**Abstract**

**Motivation:** In molecular biology, sequence alignment is a crucial tool in studying the structure and function of molecules, as well as the evolution of species. In the segment-to-segment variation of the multiple alignment problem, the input can be seen as a set of non-gapped segment pairs (diagonals). Given a weight function that assigns a weight score to every possible diagonal, the goal is to choose a consistent set of diagonals of maximum weight. We show that the segment-to-segment multiple alignment problem is equivalent to a novel formulation of the Maximum Trace problem: the Generalized Maximum Trace (GMT) problem. Solving this problem to optimality, therefore, may improve upon the previous greedy strategies that are used for solving the segment-to-segment multiple sequence alignment problem. We show that the GMT can be stated in terms of an integer linear program and then solve the integer linear program using methods from polyhedral combinatorics. This leads to a branch-and-cut algorithm for segment-to-segment multiple sequence alignment.

**Results:** We report on our first computational experiences with this novel method and show that the program is able to find optimal solutions for real-world test examples.

**Contact:** kreinert@mpi-sb.mpg.de

**Introduction**

From a mathematical point of view, the alignment problem is an optimization problem: the goal is to find a ‘best’ alignment of a given sequence set. Here, the first question is how to define what a ‘good’ alignment is. While biologists are able to assess the quality of alignments from experience and knowledge, computer programs need mathematically defined objective functions or scoring schemes by which the quality of alignments can be evaluated. The problem is to find objective functions that are, in as many situations as possible, in accordance with biology in the sense that alignments are assigned high scores if, and only if, they are biologically meaningful.

Various objective functions have been proposed for sequence alignment. Most of them rely on an idea by Needleman and Wunsch (1970). Given a similarity matrix that gives a score to every possible pair of individual amino acid residues, they defined the score of a pairwise alignment of protein sequences as the sum of similarity scores of aligned pairs of residues minus a gap penalty for every gap inserted into the sequences. This objective function is appropriate if sequences are globally related, as is known from daily experience as well as from statistical theory (see Bishop and Thompson, 1986; Thorne et al., 1992).

If sequences are only locally related, various local alignment programs can be applied (Smith and Waterman, 1981; Pearson and Lipman, 1988; Altschul et al., 1990; Schuler et al., 1991; Lawrence et al., 1993; Abdellaïm, 1997; Brocchieri and Karlin, 1998).

Recently, an objective function for sequence alignment was proposed that can be applied to both locally and globally related data sets (Morgenstern et al., 1998; Morgenstern, 1999). This function is based on the comparison of whole segments of the sequences under consideration (so-called diagonals as they appear as diagonals in the respective dynamic programming matrix). See also Vingron and Argos (1991), Vingron and Pevzner (1995) and Altschul and Erickson (1986) for similar ideas.

We say that an alignment realizes a segment pair if all residues in the segment pair are matched. Therefore, we can consider alignments as consistent collections of diagonals. Here, consistency means that an alignment exists such that all diagonals are realized. A weight function \( w(D) \) is used that gives a positive weight score to every possible diagonal \( D \), and the overall score of an alignment is then defined as the sum of weights of the diagonals it realizes. The goal in the Complete Maximum Consistent Diagonals (CMCD) problem is then to find an alignment in which a maximum weight set of diagonals is matched. In the (Sparse) Maximum Consistent Diagonals (MCD) problem, the input consists of a subset of all possible diagonals.
Morgenstern et al. (1996) gave a heuristic algorithm for the CMCD problem. In a first step, they compute for all pairs of sequences the optimal segment-to-segment alignment, i.e., a consistent collection of diagonals with maximum sum of weights. Then a greedy strategy is employed. All diagonals contained in one of the pairwise alignments are sorted according to their weights and included one-by-one into the multiple alignment as long as they are consistent with the diagonals already included. An implementation of this algorithm is called DIALIGN (short for DIagonal ALIGNment). The program is available under http://bibiserv.techfak.uni-bielefeld.de/dialign.

The greedy strategy used in the DIALIGN program works reasonably efficiently in terms of computing time and memory. However, this strategy has an inherent disadvantage. Once a diagonal is included into the alignment, it is fixed and cannot be removed at a later stage of the algorithm. In Figure 1, for example, there is a motif contained in all of the three sequences. Diagonal $D_4$, however, is not consistent with this motif. $D_4$ is the diagonal with the highest weight and is contained in the optimal pairwise alignment of sequence 1 and sequence 3. In the greedy procedure, it is the first one to be included in the multiple alignment. As a result, the motif cannot be correctly aligned. The resulting multiple alignment would consist of the diagonals $D_1$, $D_4$ and $D_5$, and therefore have a score of $1.9 + 2.6 + 0.2 = 4.7$. In contrast, an alternative alignment that would ‘correctly’ align the motif could consist of diagonals $D_1$, $D_2$, $D_3$ and $D_5$, with a score of $1.9 + 1.7 + 1.5 + 0.2 = 5.3$.

In the DIALIGN program, a modification of the greedy strategy was used that sometimes helps to exclude ‘wrong’ diagonals from the alignment. Diagonals are not sorted according to their weights, but rather according to so-called overlap weights where motifs occurring in more than two sequences are preferred to motifs occurring in only two sequences (for details, see Morgenstern et al., 1996). In situations such as the one shown in Figure 1, this may lead to the correct alignment.

However, to our knowledge, there exists no efficient algorithm that solves the (C)MCD problem to optimality. The main goal of this paper is to show that the (C)MCD problem can be formulated in terms of another multiple sequence alignment problem: the Generalized Maximum Trace (GMT) problem. The original Maximum Trace (MT) problem was introduced in 1991 by Kececioglu (Kececioglu, 1991, 1993) as a model for the final alignment phase of DNA sequence assembly. In the MT problem, the letters of the $k$ input strings are viewed as vertices $V$ in a judiciously chosen subgraph of the complete $k$-partite graph $G = (V, E)$. Every edge $e \in G$ has a positive weight representing the gain of aligning the endpoints of the edge. We say that an alignment realizes an edge if it places the endpoints into the same column of the alignment array. The set of edges realized by an alignment $A$ is called the trace of $A$, denoted trace($A$), and the weight of an alignment is the sum of the weights of the edges that it realizes (see also Figure 2 for an alignment graph and Figure 5 for a trace). The goal is to compute an alignment of maximum weight. Kececioglu (1991) also showed that the MT formulation is a NP-hard problem.

Reinert et al. (1997) gave a formulation of the MT in terms of an integer linear program (ILP). Additionally, they showed that an implementation of a branch-and-cut algorithm is able to solve instances of the problem to optimality, the size of which is intractable for dynamic programming-based approaches. We will show that the ILP can be modified such that the resulting generalized formulation accounts for numerous multiple sequence alignment problems, amongst them the original MT and the (C)MCD. Solving the GMT to optimality therefore immediately yields optimal solutions to these problems.
In the next section, we will show how to generalize the MT formulation in such a way that it captures the original MT problem and the (C)MCD problem. First, we give a graph-theoretic characterization of the GMT in terms of forbidden subgraphs. Then, we use this characterization to formulate the ILP and sketch the branch-and-cut approach for solving the ILP. We then describe in detail two possible ways to generate the input diagonals for the MCD problem. The first uses DIALIGN to generate the input, the second takes diagonals from locally optimal alignments with affine gap costs. Finally, we show computational results using an adaption of the branch-and-cut program proposed in Reinert et al. (1997).

**The Generalized MT formulation**

Let \( \Sigma \) be a finite alphabet and let \( \hat{\Sigma} = \Sigma \cup \{-\} \), where ‘-’ (dash) is a symbol to represent ‘gaps’ in strings. The input of a multiple sequence alignment algorithm is a set \( S = \{S_1, S_2, \ldots, S_k\} \) of \( k \) finite strings over the alphabet \( \Sigma \). In the GMT formulation, we view the letters of the strings \( S_i = (s_{i1}, \ldots, s_{in}) \) as vertices \( V \) in a subgraph of the complete \( k \)-partite graph \( G \), in which we allow multiple edges between two nodes. We call this graph the alignment graph (see also Figures 2 and 3). We say that an alignment realizes an edge if it places the endpoints in the same column of the alignment array. The set of edges realized by an alignment \( A \) is called the trace of \( A \), denoted \( \text{trace}(A) \).

In addition, the edge set \( E \) is partitioned into certain sets \( T \subset 2^E \) and each element \( t \in T \) is an integral unit that can be realized by an alignment, which means either all edges in \( t \) are realized or none. This implies that the elements of \( T \) must themselves be a trace. We call the elements in \( T \) blocks. Each block \( t \in T \) has a weight \( w_t \) representing the gain of realizing that block.

In contrast to the MT formulation, the weight of an alignment is now the sum of the weights of all blocks \( t \in T \) that it realizes. The goal is to compute an alignment of maximum weight. The following figures show the partitions that model the original MT and the MCD problem:

- The partition into the singletons \( \{\{e\} | e \in E\} \) (see Figure 2). Here a single edge is a block that can be chosen by an alignment. The edges may stem, for example, from the superposition of matches occurring in (sub)optimal global alignments. This partition corresponds to the original MT problem.
- A partition into sets of edges that form a consecutive run of matches (diagonal) (see Figure 3). Here several edges form a block. An alignment is required either to realize all or none of the edges in such a block. The sets may stem, for example, from matches from optimal local alignments. This partition corresponds to the MCD problem.

Note that the edge set \( E \) together with its partition \( T \) can encode a variety of different alignment formulations. The solution of the GMT will always yield the multiple alignment that agrees most with the specific input defined by the partition \( T \).

**A graph-theoretic characterization of the GMT problem**

In this section, we give a graph-theoretic characterization of traces in a form that is helpful for expressing GMT as an ILP. First we introduce some notations.
A mixed graph is a tuple \( G = (V, E, A) \), where \( V \) is a set of vertices, \( E \) is a multiset of edges and \( A \) is a multiset of arcs. We denote an edge \( e \) as an unordered pair of its incident nodes, i.e. \( e = [u, w] \) and an arc \( a \) as an ordered pair of its source and sink node, i.e. \( a = (u, v) \). A path in a mixed graph is an alternating sequence \( v_1, e_1, v_2, e_2, \ldots, v_l \) of vertices and arcs or edges such that either \( e_i \in E \) or \( e_i = (v_i, v_{i+1}) \) for all \( i, 1 \leq i < l \). A path is called a mixed path if it contains at least one arc in \( A \) and one edge in \( E \). A mixed path is called a mixed cycle if the first and the last vertex on the path are the same. Since a mixed path \( G' \) is a trace or not by simply looking for a critical mixed cycle. Consider a smallest size mixed cycle \( G' \) that realizes \( \beta \). Assume first that \( \beta \neq 0 \), then there is some \( x \in \beta \) such that \( \gamma(x) = \beta \) and \( \gamma(y) = 0 \) for all \( y \in \beta \). We define the trace polytope as the convex hull of all incidence vectors of \( T \) that are feasible, i.e.

\[
P_T := \text{conv} \left\{ \chi^M \mid (\bigcup_{t \in M} t) \text{ is a trace} \right\}
\]

The surjective function \( \nu : E \cup A \to T \cup \emptyset \) with

\[
\nu(e) = \begin{cases} 
  t & \text{if } e \in T, \\
  \emptyset & \text{if } e \in A
\end{cases}
\]

maps each edge \( e \in E \) to the element of \( T \) in which \( e \) is contained and each arc \( a \in A \) to the empty set. As a shorthand, we write \( \nu(C) := \{ \nu(e) \mid e \in C \} \). It is now easy to formulate GMT as an ILP. For every block \( t \in T \), the binary variable \( x_t \in \{0, 1\} \) indicates whether \( t \) is in the solution or not. In view of Theorem 1, the GMT problem is equivalent to

\[
\text{maximize } \sum_{t \in T} w_t \cdot x_t \text{ subject to } x_t \in P_T (G)
\]

is equivalent to

\[
\text{maximize } \sum_{t \in T} w_t \cdot x_t \text{ subject to } x_t \leq | \nu(P) | - 1.
\]

Using Theorem 1, the GMT problem now reads as follows:

**Generalized Maximum Trace problem:** Given an EAG \( G = (V, E, A) \) and a partition \( T \) of \( E \) with weights \( w_t (\forall t \in T) \).

Find a set \( M \subseteq T \) of maximum weight such that \( \bigcup_{t \in M} t \) does not induce a critical mixed cycle on \( G \).
\[ \forall \text{ critical mixed cycles } P \text{ in } G \]
\[ x_t \in \{0, 1\}, \forall t \in T \]

A review of branch-and-cut algorithms. Amongst the various approaches to solve ILPs, we choose branch-and-cut techniques to solve the problem to optimality. In the branch-and-cut approach, one relaxes the given ILP by dropping the integer condition and solves the resulting linear program. If the solution \( \pi \) of the linear program is integral, and if all mixed cycle inequalities are satisfied, then we have the optimal integral solution. Otherwise, one searches for a valid inequality \( f x \leq f_0 \) that ‘cuts off’ the solution \( \pi \), i.e. \( f y \leq f_0 \) for all \( y \in P_T(G) \) and \( f x > f_0 \); \( \{ x \mid f x = f_0 \} \) is called a cutting plane. The search for a cutting plane is called the separation problem. Any cutting plane found is added to the linear program and the linear program is resolved. The generation of cutting planes is repeated until either an optimal solution is found or the search for a cutting plane fails. In the second case, a branch step follows. One generates two subproblems by setting one fractional variable \( x_t \) to 0 in the first subproblem and to 1 in the second subproblem, and solves these subproblems recursively. This gives rise to an enumeration tree of subproblems. Lower bounds from heuristics or approximation algorithms are used to limit the size of this tree. Even if the program cannot be solved to optimality, the value of a fractional solution yields a guaranteed error bound on a heuristic solution, such as for example given by the greedy approach. Moreover, the enumeration tree could be used to find other (sub)optimal solutions to the problem.

The most effective cutting planes are the ones that define facets of the trace polytope \( P_T(G) \), because these are the constraints that appear in an irredundant description of \( P_T(G) \) by linear inequalities. Since the GMT problem contains the MT problem as a special case, the NP-hardness of the latter also applies to the GMT. Hence, we cannot expect to be able to find a full description of the facial structure of \( P_T(G) \) by linear inequalities. Nevertheless, a partial description of the facial structure of \( P_T(G) \) by linear inequalities is useful for the design of a branch-and-cut algorithm. Reinert et al. (1997) described classes of facet-defining inequalities and efficient ways to separate them. One can show that all theorems from the above paper still hold in slightly modified form and that the algorithm for the GMT differs only in the definition of the ILP variables. However, due to space limitations, the modified proofs are not given here.

Computational results

We adopted two ways to generate diagonals as input for the branch-and-cut algorithm. We used DIALIGN to compute for every sequence pair the respective optimal segment-to-segment alignment as described in Morgenstern et al. (1996). In the second approach, we computed for every sequence pair (sub)optimal local alignments with affine gap costs that do not share (mis)matches. We call the procedure that produces the diagonals LOCAL. In the following, we describe the two approaches in more detail.

Diagonals computed by DIALIGN

The DIALIGN program calculates in a first step all optimal pairwise alignments according to the underlying segment-based objective function, i.e. for every possible pair of sequences, a consistent collection of diagonals with maximum sum of weights is calculated.

We use the weight function defined by Morgenstern et al. (1998). The set of all diagonals contained in these optimal pairwise segment-to-segment alignments is called \( \mathcal{M}_1 \). Next, DIALIGN employs a greedy strategy to find a consistent subset \( \mathcal{M}_2 \subset \mathcal{M}_1 \), i.e. a multiple alignment of the input sequences. This is done iteratively until no additional diagonals can be found [see Morgenstern et al. (1996) for details]. We considered the sets \( \mathcal{M}_1 \) and \( \mathcal{M}_2 \) as constructed in the first iteration step of DIALIGN as input for the GMT problem and the sum of the weights of all diagonals contained in \( \mathcal{M}_2 \) was used as a lower bound for our branch-and-cut algorithm.

Note that the DIALIGN program tries to solve the complete GMT problem, i.e. all possible diagonals with positive weight are considered, whereas, in this study, we are solving the sparse GMT: we consider only those diagonals that are contained in one of the pairwise alignments constructed by DIALIGN, i.e. the input for our program is the set \( \mathcal{M}_1 \). Nevertheless, the set \( \mathcal{M}_2 \) in the first iteration step of DIALIGN can be seen as a solution for exactly the same sparse GMT problem, so it is actually possible to compare the two methods.

Diagonals computed by LOCAL

In this approach, we first compute optimal local alignments in the sense of Smith and Waterman (1981), using affine gap costs. This naturally gives rise to a number of diagonals by cutting the alignment at the gapped positions and taking the consecutive runs of aligned residue pairs as diagonals. Then, we compute for each sequence pair the next best local alignment that shares no residue pair with the previously computed alignment. We continue this procedure until the length of the local alignments falls below a given threshold value. In this way, we obtain for each sequence pair a collection of diagonals stemming from ‘good’ local alignments. The set of these diagonals is not necessarily consistent, but the diagonals do not share common residue pairs. The set of matching residue pairs within this diagonal. Let \( P(l_D, m_D) \) be the probability that a random diagonal of the same length \( l_D \) has at least the...
m_D matches. Then we defined \( w_D \) to be \( -\log P(l_D, m_D) \). We apply a simple greedy heuristic in order to find a consistent subset of our set of diagonals and use the sum of weights of the diagonals as a lower bound for our branch-and-cut algorithm.

We coded the algorithm in C++ using the library of efficient data types and algorithms LEDA (Mehlhorn and Näher, 1995). Further on, we used the branch-and-cut framework ABACUS by Stefan Thienel (Jünger and Thienel, 1997). As input sequences for our first example, we used a subset of a dataset of McClure et al. (1994). Figure 6 shows the result of a run of our algorithm where the alignments were obtained as described in the proof of Theorem 1. The figure shows an alignment of a set of six globin sequences where the input was generated using the LOCAL procedure. All five motifs that are used by McClure et al. for evaluating the quality of alignments are perfectly aligned. Figure 6 shows that the score of an optimal solution (2067) is much higher than the score of the heuristic solution, which is only 1278. This shows how the branch-and-cut algorithm can improve considerably the solution computed by the pure greedy approach. The algorithm ran on one processor of an Sun Enterprise 10000 and needed 36 s to compute the alignment.

On the other hand, if we use the diagonals contained in the pairwise DIALIGN alignments as input, the DIALIGN heuristic yields a lower bound of 1346.99. (Note that the values in this example are not comparable to the previous example because of the different scoring functions.) It turned out that this was already the score of an optimal solution of the GMT problem. This means that our branch-and-cut algorithm can improve considerably the solution computed by the pure greedy approach. The algorithm ran on one processor of an Sun Enterprise 10000 and needed 36 s to compute the alignment.

Taking into account the good lower bound of DIALIGN, we tried the more ambitious example of 18 prion sequences taken from the SWISSPROT database. Here, the pure greedy lower bound of LOCAL was too poor, so that the branch-and-cut algorithm did not finish within a day. If we use DIALIGN, however, the lower bound is better and we can solve the problem to optimality in 144 min. Figure 7 shows the computed alignment of the 18 sequences. Lower case letters indicate positions that are not aligned in any diagonal. It turns out that our algorithm computes a much higher scoring alignment (59 862.5) than DIALIGN does (53 475). Also, we claim that no exact algorithm based on dynamic programming can compute an optimal alignment of 18 sequences, even if they are quite similar like in this example.

**Conclusion**

In this paper, we show how to generalize the MT problem in such a way that it captures several multiple sequence alignment formulations, amongst them the original MT and the MCD formulation. The GMT can be solved to optimality by a branch-and-cut algorithm which is, to our knowledge, the first algorithm that is able to do so. We tested the algorithm with two possible methods for generating input for the MCD problem and show that the solutions may improve considerably compared to the greedy approach.

Moreover, our method can also be seen as a procedure to check heuristically computed alignments for optimality. This gives valuable information for the further development of alignment programs. If we know that a heuristic method is already able to construct (near-)optimal alignments in the sense of the underlying objective function, then there is no point in improving the optimization strategy to obtain even higher scoring alignments. If alignments are still question-
Fig. 7. Optimal alignment of 18 prion protein sequences. Input was generated using DIALIGN.
able from the biological point of view, one should rather go back and try to improve the underlying objective function. On the other hand, if heuristically calculated alignments can be shown to have poor scores compared with optimal alignments, then it makes sense to improve the optimization strategy to obtain better scoring alignments. Thus, we believe that our method is not only useful for practical problems, but will also help to improve heuristic alignment strategies.

Further research is needed in order to identify new classes of facet-defining inequalities of the trace polytope and to develop separation algorithms for these classes, as well as for already known classes. This would immediately imply faster algorithms for all exact multiple sequence alignment formulations that are captured by the GMT formulation. We view as one contribution of our work the introduction of the polyhedral approach to sequence alignment. Our experience with this novel method shows that it leads to algorithms that can cope with more sequences than dynamic programming-based exact methods. In contrast to the latter, which are already thoroughly studied and hard to improve, we see a lot of potential in our method.

References


