i|}, v^j_1\}} \leq 1 - y_{(v^i_1, v^i_{|s^i|})^j}, \quad i, j = 1, \ldots, k; i < j,\\
&z_{\{v^i_1, v^j_{|s^j|}\}} = x_{\{v^i_1, v^j_{|s^j|}\}}, \quad i, j = 1, \ldots, k; i < j,\\
&z_{\{v^i_{l+1}, v^j_m\}} \geq z_{\{v^i_l, v^j_m\}} + x_{\{v^i_{l+1}, v^j_m\}},\\
&\qquad l = 1, \ldots, |s^i| - 1; m = 1, \ldots, |s^j|; i, j = 1, \ldots, k; i < j,\\
&z_{\{v^i_l, v^j_{m-1}\}} \geq z_{\{v^i_l, v^j_m\}} + x_{\{v^i_l, v^j_{m-1}\}},\\
&\qquad l = 1, \ldots, |s^i|; m = 2, \ldots, |s^j|; i, j = 1, \ldots, k; i < j.
\end{aligned}$$

This corresponds to introducing $O(n^2)$ new variables and constraints.

Variable reduction Even for rather small instances, the number of variables in our ILP formulation is very large, namely there are $\sum_{i=1}^{k-1} \sum_{j=i+1}^{k} |s^i||s^j| = O(n^2)$ edge variables and $\sum_{i=1}^{k} (k-1)\binom{|s^i|+1}{2} = O(n^3)$ arc variables. For example, an instance with 5 strings of length 100 has 201, 000 variables. Most of these variables are very unlikely to take the value 1 in an optimal solution. In this section, we describe a simple but successful method to reduce the number of variables.

Assume we know a lower bound L on the optimum, typically found by a heuristic. For every variable v, we compute an upper bound U_v on the value of the optimal alignment in which variable v takes the value 1, and we can permanently fix the variable to 0 if $U_v \leq L$. These upper bounds are determined as follows.

First of all, we compute all optimal alignments between two strings s^i, s^j, for $i, j = 1, \ldots, k, i < j$ in $O(\sum_{i=1}^{k-1} \sum_{j=i+1}^{k} |s^i||s^j| \max\{|s^i|, |s^j|\}) = O(n^3)$ time by the following modification of the standard dynamic programming technique for the case in which $w_a = 0$ for $a \in A_g$ [15].

As already done, we implicitly represent the dynamic programming recursion by formulating the problem on a directed acyclic graph Q^{ij} having a node $n_{\{v^i_l, v^j_m\}}$ for

$l = 1, \ldots, |s^i| + 1$ and $m = 1, \ldots, |s^j| + 1$. For $l = 1, \ldots, |s^i|$ and $m = 1, \ldots, |s^j|$, node $n_{\{v^i_l, v^j_m\}}$ has a *green* outgoing arc $(n_{\{v^i_l, v^j_m\}}, n_{\{v^i_{l+1}, v^j_{m+1}\}})$ having weight $w_{\{v^i_l, v^j_m\}}$, representing the realization of edge $\{v^i_l, v^j_m\}$ in the alignment. Moreover, node $n_{\{v^i_l, v^j_m\}}$ has *red* outgoing arcs $(n_{\{v^i_l, v^j_m\}}, n_{\{v^i_p, v^j_m\}})$ for $p = l+1, \ldots, |s^i|+1$, each having weight $w_{(v^i_l, v^i_{p-1})^j}$ and representing the realization of gap arc $(v^i_l, v^i_{p-1})^j$ in the alignment, as well as *red* outgoing arcs $(n_{\{v^i_l, v^j_m\}}, n_{\{v^i_l, v^j_p\}})$ for $p = m + 1, \ldots, |s^j| + 1$, each having weight $w_{(v^j_m, v^j_{p-1})^i}$ and representing the realization of gap arc $(v^j_m, v^j_{p-1})^i$ in the alignment. It is easy to check that the optimal alignment between strings s^i and s^j corresponds to the longest (arc-weighed) path from $n_{\{v^i_1, v^j_1\}}$ to $n_{\{v^i_{|s^i|+1}, v^j_{|s^j|+1}\}}$ in Q^{ij} with the property that no consecutive arcs in the path are red. Such a path can be found in linear time in the size of Q^{ij}, which is $O(|s^i||s^j| \max\{|s^i|, |s^j|\})$, by using two labels for each node, one for the longest path and the other for the longest path whose last arc is green. Note that this computation yields the value $\pi^{ij}(l, m)$ of the optimal alignment of substring $s^i_1 \ldots s^i_l$ of s^i with substring $s^j_1 \ldots s^j_m$ of s^j, for $l = 1, \ldots, |s^i|$ and $m = 1, \ldots, |s^j|$. Moreover, in a completely analogous way and in the same running time one can find the value $\omega^{ij}(l, m)$ of the optimal alignment of substring $s^i_l \ldots s^i_{|s^i|}$ of s^i with substring $s^j_m \ldots s^j_{|s^j|}$ of s^j.

The sum of the values of all the pairwise optimal alignments is clearly an upper bound on the value of the optimal alignment, called the *pairwise upper bound*. For convenience, let σ^{ij} denote the sum of the values of the optimal alignments between all string pairs different from $\{s^i, s^j\}$.

After having computed the above values, for every alignment variable x_e corresponding to edge $e = \{v^i_l, v^j_m\}$, we let

$$U_{x_e} = \sigma^{ij} + \pi^{ij}(l-1, m-1) + w_e + \omega^{ij}(l+1, m+1),$$

and perform the associated reduction of the alignment variables in $O(n^2)$ time, whereas for every gap variable y_a corresponding to gap arc $a = (v^i_l, v^i_m)^j$, we let

$$U_{y_a} = \sigma^{ij} + \max_{v^j_p \in V^j} (\pi^{ij}(l-1, p) + \omega^{ij}(m+1, p+1)).$$

and perform the associated reduction of the gap variables in $O(n^3)$ time.

Moreover, we can further reduce the number of alignment variables by computing the following tighter upper bound. Assume edge $e = \{v^i_l, v^j_m\}$ is realized. For every string h, there must be a node $v^h_p \in V^h$ such that all characters before s^h_p are aligned only with characters before s^i_l in string i as well as with characters before s^j_m in string j, and all characters after s^h_{p+1} are aligned only with characters after s^i_{l+1} in string i as well as with characters after s^j_{m+1} in string j. If τ^{ij} denotes the sum of the values of the optimal alignments of string pairs that do not contain s^i or s^j, the tighter upper bound is

$$\begin{aligned} U_{x_e} &= \tau^{ij} + \pi^{ij}(l-1, m-1) + w_e + \omega^{ij}(l+1, m+1) \\ &+ \sum_{h \in \{1,\dots,k\} \setminus \{i,j\}} \max_{v_p^h \in V^h} (\pi^{hi}(p, l) + \omega^{hi}(p+1, l+1) \\ &+ \pi^{hj}(p, m) + \omega^{hj}(p+1, m+1)), \end{aligned}$$

and the associated further reduction of the alignment variables takes $O(n^3)$ time.

Heuristics In order to determine an initial feasible solution to the problem, we use the following heuristic. After having computed all the optimal pairwise alignments between strings, we remove all variables for which the upper bounds computed in the variable reduction phase are smaller with respect to the optimal pairwise alignment value by more than a given parameter δ. Then we find the optimal solution for this reduced problem by our branch-and-cut method. The value of δ is set to 25 for the instances in our test bed, in order to ensure that that the optimal alignment for the resulting graph can easily be computed.

Furthermore, we implemented the following LP-based primal heuristic that is applied at each LP iteration. We sort the edge variables according to decreasing LP values. Then, we iterate over the edges, and include each of them in the solution if its addition does not create a mixed cycle. Finally we add the edges transitively implied by the included ones.

4. Computational experiments

We tested our implementation, called COSA (COmbinatorial Sequence Alignment) using instances of the BAliBase library [27], containing 4 to 6 strings of about 50-130 amino acids each, subdivided into three groups according to different "identity levels", that represent an index of the similarity of the strings (Group V1: identity $> 35\%$, Group V2: identity $20-40\%$, Group V3: identity $< 25\%$). The library also offers an evaluation program that computes a score between 0 and 1 indicating the percentage of correctly aligned characters with respect to reference alignments that have been hand-created using structural information. In addition to the instances, [27] also contains a survey of the performance of the different alignment programs available.

In order to define the objective function, we computed the optimal alignment for all the instances for edge costs chosen among a set of amino acid substitution matrices and gap costs chosen among a set of affine and convex functions from the literature (recall the discussion in Section 1). Eventually, we chose the costs giving the best score according to the database evaluation program, as discussed in Section 4.3, namely substitution matrix `blosum62` for edge costs and convex function $8+2l+2\sqrt{l}$ for gap costs, where l is the number of characters in the gap.

The weights w in our maximization objective function can be negative in general (being all negative for gap arcs). In order to have nonnegative weights, we added 2ϵ to the weight w_e of each edge $e \in E$ and $l\epsilon$ to the weight w_a of each arc $a \in A_g$ corresponding to a gap with l characters, where the parameter ϵ is set to 12 in our implementation. It is easy to check that, due to constraints (2), this changes the value of all feasible solutions by the same constant.

We ran the experiments on an Intel Pentium Xeon (3.06GHz) with 4GB main memory with a CPU time limit of 10 hours. The LP solver used was CPLEX 9.0. Furthermore our implementation is based on LEDA [19] and SCIP [1]. All computing times in the tables are given in the format “hh:mm:ss”.

4.1. Initial experiments

In this section we discuss and justify experimentally some key choices in our implementation.

Solution of the LPs As the solution of the LPs is the computational bottleneck of the approach, we tried different methods. For the easy instances, reoptimizing the LPs with the dual simplex algorithm was slightly faster than solving them from scratch with the barrier method. On the other hand, contrary to what is generally observed in other cases, for the challenging instances, the barrier method widely outperformed the simplex algorithm, provided one resigned to crossover to a basic solution. (As mentioned also below, it turned out that our separation algorithms were fairly fast even for the dense LP solutions given by the barrier method.) Table 1 shows the total running times to solve the eight instances of the group V2 for which our method found an optimal solution, where, for the dual simplex algorithm, results are reported for the two pricing options of CPLEX (automatic and steepest descent).

In accordance with the results shown, in our implementation we use the barrier method without crossover for the solution of each LP. Furthermore, after each LP solution, we remove all inequalities whose associated dual variable is zero, since, although the total number of LPs to be solved increased slightly with this modification, the overall solution time decreased.

As the number of variables was quite large, we also tried to apply various pricing policies, including pricing based on a relaxation of the problem (in our case, the relaxation associated with the pairwise upper bound) following the approach described in [10]. Unfortunately, for all approaches tried, it turned out that all variables not pruned by variable reduction were eventually priced-in, and the overall LP solution time was higher than the one achieved by inserting all variables since the first LP. Therefore, we do not apply any pricing in our implementation.

Table 1. Comparison of dual simplex and barrier for the solution of the LPs

Name	Dual_Auto	Steepest_Descent	Barrier
1aab	23	20	16
1fjlA	1:05:34	47:58	7:51
1hfh	6:06:12	6:33:35	29:42
1hpi	11:56	11:40	7:49
1csy	1:14:58	1:41:54	9:24
1pfc	LIMIT	LIMIT	40:38
1tgxA	7:40	7:43	2:08
3cyr	42:29	45:26	6:55

Table 2. Integrality gaps and associated LP solution time depending on the classes of inequalities separated

Name	w/ all ineq.		w/o (10)		w/o (9), (10)		w/o (8), (10)		w/o (8), (9), (10)	
	gap	time	gap	time	gap	time	gap	time	gap	time
1aab	0	16	0	16	25	14	68	31	111	2
1fjlA	0	7:51	0	7:51	71	8:49	114	10:01	206	34
1hfh	0	29:42	2	22:38	158	9:44	399	15:55	521	42
1hpi	2	5:02	10	2:19	52	1:34	154	1:14	195	5
1csy	0	9:24	0	9:24	138	5:48	353	6:55	433	22
1pfc	0	40:38	0	40:38	101	12:44	234	8:53	367	47
1tgxA	0	2:08	0	2:08	169	1:18	144	2:27	202	4
3cyr	0	6:55	0	6:55	99	6:17	228	4:23	284	15

Separation After extensive experiments, we selected the following separation strategy. As the time for the separation algorithms was quite small compared to the solution time of the LPs, in our implementation we run all separation algorithms after each LP iteration, keeping track of all the violated inequalities found. Among these inequalities, we add to the LP the m with highest violation, where m is set to 500.

We tried some simple and fast separation heuristics to be run before the exact separation algorithms in Section 3.1, at least for the first LPs. However, when these heuristics were used, both the number of LPs to be solved and the total running time increased (keep in mind that the time for exact separation was small). Therefore, we apply only exact separation in our implementation.

In order to test the effectiveness of the three classes of inequalities (8) (with gap variables, since those without gap variables are represented in compact form), (9) and (10), we tried not to separate some of them and observed the outcome. Resigning to separate either (8) or (9) made the LP gap too big to solve any of the instances considered. This was not the case for inequalities (10), although their addition reduced the integrality gap in some cases, sometimes allowing one to skip branching. Moreover, for the instances that were not solved to optimality, the upper bound at the time limit was notably worse if (10) were not separated. In Table 2, we report the time to solve the LP relaxation at the root node and the associated (absolute) integrality gap, for the eight instances in group V2 that we could solve to optimality, in case some of the classes of inequalities were not separated. Accordingly, in our implementation we separate all three classes of inequalities.

The branch-and-cut tree There are only two instances in our test bed that were not solved to optimality at the root node. On these instances, we experimented with different branching and enumeration rules. For the enumeration, the best first policy outperformed any other rule, as the time for switching to another subproblem was negligible compared to the time for solving the associated LP relaxation. For the branching rule, we tried both branching on the variable whose LP value is closer to 0.5 and the strong branching mechanism with different bounds on the number of variables considered and on the number of simplex iterations in the associated subproblems (applying strong branching requires a crossover to a basic LP solution), and the corresponding times were similar for all parameter settings. Accordingly, in our implementation we use best first enumeration together with simple branching on the most fractional variable.

4.2. Results

The results of our code are reported in Table 3, in which the columns have the following meaning:

Name: Name of the Instance, along with an indication (k, n) of the number of strings and the overall number of characters;
PAV: Number of alignment variables that were pruned by the reduction procedure of Section 3.2;
LPAV: Number of alignment variables that remained;
PGV: Number of gap variables that were pruned by the reduction procedure of Section 3.2;
LPGV: Number of gap variables that remained;
Heur: Value of the initial feasible solution found by the method of Section 3.2;
PUB: Pairwise upper bound;
Root: Value of the LP bound at the root node – if the LP was not solved to optimality this is the value of the LP solution at termination;
Opt: Optimal solution value – if the instance was not solved to optimality the possible range of values.
No: Number of nodes in the branch-and-cut tree;
LP: Number of LPs solved;
Time: Running time;

Horizontal lines in the table separate the three groups of instances with different identity levels (the three groups appear in decreasing order of identity level).

The running time for the branch-and-cut algorithm was dominated by the time required for solving the LPs, and the difference between the total CPU time and the LP time was less than seven minutes for all instances, even if separation was applied to the dense LP solutions given by the barrier method. In almost all cases, either the optimal LP solution was integer, or the time limit was exceeded during the solution of the LP at the root node. Note that the time limit was not checked during calls to the LP solver, so the overall running time could exceed the 10 hours time limit. Considering the only two instances that were not solved at the root node, the root node LP took about 5 out of 7 minutes for instance 1dox and about 2.6 out of 7 minutes for instance 1hpi.

For all the instances, separation took less than 10 seconds for each iteration, whereas the time to solve the LPs heavily depended on the number of variables that remained in the LP after preprocessing. We could solve to proven optimality 18 out of 27 instances. The instances with larger identity level were easier, since the difference between the optimal solution and the pairwise upper bound was smaller and thus more variables could be eliminated in the preprocessing phase. In general, the initial heuristic, which is also based on our branch-and-cut approach, was capable of producing solutions of very good quality (for none of the instances for which the time limit was reached a better solution was found).

Table 3. Results of our method for the instances in our test bed

Name	PAV	LPAV	PGV	LPGV	Heur	PUB	Root	"Opt"	No	LP	"Time"
1aho (5/320)	32090	8854	15096	26568	883.2	1059.0	899.8	899.8;	1	13	3:14.
1csp (5/339)	45152	808	46007	673	1504.9	1521.1	1504.9	1504.9;	1	5	2.
1dox (4/374)	48553	3894	43184	9850	804	876.3	812.2	805.9;	19	65	7:08.
1fkj (5/517)	93562	13316	64791	43309	1573.0	1735.2	1636.9	1636.9;	1	11	5:15.
1fmb (4/400)	59337	652	60195	438	1352.3	1367.9	1352.3	1352.3;	1	6	2.
1krn (5/390)	59224	1523	57031	4961	1581.6	1616.7	1581.6	1581.6;	1	14	17.
1plc (5/470)	85540	2789	79122	10302	1802.0	1852.2	1802.0	1802.0;	1	10	34.
2fxb (5/287)	31760	1162	28612	5012	1354.8	1403.8	1354.8	1354.8;	1	6	4.
2mhr (5/572)	128992	1859	125375	6733	2398.7	2435.2	2398.7	2398.7;	1	17	30.
9rnt (5/499)	97588	1999	92145	8507	2563.3	2612.0	2578.0	2578.0;	1	10	20.
1aab (4/291)	29111	2606	25860	6447	314.9	367.9	314.9	314.9;	1	7	16.
1fjlA (6/398)	59053	6900	38181	29059	751.4	858.7	751.4	751.4;	1	12	7:51.
1hfh (5/606)	128442	18375	91912	56504	927.5	1077.9	927.5	927.5;	1	18	29:42.
1hpi (4/293)	27919	4233	23726	9031	391.0	464.4	397.0	391.0;	10	50	7:49.
1csy (5/510)	91320	12715	62629	42451	643.6	783.0	643.6	643.6;	1	14	9:24.
1pfc (5/560)	106039	19374	66028	60640	963.5	1151.4	979.1	979.1;	1	27	40:38.
1tgxA (4/239)	17492	3915	13491	8328	218.1	295.3	227.0	227.0;	1	14	2:08.
1ycc (4/426)	52692	15164	41710	27575	275.3	388.1	314.0	[275.3,314.0];	1	51	10:02:04.
3cyr (4/414)	54682	9537	44215	20843	545.4	662.9	545.4	545.4;	1	16	6:55.
451c (5/400)	28290	35631	1569	63547	29.4	380.9	200.7	[29.4,200.7];	1	45	12:33:06.
1aboA (5/297)	5425	29554	37	37059	–645.5	–299.4	–399.3	[–645.5,–399.3];	1	34	10:08:14.
1idy (5/269)	14774	14141	4168	25432	–353.7	–210.7	–268.1	[–353.7,–268.1];	1	120	10:23:22.
1r69 (4/277)	16837	11876	10300	19070	–265.0	–149.3	–213.7	[–265.0,–219.4];	9	149	10:06:15.
1tvxA (4/242)	6567	15264	880	21836	–381.9	–176.4	–226.7	[–381.9,–237.8];	1	109	10:28:33.
1ubi (4/327)	21346	18644	10734	30180	–352.4	–193.4	–204.0	[–352.4,–204.0];	1	92	10:30:51.
1wit (5/484)	66712	26908	30617	64383	–181.4	7.0	–152.0	[–181.4,–152.0];	1	30	10:37:44.
2trx (4/362)	26155	22932	10100	39748	–238.9	–15.2	–118.3	[–238.9,–118.3];	1	31	10:00:06.

4.3. Comparison with other programs

We compared our algorithm with the most recent structural alignments algorithms, considering the overall best performing programs from the survey [27]: PRRP, ClustalX, and Dialign, together with a recently published program T-Coffee [22], which generally outperforms the other programs. The source code of all these programs was downloaded and they were run with default parameters. We disregarded all other programs, since they perform significantly worse (see [27]). Within this comparison, we used the results from our initial heuristic, based on branch-and-cut, since these can be obtained for all the instances in reasonable time. The quality of the heuristic solution and the optimal solution with respect to the score did not significantly differ.

The results are given in Table 4, in which the first column contains the name of the instance and the subsequent columns the results for the alignment programs considered. Each entry in the table is the score computed by the evaluation program provided with the BAliBase library. In brackets we give the percentage relative to the best program for this example. In addition, we provide an average for each group. The results show that our approach was on average superior to all other programs. Our comparison with other programs reveals that computing near-optimal alignments does pay off in terms of quality, because our approach produced on average the biologically most meaningful alignments.

On the other hand, it has to be remarked that, for the larger instances in the library, the LPs that arise are at present not solvable within acceptable time by our approach, unless we use a very small value of δ in the heuristic preprocessing, which has the effect

Table 4. Comparison of the Core scores for COSA, TCOFFE and other programs. In the first column we give the name of the instance and its size (in number of strings/total number of characters)

Data	COSA	T-Coffee	PRRP	ClustalX	Dialign
1aho (5/320)	1 (100)	1 (100)	0.937 (94)	0.857 (86)	0.943 (94)
1csp (5/339)	0.993 (100)	0.993 (100)	0.967 (97)	0.993 (100)	0.967 (97)
1dox (4/374)	0.918 (100)	0.918 (100)	0.922 (100)	0.911 (99)	0.848 (92)
1fkj (5/517)	0.987 (100)	0.987 (100)	0.92 (93)	0.917 (93)	0.895 (91)
1fmb (4/400)	0.978 (100)	0.978 (100)	0.967 (99)	0.978 (100)	0.961 (98)
1krn (5/390)	1 (100)	1 (100)	1 (100)	0.992 (99)	0.84 (84)
1plc (5/470)	0.947 (97)	0.947 (97)	0.976 (100)	0.935 (96)	0.838 (86)
2fxb (5/287)	0.963 (98)	0.963 (98)	0.963 (98)	0.981 (100)	0.963 (98)
2mhr (5/572)	1 (100)	0.996 (100)	1 (100)	1 (100)	0.922 (92)
9rnt (5/499)	0.99 (99)	0.995 (100)	0.977 (98)	0.995 (100)	0.854 (86)
avg.	0.978 (100)	0.978 (100)	0.963 (98)	0.956 (98)	0.903 (92)
1aab (4/291)	0.929 (100)	0.929 (100)	0.929 (100)	0.869 (94)	0.929 (100)
1fjlA (6/398)	1 (100)	0.993 (99)	1 (100)	1 (100)	1 (100)
1hfh (5/606)	0.958 (100)	0.922 (96)	0.869 (91)	0.694 (72)	0.259 (27)
1hpi (4/293)	0.841 (99)	0.727 (85)	0.852 (100)	0.841 (99)	0.545 (64)
1csy (5/510)	0.975 (100)	0.953 (98)	0.946 (97)	0.907 (93)	0.865 (89)
1pfc (5/560)	0.892 (94)	0.944 (100)	0.861 (91)	0.826 (88)	0.438 (46)
1tgxA (4/239)	0.641 (69)	0.7 (75)	0.865 (93)	0.929 (100)	0.459 (49)
1ycc (4/426)	0.979 (99)	0.921 (93)	0.958 (97)	0.991 (100)	0.636 (64)
3cyr (4/414)	0.77 (100)	0.713 (93)	0.713 (93)	0.751 (98)	0.355 (46)
451c (5/400)	0.801 (100)	0.713 (89)	0.618 (77)	0.646 (81)	0.571 (71)
avg.	0.879 (100)	0.852 (97)	0.861 (98)	0.845 (96)	0.606 (69)
1aboA (5/297)	0.558 (89)	0.509 (81)	0.212 (34)	0.63 (100)	0.624 (99)
1idy (5/269)	0.646 (95)	0.065 (10)	0.681 (100)	0.415 (61)	0.031 (5)
1r69 (4/277)	0.493 (100)	0.28 (57)	0.427 (87)	0.48 (97)	0.107 (22)
1tvxA (4/242)	0.158 (72)	0.219 (100)	0.132 (60)	0.105 (48)	0 (0)
1ubi (4/327)	0.46 (61)	0.447 (59)	0.46 (61)	0.76 (100)	0.06 (8)
1wit (5/484)	0.854 (100)	0.792 (93)	0.833 (98)	0.721 (84)	0.417 (49)
2trx (4/362)	0.723 (98)	0.587 (80)	0.364 (50)	0.705 (96)	0.735 (100)
avg.	0.556 (100)	0.414 (74)	0.444 (80)	0.545 (98)	0.282 (51)

of eliminating a wide set of useful variables. Currently, our method is not competitive for these instances.

5. Conclusions and further work

A comparison of the known multiple sequence alignment tools shows that almost all of these programs are heuristics that try to find good approximations of the optimal multiple alignment. Each of these tools makes use of its own scoring function. We have presented for the first time an approach to general multiple sequence alignment with arbitrary gap costs. In [20], we have proved that, for example, the DIALIGN scoring function can easily be incorporated within our approach. Moreover, for some of the most interesting scoring functions, the integration into our approach is at present the only way to find multiple alignments that are guaranteed to be optimal.

Our implementation is competitive with or better than the currently best programs for the real-world benchmark instances of medium size, with a few strings of about a hundred characters each, for which our heuristic, based on branch-and-cut applied after having heuristically reduced the set of variables, is very successful. On the other hand, the solution of the LP relaxation of larger instances, even after the heuristic reduction of the set of variables, is too time consuming.

In order to overcome this drawback, further research should be devoted to (a) the implementation and testing of approximate algorithms for the solution of the LPs [3] and (b) the parallelization of the approach, based on the fact that barrier methods can be parallelized.

Appendix: Table of notation

$G = (V, E, A)$	gapped alignment graph
$V = V^1 \cup \cdots \cup V^k$	nodes corresponding to characters subdivided according to the strings
$v^i_l \in V^i$	node corresponding to the l-th character of string s^i
$E = \cup_{1 \le i < j \le k} E^{ij}$	undirected alignment edges subdivided according to the strings of the endpoints
$\{v^i_l, v^j_m\}$	alignment edge, whose choice represents the fact that the two corresponding characters are aligned
$A = A_p \cup A_g$	arcs
A_p	consecutivity arcs
$(v^i_l, v^i_{l+1}) \in A_p$	consecutivity arc, representing the fact that v^i_{l+1} follows v^i_l in string s^i
$A_g = \cup_{1 \le i,j \le k} A^{ij}$	gap arcs subdivided according to the associated (ordered) pair of strings
$(v^i_l, v^i_m)^j \in A^{ij}$	gap arc, whose choice represents the fact that no character in string s^i corresponding to nodes between v^i_l and v^i_m in V^i is aligned with a character from string s^j
w_e, w_a	cost of an edge/arc
$A^{ij}(l \leftrightarrow m)$	set of arcs in A^{ij} spanning $v^i_l, \ldots, v^i_m$ – we also allow $l = m + 1$, to denote the set of arcs that span either v^i_m or v^i_{m+1}
$\mathcal{E}^{ij}(l_b \leftrightarrow l_e, m_b \leftrightarrow m_e)$	collection of all maximal pairwise-incompatible sets of edges $S \subseteq E^{ij}$ such that, for each edge $\{v^i_l, v^j_m\} \in S$, $l_b \le l \le l_e$ and $m_b \le m \le m_e$

Acknowledgements. We are grateful to an anonymous referee for the very detailed and helpful report and Tobias Achterberg for his patience in explaining us the details of this branch-and-cut framework SCIP.

This work was partially supported by a fellowship within the ICSI Postdoc Program of the German Academic Exchange Service (DAAD), and by the German Federal Ministry of Education and Research, grant no. 0312705A "Berlin Center for Genome Based Bioinformatics". Moreover, part of this work was done while the second author was visiting the Max Planck Institut für Informatik in Saarbrücken.

References

1. Achterberg, T.: SCIP - a framework to integrate constraint and mixed integer programming. Technical Report 04-19, Zuse Institute Berlin, 2004. `http://www.zib.de/bib/pub/pw`

2. Altschul, S.F., Gish, W., Miller, W., Myers, E.W., Lipman, D.J.: Basic local alignment search tool. J. Mol. Biol. **215**, 403–410 (1990)
3. Bienstock, D.: Potential function methods for approximately solving linear programming problems, Theory and Practice. Kluwer Academic Publishers, Boston, 2002
4. Carr, R.D., Lancia, G.: Compact vs exponential-size lp relaxations. Operations Research Letters **30**, 57–65 (2002)
5. Carr, R.D., Lancia, G.: Compact optimization can outperform separation: A case study in structural proteomics. 4OR **2**, 221–233 (2004)
6. Carrillo, H., Lipman, D.J.: The multiple sequence alignment problem in biology. SIAM J. Appl. Math. **48** (5), 1073–1082 (1988)
7. Dayhoff, M., Schwartz, R., Orcut, B.: A model of evolutionary change in proteins. In: M. Dayhoff (ed.) Atlas of Protein Sequence and Structure, vol 5, National Biomedical Research Foundation, Washington, D.C., 1979, pp 345–352
8. Delcher, A., Kasif, S., Fleischmann, R., J. Peterson, W. O., Salzberg, S.: Alignment of whole genomes. Nucleic Acids Res **27**, 2369–2376 (1999)
9. Eppstein, D.: Sequence comparison with mixed convex and concave costs. J Algorithms (11), 85–101 (1990)
10. Fischetti, M., Toth, P.: A polyhedral approach to the asymmetric traveling salesman problem. Management Sci **43** (11), 1520–1536 (1997)
11. Garey, M., Johnson, D.: Computers and Intractability: A Guide to the Theory of NP-Completeness. W.H. Freeman, 1979
12. Golumbic, M.C.: Algorithmic graph theory and perfect graphs. Academic Press, New York, 1980
13. Gotoh, O.: An improved algorithm for matching biological sequences. J. Mol. Biol. **162**, 705–708 (1982)
14. Gupta, S., Kececioglu, J., Schaeffer, A.: 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)
15. Gusfield, D.: Algorithms on strings, trees and sequences: computer science and computational biology. Cambridge University Press, Cambridge, 1997
16. Henikoff, S., Henikoff, J.: Amino acid substitution matrices from protein blocks. Proceedings of the National Academy of Science **89**, 10915–10919 (1992)
17. Kipp Martin, R.: Using separation algorithms to generate mixed integer model reformulations. Oper. Res. Lett. **10**, 119–128 (1991)
18. Larmore, L., Schieber, B.: Online dynamic programming with applications to the prediction of rna secondary structure. In: Proceedings of the First Symposium on Discrete Algorithms 1990, pp 503–512
19. LEDA (Library of Efficient Data Types and Algorithms), 2004. `www.algo\-rith\-mic-solu\-tions.com`
20. Lenhof, H.-P., Morgenstern, B., Reinert, K.: An exact solution for the segment-to-segment multiple sequence alignment problem. Bioinformatics **15** (3), 203–210 (1999)
21. Lermen, M., Reinert, K.: The practical use of the $\mathcal{A}^*$ algorithm for exact multiple sequence alignment. J. Comput. Biol. **7**(5), 655–673 (2000)
22. Notredame, C., Higgins, D.G., Heringa, J.: T-coffee : A novel method for fast and accurate multiple sequence alignment. J. Mol. Biol. **302**, 205–217 (2000)
23. Reinert, K.: A Polyhedral Approach to Sequence Alignment Problems. PhD thesis, Universität des Saarlandes, 1999
24. Reinert, K., Lenhof, H.-P., Mutzel, P., Mehlhorn, K., Kececioglu, J.: A branch-and-cut algorithm for multiple sequence alignment. In: Proceedings of the First Annual International Conference on Computational Molecular Biology (RECOMB-97), 1997, pp 241–249
25. K. Reinert, J. Stoye, and T. Will. An iterative methods for faster sum-of-pairs multiple sequence alignment. BIOINFORMATICS 16(9):808–814, 2000.
26. SCIL–Symbolic Constraints for Integer Linear programming, 2002. `www.mpi-sb.mpg.de/SCIL`
27. Thompson, J.D., Plewniak, F., Poch, O.: BAliBASE: A benchmark alignment database for the evaluation of multiple alignment programs. Bioinformatics **15** (1), 87–88 (1999) `http://www-ig\-bmc.u-stras\-bg.fr/Bio\-Info/BA\-li\-BASE/prog_scores.html`
28. Wang, L., Jiang, T.: On the complexity of multiple sequence alignment. J. Comput. Biol. **1**, 337–348 (1994)