BIOINFORMATICS ORIGINAL PAPER

Vol. 26 no. 8 2010, pages 1015–1021
doi:10.1093/bioinformatics/btq082

Sequence analysis

Advance Access publication February 25, 2010

# A *min-cut* algorithm for the consistency problem in multiple sequence alignment

Eduardo Corel[1,*], Florian Pitschi[2] and Burkhard Morgenstern[1]

[1]Georg-August-Universität, Institut für Mikrobiologie und Genetik, Goldschmidtstraße 1, 37077 Göttingen, Germany and [2]Partner Institute for Computational Biology, CAS-MPG, 320 Yue Yang Rd, 200031 Shanghai, China

Associate Editor: Limsoon Wong

**ABSTRACT**

**Motivation:** Multiple sequence alignments can be constructed on the basis of pairwise local sequence similarities. This approach is rather flexible and can combine the advantages of global and local alignment methods. The restriction to pairwise alignments as building blocks, however, can lead to misalignments since weak homologies may be missed if only pairs of sequences are compared.

**Results:** Herein, we propose a graph-theoretical approach to find local multiple sequence similarities. Starting with pairwise alignments produced by DIALIGN, we use a min-cut algorithm to find potential (partial) alignment columns that we use to construct a final multiple alignment. On real and simulated benchmark data, our approach consistently outperforms the standard version of DIALIGN where local pairwise alignments are greedily incorporated into a multiple alignment.

**Availability:** The prototype is freely available under GNU Public Licence from E.C.

**Contact:** ecorel@gwdg.de

Received on November 25, 2009; revised on February 12, 2010; accepted on February 19, 2010

## 1 INTRODUCTION

Multiple sequence alignment (MSA) is a requisite for almost all aspects of computational sequence analysis, but it is a notoriously difficult task, see Edgar and Batzoglou (2006), Morrison (2006) or Kemena and Notredame (2009) for recent reviews. Traditionally, alignment methods have been characterized as either *local* (Altschul *et al.*, 1997; Bailey and Elkan, 1994; Smith and Waterman, 1981) or *global* (Edgar, 2004; Lassmann and Sonnhammer, 2005; Needleman and Wunsch, 1970; Thompson *et al.*, 1994). The idea of including local alignment information into global alignment tools was first implemented in *DIALIGN* (Morgenstern *et al.*, 1996) and is also used in more recent tools such as *T-Coffee* (Notredame *et al.*, 2000) or MAFFT (Katoh *et al.*, 2005).

Most approaches for MSA are based on well-defined objective functions and try to find optimal or near-optimal MSAs according to these functions. Traditional objective functions for pairwise alignment are defined by summing up individual substitution scores and gap penalties (Needleman and Wunsch, 1970); this can be extended to multiple alignment using a (weighted) sum-of-pairs (SPs) or tree-alignment approach. These objective functions have a probabilistic interpretation and can be formulated in a hidden Markov model (HMM) framework (Durbin *et al.*, 1998; Eddy, 1995). More recently, segment-based objective functions (Morgenstern *et al.*, 1996) and objective functions based on *probabilistic consistency* (Do *et al.*, 2005) have been proposed. A promising new approach is the use of *conditional random fields* for MSA as proposed by Do *et al.* (2006).

Since the optimal multiple alignment problem is NP-hard under any reasonable objective function, virtually all MSA programs are based on heuristic optimization algorithms. Most of these heuristics work by integrating rather simple *partial* alignments into a final multiple alignment. For *global* alignment, paradigmatically represented by *ClustalW* (Thompson *et al.*, 1994), this is usually done by *progressive* alignment. Here, single sequences and *profiles* of related sequences are aligned until all input sequences are included in a multiple alignment. In contrast, in the segment-based approach implemented in *DIALIGN*, local similarities are integrated sequentially under the constraints imposed by a certain *consistency* criterion. More recently, the progressive approach has also been applied to the segment-based alignment though (Subramanian *et al.*, 2008).

Consistency is an order-theoretic condition that ensures the compatibility of the local similarities with the linear structure of a global alignment (Abdeddaïm and Morgenstern, 2001; Morgenstern *et al.*, 1996). Shortly spoken, a set of (partial) alignments $A_1,\ldots,A_k$ is called consistent, if an alignment $A$ of the input sequences exists such that each of the alignments $A_i$ is represented in $A$. In the DIALIGN program, the raw material from which a multiple alignment is built is a set of pairwise gap-free local alignments. Such partial alignments are called *fragment alignments* or, shorter, *fragments*. Thus, a fragment is represented as a pair of segments of the same length from two of the input sequences. Each fragment is given a *weight score* based on the probability of its random occurrence, and the optimization task is to find a consistent set of fragments with maximum total weight.

For pairwise alignment, a consistent set of fragments is a *chain* of fragments, i.e. a set of fragments where for any two fragments one of them is strictly to the left of the other one. An optimal alignment in this approach is, therefore, a chain of fragments with maximum total weight and can be found by dynamic programming, either using standard fragment-chaining algorithms (Gusfield, 1997), or by a more space-efficient algorithm that is implemented in DIALIGN (Morgenstern, 2000, 2002). For multiple alignment, fragments from the respective optimal pairwise alignments are greedily included or discarded according to their consistency with each other.

*To whom correspondence should be addressed.

A general problem with the algorithmic approach adopted in DIALIGN is that weakly conserved homologies can easily be missed since they may not appear statistically significant in the pairwise alignments that are carried out as the first step of the algorithm. Moreover, even if a weak local similarity is detected and represented by fragments in some of the optimal pairwise alignments, these fragments may be outweighed by spurious random similarities in other pairwise alignments. Therefore, alternative optimization algorithms have been proposed for the segment-based alignment approach, e.g. *integer linear programming* (Kececioglu *et al.*, 2000; Lenhof *et al.*, 1999). However, these approaches are computationally expensive. Alternatively, it is possible to search for *multiple* local alignments in a first step, and to use these alignments as *anchor points* (Morgenstern *et al.*, 2005, 2006) for a subsequent global alignment procedure. Such an approach has been recently proposed by Pitschi (2008). Complementary to this *anchored-alignment approach*, it is possible for the user to *exclude* certain regions of the input sequences from being aligned to each other (Dress *et al.*, 2008).

## 2 APPROACH

In this article, we study a different approach to the segment-based multiple alignment. Like in the original approach, we start with constructing all optimal pairwise alignments in the sense of DIALIGN, i.e. by finding optimal chains of fragments for all sequence pairs. However, instead of inserting these fragments directly into a multiple alignment, we first search for local similarities shared by more than two sequences. Our approach is based on the following observation: typically, local homologies involving more than two sequences correspond to groups of overlapping fragments (Fig. 1). In contrast, spurious random similarities are rarely part of such fragment groups.

In our approach, we therefore search for groups of positions in the sequences that are connected by many DIALIGN fragments. In addition, we require each of these groups to contain at most one position from each input sequence. Such groups are potential (partial) columns of a biologically meaningful multiple alignment. Consequently, the core of our new approach is to extract partial alignment columns from the fragments produced by DIALIGN. If the positions of the sequences are seen as vertices in a graph and we have an edge between any two positions that are aligned by one of the selected fragments, the core problem of our approach is to extract highly connected subgraphs that contain at most one vertex from each of the input sequences. We use a *min-cut, max-flow* algorithm to identify such subgraphs.

The partial alignment columns that we find in this way need not be consistent with each other. In a second step, we therefore use a novel algorithm proposed by Pitschi (2008) to obtain a *directed acyclic graph* (DAG) from our potential alignment columns. This way, we obtain a consistent set of (partial) alignment columns. To align those parts of the sequences that are not yet aligned by the selected columns, we use these columns to define *anchor points* for DIALIGN in order to find further sequence similarities that are consistent with the partial alignment columns that we already obtained with our graph-theoretical approach.

Test runs on BAliBASE and on simulated local sequence homologies show that our approach is a considerable improvement compared to previous versions of DIALIGN. Interestingly, the

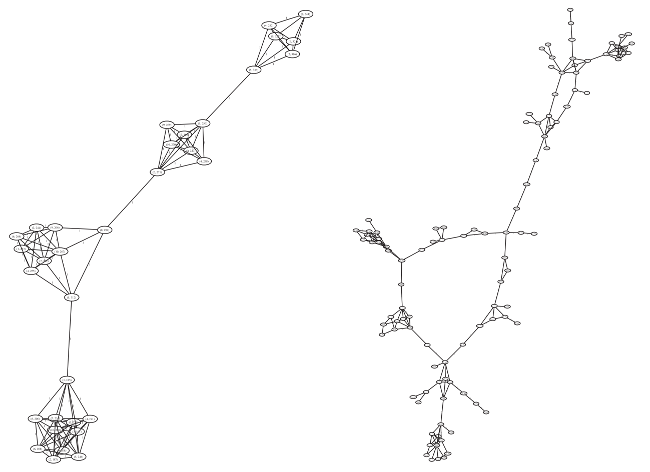

**Fig. 1.** Two connected components of the incidence graphs for datasets BB12001 (left) and BB50005 (right) from BAliBASE 3. The modular structure of these graphs is obvious. While the entire graph is not connected and rather sparse, it is composed of dense subgraphs. Typically, these subgraphs involve at most one site from each of the input sequences, they therefore represent partial alignment columns. The goal of the algorithm described in Section 3.1 is to remove a minimal set of edges from $G_{\mathcal{F}}$ such that each of the resulting connected components contains at most one site from every sequence.

*numerical* scores of our alignments are generally *not* better than with the standard version of DIALIGN. This indicates, that it is not the greedy optimization algorithm but rather the objective function used in DIALIGN that is to be blamed for the known limitations of this program on weakly but globally related sequence sets.

## 3 ALGORITHM

We consider a collection $S$ of $n$ sequences over a finite alphabet where $\ell(i)$ is the length of the $i$-th sequence. The *site* $(i,p)$ is the $p$-th position of the $i$-th sequence. The *site space*

$$\mathcal{S}=\{(i,p)|1\leq i\leq n, 1\leq p\leq \ell(i)\}$$

is endowed with a natural partial ordering '$\preceq$' such that $(i,p)\preceq(i',p')$ holds if and only if $i=i'$ and $p\leq p'$. By $\mathcal{S}_i$ we denote the sites in the $i$-th sequence, i.e. the set $\{(i,p)|1\leq p\leq \ell(i)\}$. In the following, we freely identify the $i$-th sequence with the set $\mathcal{S}_i$. An *alignment* of $S$ is an equivalence relation $A$ of $\mathcal{S}$ satisfying the following consistency criterion: the preorder[1] $\preceq_A=(\preceq\cup A)_t$ restricted to any individual sequence $s\in S$ coincides with $\preceq$. Here, $R_t$ denotes the *transitive closure* of a relation $R$. The equivalence classes of $A$ correspond to columns of aligned positions of the sequences, and for any two sites $x,x'\in\mathcal{S}$, we have $x\preceq_A x'$ if and only if $x$ is to the left or in the same column as $x'$ in $A$.[2]

We call a subset $C\subset\mathcal{S}$ *ambiguous*, if there is a sequence $\mathcal{S}_i$ such that the intersection $C\cap\mathcal{S}_i$ contains at least two elements $(i,p)$ and $(i,p')$. In this case, we also call the sites $(i,p)$ and $(i,p')$ *ambiguous* with respect to $C$. A non-ambiguous subset $C\subset\mathcal{S}$ is

[1]By definition, a preorder is a transitive and reflexive relation on a set $X$.
[2]Note that the term *consistency* has been used in a different sense by various authors, such as Gotoh (1990), Vingron and Argos (1991), Notredame *et al.* (2000) and ourselves. This has led to some confusion in the literature.

called a *partial alignment column*. Moreover, we call an equivalence relation $E$ on $\mathcal{S}$ *ambiguous*, if it contains an ambiguous equivalence class. A non-ambiguous equivalence relation consists of partial alignment columns only. Consistency of an equivalence relation obviously implies non-ambiguity, but the converse is in general not true.

In our approach to multiple alignment, we start with a set $\mathcal{F}=\{f_1,\dots,f_n\}$ of fragments. In the present study, $\mathcal{F}$ is the set of fragments contained in the respective pairwise alignments produced by DIALIGN, i.e. the union of all optimal chains of fragments from all pairs of input sequences. However, for the algorithm described below, the composition of $\mathcal{F}$ is not essential, and $\mathcal{F}$ could be any set of fragments or other (partial) alignments. Thus, our first goal is to extract a set of potential (partial) alignment columns $C_i$ from these fragments. Formally, we consider the equivalence relation $E$ induced by $\mathcal{F}$, and we are looking for a non-ambiguous equivalence relation $E'$ that is *contained* in $E$. In a graph-theoretical setting, we consider sites as vertices in a graph and pair of sites aligned by our fragments from $\mathcal{F}$ as edges. According to our above definition, we call this graph *non-ambiguous*, if each of its connected components contains at most one vertex from each of the input sequences. Our goal is then to remove a minimal number of edges from $G_{\mathcal{F}}$ to make the resulting graph non-ambiguous.

## 3.1 The incidence graph of a set of fragments

For a set $\mathcal{F}$ of fragments as above, the *incidence graph* $G_{\mathcal{F}}$ of $\mathcal{F}$ is the undirected graph $(\mathcal{S},E_{\mathcal{F}})$ where $\mathcal{S}$ is the set of sites of our input sequences and a pair of sites forms an edge $\{u,v\}\in E_{\mathcal{F}}$ if there exists a fragment $f\in\mathcal{F}$ such that $u$ is aligned to $v$ by $f$. The equivalence classes of the equivalence relation induced by the fragment set $\mathcal{F}$ are the *connected components* of $G_{\mathcal{F}}$.

As a rule, one observes that these components are extremely modular. While the graph $G_{\mathcal{F}}$ as a whole is sparse, it usually consists of highly connected clusters of vertices, i.e. by *dense* subgraphs. These subgraphs are typically linked to each other by only a small number of edges (Fig. 1). Moreover, one can observe that these dense subgraphs are often non-ambiguous, i.e. each cluster contains at most one vertex per sequence. Thus, the well-connected clusters of vertices in the graph $G_{\mathcal{F}}$ are potential (partial) alignment columns that we would like to include into a final MSA.

Our goal is therefore to turn the graph $G_{\mathcal{F}}$ into a non-ambiguous graph by removing some of its edges. This problem is equivalent to clustering a coloured undirected graph, such that no cluster has two vertices of the same colour. Whenever we detect an ambiguous connected component $C$ in our graph, i.e. a connected component containing two vertices $u$ and $v$ from the same sequence, we partition $C$ into two subsets $C_1$ and $C_2$ by removing a minimal number of edges such that we have $u\in C_1$ and $v\in C_2$. To this end, we consider the subgraph $G$ induced by $C$ as a *flow network* and we use the *max-flow min-cut* theorem, that we recall now. We repeat this procedure until no ambiguous connected components are left.

## 3.2 Flow networks

A flow network $N$ is a directed graph $G=(V,E)$ with two distinguished vertices $s$ and $t$—the 'source' and the 'sink'—where each edge $(u,v)$ is assigned a non-negative real number, its 'capacity' $c(u,v)$. Flow networks are used to model the situation where some material, e.g. some fluid or current, can 'flow' from $s$ to $t$ along the edges of the graph $G$. The capacity of an edge $(u,v)$ can be thought of as the maximum possible flow that goes directly from $u$ to $v$. A *cut* of a flow network is a partition $V=S\cup T$ with $s\in S$ and $t\in T$. The capacity $c(S,T)$ of a cut $(S,T)$ is the sum of the capacities of all edges going from $S$ to $T$.

A *flow* in a flow network $N$ is an assignment of real numbers $f(u,v)$ to the edges $(u,v)$ of $N$ that can be interpreted as the actual movement of material. The flow through any edge is bounded by its capacity and the total incoming flow for any vertex must equal the total outgoing flow, except for the source and the sink. The *value* $|f|$ of the flow $f$ is defined as the total flow leaving the source, which can be shown to equal the total flow entering the sink. Note that for any flow $f$ in $N$ and any cut $(S,T)$ of $N$, we have $c(S,T)\geq|f|$. The *maximum-flow problem* is the problem of finding a flow $f$ with maximum value $|f|$. According to the *max-flow min-cut theorem*, a flow $f$ in $N$ is maximal if and only if there is a cut $(S^*,T^*)$ of the network with $c(S^*,T^*)=|f|$. There are several known polynomial-time algorithms to compute a maximum flow on a flow network, for example, Ford and Fulkerson (1956) or Edmonds and Karp (1972) (see also Cormen *et al.*, 2001, pp. 651–664 ).

## 3.3 Step one: resolving ambiguities to construct partial alignment columns

Let $C$ be a connected component in $G_{\mathcal{F}}$ containing two nodes $x\neq y$ from the same sequence. We turn $C$ into a flow network $N$ by defining two directed edges $(u,v)$ and $(v,u)$ for any two nodes $u$ and $v$ that are connected by an undirected edge in $G_{\mathcal{F}}$, i.e. are aligned by one of the fragments in $\mathcal{F}$, and we define the capacity of every edge to be 1. The two 'ambiguous' nodes $x$ and $y$ are defined as the source and the sink, respectively, of our flow network.

We then use the Edmonds–Karp algorithm to compute a maximum flow $f_{\max}$ and we consider the so-called *residual network* $N'$, which is obtained from $N$ by subtracting the maximum flow $f_{\max}$ from the capacities of the edges, i.e. the capacity $c'(u,v)$ in $N'$ is defined as $c(u,v)-f_{\max}(u,v)$. Edges $(u,v)$ with capacity $c'(u,v)=0$ are removed. Note that the capacity of each cut in $N$ is at least $|f_{\max}|$ and, according to the Edmonds–Karp theorem, there is a cut $(S,T)$ with capacity $c(S,T)=|f_{\max}|$. Thus, in the residual network $N'$ all edges between $S$ and $T$ are removed and our ambiguous subgraph $G$ is split into two connected subgraphs by removing a minimal set of edges from $E_{\mathcal{F}}$. We apply this algorithm successively to split ambiguous subgraphs of $G_{\mathcal{F}}$ by removing minimal sets of edges from $E_{\mathcal{F}}$ until $G_{\mathcal{F}}$ is non-ambiguous.

One problem in our approach is that the connected components of our incidence graph $G_{\mathcal{F}}$ can be very large. We therefore define a threshold $k$ and apply the above graph algorithm only to those nodes that have degree $\geq k$. We start with $k=\max\{\deg v|v\in\mathcal{S}\}$ and $k$ is successively lowered until all nodes of $G_{\mathcal{F}}$ have been considered. A high-level description of our algorithm is as follows:

There are two subtleties left. (i) In general, a connected component $C$ of our incidence graph $G_{\mathcal{F}}$ may contain multiple ambiguities: there may be several input sequences each of which containing two or more nodes from $C$. In this case, we need to decide, in which order these conflicts are to be resolved. (ii) There may be more than one minimal cuts of an ambiguous subgraph $C$. In our implementation, we adopted the following procedure: (i) In case there are several sequences involving ambiguous nodes from $C$, choose the sequence $\mathcal{S}_i$ involving the greatest number

**Algorithm 1** Ambiguity resolving algorithm

```
Input: $G_{\mathcal{F}} = (\mathcal{S}, E_{\mathcal{F}})$
  $k \leftarrow \max\{\deg v \mid v \in \mathcal{S}\}$
  $E \leftarrow E_{\mathcal{F}}$
  while $k \geq 0$ do
    $E_k \leftarrow \{(u,v) \in E \mid \min(\deg u, \deg v) \geqslant k\}$
    Compute connected components of $(\mathcal{S}, E_k)$
    while there is an ambiguous connected component $C$ of $(\mathcal{S}, E_k)$,
    i.e. with vertices $x, y$ from same sequence $\mathcal{S}_i$ do
      Compute connected components of $(\mathcal{S}, E)$
      while there is an ambiguous connected component $C$ of
      $(\mathcal{S}, E)$, i.e. with vertices $x, y$ from same sequence $\mathcal{S}_i$ do
        1. Define flow network on $C$ with $x$ and $y$ as source and sink
        2. Apply Edmonds-Karp to find minimal cut $(C_1, C_2)$ of $C$
        3. Remove edges between $C_1$ and $C_2$ from $E$
      end while
    end while
    $k \leftarrow k - 1$
  end while
  return $(\mathcal{S}, E)$ non-ambiguous subgraph of $G_{\mathcal{F}}$
```

of ambiguous nodes. (ii) For the selected sequence $\mathcal{S}_i$, choose a pair of vertices $(x,y) \in \mathcal{S}_i$ such that $x$ has the lowest possible degree and $y$ the highest possible degree, respectively. When computing the min-cut, orient the flow by choosing vertex $x$ as the source and vertex $y$ as the sink. (iii) If there are multiple minimal cuts of $C$, define $C_1$ as the connected component of the residual network $N'$ containing $x$ and $C_2$ as its complement $C \setminus C_1$.

## 3.4 Step two: finding consistent sets of partial alignment columns

The above described procedure returns a set $\mathcal{C}$ of partial alignment columns. In general, however, these columns will not be *consistent* with each other. Thus, we may have to further reduce the set $\mathcal{C}$ in some way in order to obtain a consistent set of partial alignment columns that we can use as a core of a multiple alignment of our input sequences. To this end, we introduce another combinatorial structure that will be used to identify the positions that are likely responsible for the inconsistency. The *succession graph* of a set $\mathcal{C}$ of partial columns is an edge-weighted directed graph $SG(\mathcal{C}) = (\mathcal{C}, E, w)$ where we have an edge $e = (C, C')$ if and only if there is a sequence $\mathcal{S}_i$ such that there are sites $(i,p) \in C$ and $(i,p') \in C'$ with $p < p'$ and there is no partial column $C''$ such that $(i,p'') \in C''$ and $p < p'' < p'$. That is, we have an edge from $C$ to $C'$ if there is at least one sequence $\mathcal{S}_i$ where $C$ occurs to the left of $C'$, and no other partial column occurs in $\mathcal{S}_i$ between the occurrences of $C$ and $C'$. The *weight* of the edge $(C, C'')$ is then defined as the number of sequences $\mathcal{S}_i$ with this property. For convenience, we also add an initial vertex $v_{\text{start}}$ and a terminal one $v_{\text{end}}$.

It is easy to see that a set $\mathcal{C}$ of partial alignment columns is consistent if and only if the graph $SG(\mathcal{C})$ is a DAG. To resolve potential inconsistencies in our set $\mathcal{C}$ of partial alignment columns, we use an algorithm that was originally introduced by Pitschi (2008). Finding a consistent set of partial alignment columns amounts to finding a set of partial columns whose succession graph is a DAG. To turn our (possibly) inconsistent set of classes $\mathcal{C}$ into a consistent one, we proceed in two steps: (i) successively remove the lowest weighted edges from the graph $SG(\mathcal{C})$ until all cycles have disappeared and (ii) delete sites from the partial columns corresponding to the suppressed edges to ensure that the succession graph of this new set of partial columns is itself a DAG. The details of this algorithm are to be found in Pitschi (2008), and will also be published in a forthcoming paper (F. Pitschi *et al.,* submitted for publication).

## 3.5 Constructing a final multiple alignment from partial alignment columns

With the algorithms outlined in Sections 3.3 and 3.4, we obtain a consistent set of partial alignment columns, that is an alignment $A$ in the sense of our above set-theoretical alignment definition. In general, however, it will be possible to further extend this alignment. In the sense of our set-theoretical alignment definition, there may be an alignment $A'$ that is a proper superset of $A$. To find a suitable extension $A'$ of $A$, we run DIALIGN by using our partial columns as *anchor points* (Morgenstern *et al.*, 2006). Here, anchor points are considered as ungapped local pairwise alignments, i.e. as *fragments* in the sense of DIALIGN. In general, the user can specify a set of potential anchor points with user-defined weights from which DIALIGN selects a consistent subset in a greedy way based on their weights. In our case, we can use any set of anchor points representing our selected partial alignment columns, without worrying about their weights, since our set of anchor points is consistent anyway.

To study the influence of the two steps of our approach described in Sections 3.3 and 3.4, respectively, we tested a second version of our method where we only performed the first step as described in Section 3.3 to calculate non-ambiguous connected sets of sites. Here, we use these partial alignment columns *directly* to define anchors for DIALIGN, without first selecting a consistent set of partial columns. So in this case, DIALIGN may have to reject some of the anchors in order to ensure that a consistent set of anchors is used. Therefore, a *weight* needs to be defined for each of the anchor points, and the proposed anchor points are selected greedily according to these weights. Thus, the resulting alignment depends on how the weights of the anchor points are exactly defined.

To obtain a set of anchor points from a set $\mathcal{C}$ of partial alignment columns, we consider all maximal pairs of segments $(i,p), \ldots, (i,p+k)$ and $(i',p'), \ldots, (i',p'+k)$ such that every pair of sites $(i,p+l)$ and $(i',p'+l), 1 \leq l \leq k$, belongs to some partial alignment column in $\mathcal{C}$. Each of these segment pairs (fragments) defines an anchor point. We defined the weights of these anchor points in two different ways, namely (i) by their length and (ii) by using the fragment-weighting function that is used in DIALIGN (Morgenstern, 1999).

## 3.6 Time complexity

Since the time complexity of the Edmonds–Karp algorithm depends quadratically on the number of nodes of the input graph, the run time of our method strongly depends on the size of the connected components of our incident graphs. In the worst case, there is a single connected component comprising the entire site space $\mathcal{S}$ and each node $(i,p)$ is connected to each sequence $\mathcal{S}_j \neq \mathcal{S}_i$ by some edge. Thus, a single connected component has up to $n \cdot \ell$ nodes and $n^2 \cdot \ell$ edges where $n$ is the number of sequences and $\ell$ their maximum length. The time complexity of the Edmonds–Karp algorithm is $O(|V| \cdot |E|^2)$, so the worst-case complexity of our algorithm to find a minimal cut for a *single* ambiguous connected component is $O(n^5 \cdot \ell^3)$. In the worst case, each run of Edmonds-Karp splits off a single node, so the algorithm is run $n \cdot \ell$ times. The worst-case time complexity of our full algorithm is, therefore, $O(n^6 \cdot \ell^4)$.

For realistic datasets, the connected components of our incidence graphs are, fortunately, much smaller than in the theoretical worst case, so we could run our method on most datasets in the benchmark databases in reasonable time. An example is reference set RV12 of BAliBASE. RV12 consists of 88 sequence families with an average of 10 sequences per sequence family. The average incidence graph $G_{\mathcal{F}}$ in RV12 consists of 2877 nodes 10 952 edges and 223 connected components and, on average, the Edmonds–Karp algorithm is run 649 times, thereby removing 899 edges. On RV12, the mean CPU time per sequence family is 48 s on an Opteron machine with 2.4 GHz.

As will be discussed in Section 4.1, we had to terminate the program runs on some large sequence families from BAliBASE. To obtain results with our method in reasonable time, we applied a threshold $T$ and removed all fragments $f$ with weight scores $w(f) < T$ from $\mathcal{F}$. An extreme case was sequence family BB30003 from BAliBASE. This sequence set comprises 142 sequences, and the incident graph consists of one single connected component with around $1.5 \times 10^6$ edges. We aborted the program run on this dataset after 20 h without results. With a threshold of $T = 4$, the graph still consists of more than $8.3 \times 10^5$ edges, and we obtained a multiple alignment of these 142 sequences after 13 h.

## 4 TEST RESULTS

To evaluate the performance of our method, we used three benchmark databases for multiple alignment. For global protein alignment, we used the well-known database *BAliBASE 3* developed by Thompson *et al.* (2005); for local alignment, we used the databases *IRMBASE* (for protein alignment) and *DIRMBASE* (for DNA alignment) developed by Subramanian *et al.* (2005, 2008). All reference sequence sets from these three databases contain so-called *core-blocks* for which a correct alignment is known.

The performance of alignment programs can be measured in two different ways: the *SPs* score measures the proportion of pairs of sites in the core blocks of the reference alignment that are correctly aligned by the method under evaluation. The *total column (TC)* score measures the proportion of alignment columns from the core blocks that are correctly aligned. The total column score is a more stringent measure, and it can be applied in a meaningful way only to those benchmark sequences where the core blocks involve *all* of the input sequences. We used the program `aln_compare` (Notredame *et al.*, 2000) to calculate these scores. Note that both, SP and TC scores, measure the *sensitivity* of alignment methods. In BAliBASE, we only used the non-truncated *long* versions of the reference sequence sets.

We evaluated two versions of our software. In a first set of test runs, we only used our min-cut algorithm to resolve *ambiguity* conflicts in the fragment set $\mathcal{F}$ and constructed sets of *partial alignment columns* as explained in Section 3.3. Note that these sets of partial alignment columns are not necessarily *consistent*. Anchor points were extracted from these data as outlined in Section 3.5 and given to DIALIGN. In a second set of test runs, the set of partial alignment columns was further processed as outlined in Section 3.4 to obtain a consistent set of partial alignment columns that were directly given to DIALIGN as anchor points.

### 4.1 Global alignment benchmark: BAliBASE 3

Tables 1 and 2 summarize the performance of our method on BAliBASE 3. On all six reference sets of BAliBASE, both versions of our approach consistently achieve a considerable improvement over the standard version of DIALIGN. This is true for both, SP and TC scores. For the TC score, however, the improvement is most substantial. It also becomes clear from Table 1 that the second part of our method achieves a further improvement. This holds, again, for all six reference sets and for both, SP and TC measures of alignment quality. Thus, resolving consistency conflicts before partial alignment columns are used as anchor points for DIALIGN gives consistently better results than using the partial alignment columns directly, after resolving the ambiguity conflicts as in Section 3.3.

It must be noted that for some of the sequence sets from BAliBASE 3, the incidence graphs proved too large for the current implementation of our *max-flow min-cut* algorithm. In particular, this was the case for some data families in reference set RV30; here, the incidence graphs consisted of up to 1.5 million edges. For these reasons, we could apply the algorithm described in Section 3.3 only to 359 out of 386 sequence families in BAliBASE. To align the remaining sequences, we filtered their incidence graphs by considering only those fragments that have a *DIALIGN* weight greater than a certain threshold $T$. At a value $T = 4$, all datasets could be processed, whereas for the lower values, some could not. Figure 2 shows the influence of this threshold on the quality of the produced alignments on BAliBASE.

Finally, we studied how the *numerical* scores of our min-cut alignments compare to the ones of the standard DIALIGN alignments on BAliBASE. In the DIALIGN approach, the score of an alignment is defined as the sum of the *weights* of the fragments it consists of. The weight of a fragment is defined as the negative logarithm of the probability of finding a fragment of the same length with at least the same sum of amino acid substitution scores in random sequences of the same length as the input sequences (Morgenstern, 1999). The results of this comparison are shown in Table 3.

**Table 1.** Performance of our min-cut method compared with other MSA methods on reference sets RV11 to RV50 in BAliBASE 3 based on an *SP* evaluation scheme

| | RV11 | RV12 | RV20 | RV30 | RV40 | RV50 |
|---|---|---|---|---|---|---|
| min-cut, cons. PAC | 53.14 | 88.53 | 90.51 | 78.56 | 87.67 | 86.45 |
| min-cut, all PAC | 50.61 | 88.36 | 90.07 | 78.19 | 87.21 | 85.58 |
| DIALIGN-TX | 51.52 | 89.18 | 87.88 | 76.18 | 83.64 | 82.28 |
| DIALIGN 2.2 | 50.73 | 86.66 | 86.92 | 74.05 | 83.31 | 80.69 |
| CLUSTALW 2.0 | 49.27 | 86.89 | 86.23 | 70.71 | 79.65 | 70.56 |
| MAFFT 6.717b | 66.19 | 93.36 | 92.72 | 87.08 | 92.19 | 90.25 |
| MUSCLE 3.7 | 57.16 | 91.54 | 88.91 | 81.45 | 86.49 | 83.52 |
| PROBCONS 1.12 | 66.97 | 94.12 | 91.68 | 84.53 | 90.34 | 89.41 |
| T-COFFEE 7.81 | 66.77 | 94.08 | 91.62 | 83.81 | 89.96 | 89.43 |

We used the (possibly non-consistent) partial alignment columns produced by our min-cut algorithm (all PAC) as discussed in Section 3.3. The resulting anchor points are sorted according to their length. As an alternative, we selected *consistent* partial alignment columns before using them as anchor points (cons. PAC) as outlined in Section 3.4.

**Table 2.** Performance on BAliBASE 3 based on the *TC* score

| | RV11 | RV12 | RV20 | RV30 | RV40 | RV50 |
|---|---|---|---|---|---|---|
| min-cut, cons. PAC | 30.51 | 73.99 | 35.21 | 41.01 | 50.17 | 50.90 |
| min-cut, all PAC | 28.82 | 74.07 | 34.07 | 40.05 | 47.54 | 49.27 |
| DIALIGN-TX | 26.81 | 75.69 | 30.78 | 38.90 | 45.17 | 47.05 |
| DIALIGN 2.2 | 26.84 | 70.03 | 29.71 | 31.62 | 44.53 | 42.94 |
| CLUSTALW 2.0 | 24.00 | 72.32 | 20.44 | 26.87 | 40.04 | 34.21 |
| MAFFT 6.717b | 44.13 | 83.83 | 45.46 | 58.90 | 60.56 | 59.52 |
| MUSCLE 3.7 | 32.06 | 80.90 | 35.30 | 41.19 | 45.32 | 46.39 |
| PROBCONS 1.12 | 41.96 | 86.05 | 41.15 | 54.73 | 53.61 | 57.89 |
| T-COFFEE 7.81 | 42.65 | 85.71 | 39.21 | 49.99 | 56.30 | 59.11 |

Notation as in Table 1.

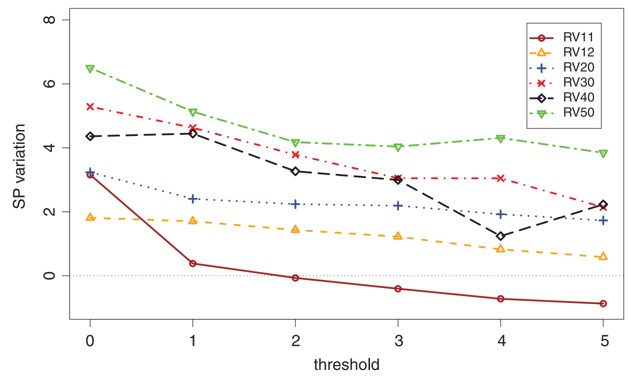

**Fig. 2.** Variation of the SP score improvement with consistent columns on BAliBASE 3 according to the threshold for selecting *DIALIGN* fragments.

**Table 3.** Numerical alignment scores in the sense of the *DIALIGN* objective function on BAliBASE for DIALIGN 2.2 and our *min-cut* approach using *consistent* partial alignment columns as anchor points for DIALIGN

| | RV11 | RV12 | RV20 | RV30 | RV40 | RV50 |
|---|---|---|---|---|---|---|
| DIALIGN 2.2 | 267 | 3703 | 113 867 | 70 220 | 27 680 | 28 291 |
| min-cut | 255 | 3758 | 114 174 | 68 555 | 26 099 | 27 855 |

## 4.2 Local alignment benchmarks: IRMBASE2 and DIRMBASE1

Since BAliBASE and other standard alignment benchmark data consist almost exclusively of sequence families that are *globally* related, i.e. with similarity extending over the entire length of the sequences, we additionally used benchmark databases for *local* protein and DNA alignment, namely IRMBASE 2 and DIRMBASE 1. Following an approach first proposed by Lassmann and Sonnhammer (2002), these databases consist of simulated protein and DNA sequence families, respectively. Locally conserved sequence motifs created using the *ROSE* software program (Stoye *et al.*, 1998) are inserted into non-related random sequences.

IRMBASE and DIRMBASE consist of four reference sets each with randomly implanted ROSE motives. Unlike the core blocks in BAliBASE, these motifs do not necessarily span all sequences in a reference sequence family. Thus, the TC score cannot be defined in a meaningful way and is, therefore, not considered in this study. Table 4 contains the results of our test runs. On the simulated local protein homologies in IRMBASE, our results were comparable to the results of DIALIGN 2.2 and DIALIGN-TX. On the locally related DNA sequences in DIRMBASE our min-cut approach performed worse though. On all locally related benchmark data, however, our method outperformed the global aligners, with the remarkable exception of *MAFFT* (Katoh *et al.*, 2005), which consistently performed well on global and local sequence data.

## 5 DISCUSSION

In this article, we introduced a new way of composing MSAs from local pairwise alignments. We first use DIALIGN to construct all optimal pairwise alignments in the sense of Morgenstern *et al.* (1996), i.e. for each sequence pair, we search for a chain of fragments with total maximum weight. Unlike DIALIGN, however, we do not include these fragments directly into a multiple alignment. Instead, we use a graph-theoretical approach to obtain potential (partial) alignment columns based on the DIALIGN fragments. To align

**Table 4.** Results on the local benchmark databases DIRMBASE 1 (D1–D4) and IRMBASE 2 (I1–I4)

| Dataset | I1 | I2 | I3 | I4 | D1 | D2 | D3 | D4 |
|---|---|---|---|---|---|---|---|---|
| min-cut, cons. PAC | 90.7 | 92.8 | 92.9 | 92.5 | 76.4 | 75.3 | 78.5 | 81.7 |
| min-cut, all PAC | 91.8 | 89.7 | 90.8 | 93.1 | 75.3 | 75.2 | 79.8 | 81.1 |
| DIALIGN-TX 1.0.2 | 89.4 | 94.9 | 93.8 | 93.6 | 94.4 | 92.9 | 95.4 | 95.7 |
| DIALIGN 2.2 | 90.4 | 93.4 | 91.8 | 93.0 | 92.6 | 91.1 | 94.6 | 94.1 |
| CLUSTALW 2.0 | 9.3 | 12.4 | 19.6 | 29.1 | 10.7 | 9.8 | 15.6 | 22.5 |
| MAFFT 6.717b | 87.7 | 92.0 | 89.9 | 88.3 | 92.5 | 83.7 | 87.3 | 86.5 |
| MUSCLE 3.7 | 30.4 | 34.5 | 54.0 | 57.8 | 47.3 | 53.2 | 56.0 | 67.7 |
| PROBCONS 1.12 | 78.8 | 85.7 | 87.1 | 87.7 | 29.9 | 31.3 | 41.5 | 52.9 |
| T-COFFEE 7.81 | 82.1 | 89.4 | 89.6 | 91.3 | 8.6 | 8.8 | 17.7 | 32.0 |

These databases consist of simulated sequences with local homologies in otherwise unrelated sequences. Notation as in Table 1.

the remainder of the sequences, we use these partial columns as anchor points in DIALIGN. Thus, the major difference between our approach and the original version of DIALIGN is the fact that we use local *multiple* sequence similarities rather than *pairwise* similarities as a basis for MSA.

The restriction to local *pairwise* similarities as building blocks for multiple alignment is a major drawback of DIALIGN. DIALIGN is one of the best methods for *local* multiple alignment. It also produces good global alignments if sequences are related over their entire length. For sequences with *weak* global sequence similarity, however, DIALIGN is often outperformed by global alignment methods. With the new approach that we proposed, we focus on local similarities that span more than two sequences; this seems to be a promising way to overcome the current limitations of DIALIGN.

Approaches to multiple alignment have two basic components: an *objective function* assigning quality scores to possible alignments and an *optimization procedure* for finding high-scoring alignments in the sense of the chosen objective function. Thus, the failure of an alignment program to produce reasonable alignments can have two reasons: the objective function may assign optimal scores to biologically wrong alignments, or the optimization algorithm may fail to find a (near-)optimal alignment. In Table 3, we compared the numerical scores of the improved alignments calculated with our new method to the scores of the standard DIALIGN alignments. In four out of six reference sets in BAliBASE, the DIALIGN alignments have on average *higher* numerical scores than the biologically superior alignments produced by our new approach. This clearly demonstrates that the objective function currently used in DIALIGN is flawed. Possible reasons for the shortcomings of this objective function are discussed in Subramanian *et al.* (2005).

Thus, it is unlikely that DIALIGN can be further improved by applying more efficient optimization algorithms on the basis of the current objective function. Instead, it seems worthwhile to investigate novel objective functions for the segment-based alignment problem. Probabilistic approaches may provide a way of optimising the weight parameters used by DIALIGN, for example, by using *conditional random fields*. Also, it seems worthwhile to use (partial) multiple local alignments instead of pairwise fragments as building blocks for MSA and to develop improved objective functions based on such partial multiple alignments.

## ACKNOWLEDGEMENTS

We thank Andreas Dress and Etienne Birmelé for helpful discussions and Rasmus Steinkamp for help with the computer cluster of the *Department of Bioinformatics* in Göttingen.

*Funding*: Deutsche Forschungsgemeinschaft, in English: German Research Foundation (grant MO 1048/6-1 to B.M.); a fellowship of the Max-Planck-Society (to F.P.).

*Conflict of Interest*: none declared.

## REFERENCES

Abdeddaïm,S. and Morgenstern,B. (2001) Speeding up the DIALIGN multiple alignment program by using the 'greedy alignment of biological sequences library' (GABIOS-LIB). *Lect. Notes Comput. Sci.*, **2066**, 1–11.

Altschul,S.F. *et al.* (1997) Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Res.*, **25**, 3389–3402.

Bailey,T.L. and Elkan,C. (1994) Fitting a mixture model by expectation maximization to discover motifs in biopolymers. In *Proceedings of the Second International Conference on Intelligent Systems for Molecular Biology*. AAAI Press, Atlanta, pp. 28–36.

Cormen,T. *et al.* (2001) *Introduction to Algorithms*. MIT Press, Cambridge, MA; London, England.

Do,C.B. *et al.* (2005) ProbCons: probabilistic consistency-based multiple sequence alignment. *Genome Res.*, **15**, 330–340.

Do,C.B. *et al.* (2006) CONTRAlign: discriminative training for protein sequence alignment. In *Proceedings Research in Computational Molecular Biology '06*. Vol. of 3909 *Lecture Notes in Computer Science*. Springer-Verlag.

Dress,A. *et al.* (2008) Stability of multiple alignments and phylogenetic trees: an analysis of ABC-transporter proteins. *Algorithms Mol. Biol.*, **3**, 15.

Durbin,R. *et al.* (1998) *Biological sequence analysis*. Cambridge University Press, Cambridge.

Eddy,S. (1995) Fast and sound two-step algorithms for multiple alignment of nucleic sequences. In *Proceedings of Intelligent Systems for Molecular Biology '95*. AAAI Press, Atlanta, pp. 114–120.

Edgar,R. (2004) MUSCLE: multiple sequence alignment with high score accuracy and high throughput. *Nucleic Acids Res.*, **32**, 1792–1797.

Edgar,R.C. and Batzoglou,S. (2006) Multiple sequence alignment. *Curr. Opin. Struct. Biol.*, **16**, 368–373.

Edmonds,J. and Karp,R.M. (1972) Theoretical improvements in algorithmic efficiency for network flow problems. *J. ACM*, **19**, 248–264.

Ford,L. and Fulkerson,D. (1956) Maximal flow through a network. *Can. J. Math*, **8**, 399–404.

Gotoh,O. (1990) Consistency of optimal sequence alignments. *Bull. Math. Biol.*, **52**, 509–525.

Gusfield,D. (1997) *Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology*. Cambridge University Press, Cambridge.

Katoh,K. *et al.* (2005) MAFFT version 5: improvement in accuracy of multiple sequence alignment. *Nucleic Acids Res.*, **33**, 511–518.

Kececioglu,J.D. *et al.* (2000) A polyhedral approach to sequence alignment problems. *Discrete Appl. Math.*, **104**, 143–186.

Kemena,C. and Notredame,C. (2009) Upcoming challenges for multiple sequence alignment methods in the high-throughput era. *Bioinformatics*, **25**, 2455–2465.

Lassmann,T. and Sonnhammer,E.L. (2002) Quality assessment of multiple alignment programs. *FEBS Lett.*, **529**, 126–130.

Lassmann,T. and Sonnhammer,E.L. (2005) Kalign an accurate and fast multiple sequence alignment algorithm. *BMC Bioinformatics*, **6**, 298.

Lenhof,H.-P. *et al.* (1999) An exact solution for the segment-to-segment multiple sequence alignment problem. *Bioinformatics*, **15**, 203–210.

Morgenstern,B. (1999) DIALIGN 2: improvement of the segment-to-segment approach to multiple sequence alignment. *Bioinformatics*, **15**, 211–218.

Morgenstern,B. (2000) A space-efficient algorithm for aligning large genomic sequences. *Bioinformatics*, **16**, 948–949.

Morgenstern,B. (2002) A simple and space-efficient fragment-chaining algorithm for alignment of DNA and protein sequences. *Appl. Math. Lett.*, **15**, 11–16.

Morgenstern,B. *et al.* (1996) Multiple DNA and protein sequence alignment based on segment-to-segment comparison. *Proc. Natl Acad. Sci. USA*, **93**, 12098–12103.

Morgenstern,B. *et al.* (2005) Multiple sequence alignment with user-defined constraints at GOBICS. *Bioinformatics*, **21**, 1271–1273.

Morgenstern,B. *et al.* (2006) Multiple sequence alignment with user-defined anchor points. *Algorithms Mol. Biol.*, **1**, 6.

Morrison,D.A. (2006) Multiple sequence alignment for phylogenetic purposes. *Aust. Syst. Bot.*, **19**, 479–539.

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

Notredame, C., Higgins, D., and Heringa, J. (2000). T-Coffee: a novel algorithm for multiple sequence alignment. *J. Mol. Biol.*, **302**, 205–217.

Pitschi,F. (2008) Sequence similarity, motif detection and alignments with N-local decoded anchor points. PhD Thesis, Universität Leipzig, Faculty of Mathematics and Computer Science, Germany.

Smith,T.F. and Waterman,M.S. (1981) Identification of common molecular subsequences. *J. Mol. Biol.*, **147**, 195–197.

Stoye,J. *et al.* (1998) Rose: generating sequence families. *Bioinformatics*, **14**, 157–163.

Subramanian,A.R. *et al.* (2005) DIALIGN-T: an improved algorithm for segment-based multiple sequence alignment. *BMC Bioinformatics*, **6**, 66.

Subramanian,A.R. *et al.* (2008) DIALIGN-TX: greedy and progressive approaches for the segment-based multiple sequence alignment. *Algorithms Mol. Biol.*, **3**, 6.

Thompson,J.D. *et al.* (1994) CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice. *Nucleic Acids Res.*, **22**, 4673–4680.

Thompson,J.D. *et al.* (2005) BAliBASE 3.0: latest developments of the multiple sequence alignment benchmark. *Proteins Struct. Funct. Bioinform.*, **61**, 127–136.

Vingron,M. and Argos,P. (1991) Motif recognition and alignment for many sequences by comparison of dot-matrices. *J. Mol. Biol.*, **218**, 33–43.