Algorithms for Improving Multiple Sequence Alignments and Building Evolutionary Trees Using Insertion and Deletion Events

Chantal Korostensky, Ulrike Stege and Gaston Gonnet
e-mail: {korosten, stege, gonnet}@inf.ethz.ch
Institute for Scientific Computing, ETH Zürich, Switzerland

The construction of multiple sequence alignments (MSAs) is a fundamental problem in computational biology. Usually practical algorithms fail to produce an exact solution corresponding to the underlying model due to the complexity of the problem. The main problem in the biological point of view is the misplacement of gaps. Yet biological meaningful MSAs are important for both structure and function prediction. We present an algorithm that takes as input an MSA that was built by any algorithm and produces an improved MSA. With the use of a well defined scoring function we can determine the maximum possible score of a given MSA. The algorithm uses the fact that if the sequences in the MSA are related, there exists a (usually unknown) evolutionary tree for the sequences. We treat gaps as a special character, as gaps play an important role in both protein structure and evolutionary history in order to improve the alignment. In addition, we show how gaps can be used to reconstruct evolutionary trees.

1 Introduction

Several structure prediction methods and methods for evolutionary tree construction are based on multiple sequence alignments (MSAs)[1,2,3,4,5,6,7,8,9]. The quality of an MSA highly influences the outcome of structure prediction. As a first step the MSA is scanned for parse regions. Parses are regions in the protein between secondary structures ($\alpha-$helices or $\beta-$sheets), such as loops. Usually gaps (insertions or deletion events) indicate parses. This is the most important step for assigning parses, and misplaced gaps are problematic for structure prediction, because once a parse region is chosen, no other secondary structures can traverse that position anymore.

The problem of calculating MSAs can be viewed as the problem of inserting gaps at the correct places. There are many MSA algorithms[11,12,13,14,15], but none of them produce optimal alignments (for realistic instances of the problem) due to the complexity (the common models for calculating MSAs in are known to be NP-complete[10]).

In this paper we present two different algorithms for dealing with misplaced gaps in MSAs. The first algorithm improves given MSAs by reducing the number of evolutionary events. The second algorithm shows how to build an evolutionary tree from an MSA where misplaced gaps prevent a unique solution.

Gaps play a special role both in terms of structure and evolution (see following sections). This fact can be used to improve MSAs and to construct the corresponding evolutionary tree. It is important to develop algorithms that use as much biological information as possible, such that structure prediction and evolutionary tree construction based on MSAs become more reliable.

```
A: RPCVCP___VLRQAAQ__QVLQRQIIQGPQQLRRLF_AA
B: RPCACP___VLRQVVQ__QALQRQIIQGPQQLRRLF_AA
C: KPCLCPKQAAVKQAAH__QQLYQGQLQGPKQVRRAFRLL
D: KPCVCPRQLVLRQAAHLAQQLYQGQ____RQVRRAF_VA
E: KPCVCPRQLVLRQAAH__QQLYQGQ____RQVRRLF_AA
```

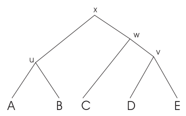

Figure 1: The sequences $\langle A, B, C, D, E\rangle$ of the MSA on the left are related by the evolutionary tree on the right. The internal nodes in the tree represent unknown ancestor sequences.

Definition 1.1 *Given is a set of sequences $S = \{s_1, .., s_n\}$ with $s_i \in \Sigma^*$ where Σ is a finite alphabet. A Multiple Sequence Alignment (in short, MSA) consists of a set of sequences $\mathrm{A} = \langle a_1, a_2, .., a_n\rangle$ with $a_i \in \Sigma'^*$ where $\Sigma' = \Sigma \cup \{\_\}$, $\_ \notin \Sigma$. Furthermore $|a_i| = |a_j|$ for all $i, j \in \{1..n\}$. The sequence obtained by removing all gap "_" characters from a_i is equal to s_i.*

Definition 1.2 *The character "_" or any contiguous sequence of "_" within an aligned sequence $a_i \in \mathrm{A}$ is called a gap. A gap corresponds to an insertion or deletion event (indel). The length of a gap $|g|$ is the number of "_".*

Definition 1.3 *A tree $T(S) = (V, E, L)$ is a binary, leaf-labeled tree with vertices V, edges E and leaf labels $L = \{s_1, .., s_n\}$.*

In our context, for a set of sequences $S = \{s_1, .., s_n\}$ the tree $T(S)$ is the tree that corresponds to the evolutionary history of the sequences of S. The internal nodes V represent the (usually unknown) ancestor sequences.

Definition 1.4 *Let $\mathcal{A}$ be the set of all possible MSAs that can be generated for a given set of sequences $S = \{s_1, s_2, .., s_n\}$. An MSA scoring function $F : \mathcal{A} \to I\!R$ is a function, where a larger value corresponds to a better alignment (otherwise the function can be multiplied by -1). The optimal MSA $\bar{\mathrm{A}} \in \mathcal{A}$ is an MSA such that $F(\bar{\mathrm{A}}) = \max_{\mathrm{A} \in \mathcal{A}} F(\mathrm{A})$.*

1.1 Gaps and Trees

In contrast to mutations indels do not happen very frequently, and a second indel event at the same place of the same size are very rare. Indels usually have a great impact on the resulting protein structure, whereas a mutation usually has little effect. In the Markovian model of evolution, we assume that mutations can happen anywhere in a protein and are independent of each other.

This is a useful and logical assumption, but it is definitely not true for indels [23]. Hence gaps should be treated separately.

Gaps and protein structure Indels appear more frequently in loops than in α helices or β strands, as loops are usually more flexible and do not influence the overall structure of the protein as much [23].

Gaps and Trees Assume we are given an optimal MSA A and assume further that we know the tree $T(\mathrm{A})$. Each indel event can be mapped to exactly one edge in the tree (see Figure 2). The result of a deletion event are gaps that appear in all sequences $a_i \in \mathrm{A}$ of the subtree where the event took place. In contrast, when the event was an insertion, gaps appear in all sequences that are not in the subtree below the event.

Example 1.1: In the alignment of Figure 2, there are 4 indel events, which we call gap blocks (see Definition 3.2). The first gap block consists of two gaps that appear in sequences A and B. If x is the ancestor of all the sequences, then the event must either be a deletion in the ancestor of sequences A and B, or it can be an insertion on the edge $x \to w$. Accordingly, event 2 is an insertion in sequence D, event 3 represents a deletion in the ancestor of D and E, and finally event 4 is an insertion in sequence C.

```
Event    1        2       3        4
A: RPCVCP___VLRQAAQ__QVLQRQIIQGPQQLRRLF_AA
B: RPCACP___VLRQVVQ__QALQRQIIQGPQQLRRLF_AA
C: KPCLCPKQAAVKQAAH__QQLYQGQLQGPKQVRRAFRLL
D: KPCVCPRQLVLRQAAHLAQQLYQGQ____RQVRRAF_VA
E: KPCVCPRQLVLRQAAH__QQLYQGQ____RQVRRLF_AA
```

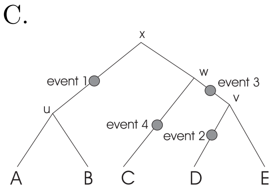

Figure 2: Indel events are shown in the evolutionary tree (right) of the MSA (left).

1.2 Scoring Functions

In all our examples, the scoring function $F(\mathrm{A})$ of an MSA is similar to the sum-of-pairs measure [21], except that we do not add all scores of all pairwise alignments, but only the scores of the pairwise alignments in a circular order $C(S)$ with respect to the associated evolutionary tree. Such an order can be determined without the knowledge of an evolutionary tree. For a detailed explanation see [22].

Definition 1.5 *The score of the induced MSA-derived pairwise alignment $\underline{MPA(a_i, a_j)}$ of two sequences $a_i, a_j \in \mathrm{A}$ is the score of the alignment of the two strings a_i and a_j, where any scoring function for pairwise alignments can be used. Opposing gaps may be removed.*

Usually the optimal score for pairwise alignments is determined via standard dynamic programming [18,19]. An affine gap cost is used according to the formula

$a + l \cdot b$, where a is a fixed gap cost, l is the length of the gap and b is the incremental cost [20].
The circular sum score $CS(\mathrm{A})$ of an MSA A is the sum of MPA scores in circular order C divided by two:

Definition 1.6 *The score* $\underline{CS(\mathrm{A})}$ *of an MSA* A *is defined as:* $CS(\mathrm{A}) = \frac{1}{2}\sum_{i=1}^{n} MPA(a_{C_i}, a_{C_{i+1}})$ *where* $C_{n+1} = C_1$*, and* C *is a circular order with respect to the evolutionary tree of* A.

Whenever we use the term score of an MSA, we mean the CS score. The term "best possible score" of an MSA refers to the CS score using the optimal scores of the pairwise alignments instead of the MPA scores. For a better explanation see [22].

In the following we classify the major problems that can arise when MSAs are built. We then present an algorithm that is derived from biological observation to improve MSAs. Then we show how to reconstruct an evolutionary tree from the gaps. Furthermore, we show how the solution of a vertex cover problem can be used for resolving conflicts that can arise when constructing trees from gaps.

2 Classification of the problems

When an MSA is not optimal, then there are three major types of misplacement of gaps: a) *shifted gaps*: gap blocks of same size appear shifted horizontally b) *split gaps*: gap blocks are cut apart vertically and appear as two gap blocks and c) *shifted and split gaps*, which is a combination of a) and b).

Shifted gaps Very often algorithms produce MSAs where gaps of the same size appear slightly shifted horizontally. From a biological point of view there are two possibilities: a) the gaps are in a loop. Since gaps appear frequently in loops, it is possible that two events produce indels of the same size at the same place. So it makes sense to group the gaps that are within a short distance for structural reasons. b) the gaps are *not* in a loop. Since gaps do not appear so frequently outside of loops, it is not very likely that two *different* events produce gaps of the same size at the same place. It is more likely that due to mutations at the ends of the gaps, the algorithm was forced to place the gap incorrectly. So in this case it also makes sense to group the gaps together.

When gaps are grouped together, the number of evolutionary events is reduced (e.g gaps 1 and 1' from example 2.1). This can be viewed as "pushing" and fusing the events up in the tree, if the corresponding tree is known (see Figure 3). Since gaps are expensive (large negative score), reducing the number of indel events reduces the cost and therefore the score of the MSA will be increased in most cases.

Example 2.1: In this example gaps 1 and $1'$ have the same size (length 3) and appear at almost the same place. When the gaps are grouped together, the score increases and the alignment makes more biological sense (if we ask an expert [23]).

```
Score:   135.466      Score:   240.139
Maximum: 240.139      Maximum: 240.139

Event 1   1'          Event     1
A: RP___CVCPVLRQ      A: RPCVCP___VLRQ
B: RPCACP___VLRQ      B: RPCACP___VLRQ
C: KPCVCPRQLVLRQ  =>  C: KPCVCPRQLVLRQ
D: KPCVCPRQLVLRQ      D: KPCVCPRQLVLRQ
E: KPCLCPKQAAVKQ      E: KPCLCPKQAAVKQ
```

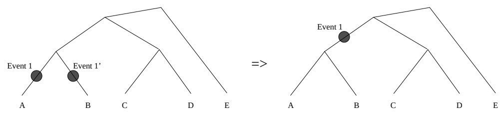

Figure 3: When gaps of the same size but from different sequences are grouped together, this corresponds to "pushing" the event up in the evolutionary tree.

Split gaps Often algorithms produce several gaps close together in one sequence. In reality such gaps often belong to just one indel event. When two such gaps g_i and g_j are depicted in a tree, they appear on the same edge in the tree. When the gaps are combined only one event is left (see figure 4).

```
Event 1    1'         Event     1
A: RPC__VCP_VLRQ      A: RPCVCP___VLRQ
B: KPCVCPRQLVLRQ  =>  C: KPCVCPRQLVLRQ
B: KPCVCPRQLVLRQ      D: KPCVCPRQLVLRQ
D: KPCLCPKQAAVKQ      E: KPCLCPKQAAVKQ
```

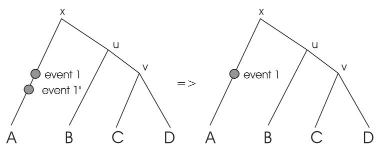

Figure 4: When close gaps are grouped together, this corresponds to "fusing" the event in the evolutionary tree.

Split and shifted gaps In many MSAs both situations will arise: gaps are shifted and split.

Example 2.2: In the MSA of Figure 5, sequence A has two gaps and sequence B has three gaps. If we push the gaps together and move them to the same position, we combine 5 events to one single event. In addition, the resulting alignment has a better score (the optimal score in this case) and also makes more sense from a biological point of view.

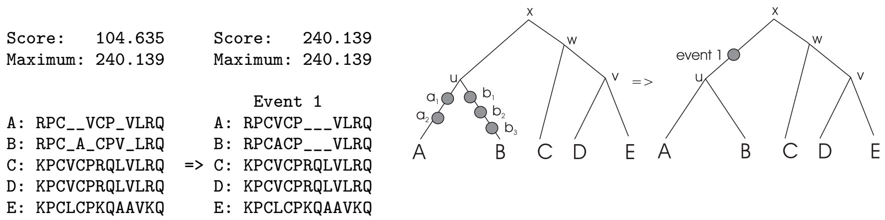

Figure 5: Here 5 events can be reduced to one event by shifting and fusing the gaps.

3 Methods

We can build improved MSAs by shifting and combining gaps. The idea is to find split and/or shifted gaps, combine them and move them to a position that makes more sense biologically, provided that the global score increases. In our experience, the scoring function we use has always been in accordance with the experts [23].

3.1 An Algorithm For Finding All Possible Gap Blocks

For simplicity, in the following each gap is represented by an integer which is its length. All gaps of the MSA are represented by L:

Definition 3.1 *The gaps of an MSA* $\mathrm{A} = \langle a_1, a_2, .., a_n \rangle$ *are represented as an* *ordered list L* *of lists,* $L = \langle L_1, .., L_n \rangle$*, where each* L_i *is the sequence of gaps in* a_i*.* $|L_i|$ *is the number of gaps in* a_i*, and the element* L_{ij} *is the length of the* j^{th} *gap in* a_i*.*

Definition 3.2 *A* *gap block* B *of an MSA* $\mathrm{A} = \langle a_1, a_2, .., a_n \rangle$ *is defined as* $B = \langle B_1, .., B_n \rangle$*, where each* B_i *is either a contiguous subsequence denoted by the* first *and* last *indices in* L_i *(i.e.* $B_i[first], B_i[last]$*) or* B_i *is empty. For each non empty* B_i*, the sum of the gap lengths must be the same:* $\forall i, j : \sum_{k=B_i[first]}^{B_i[last]} L_{ik} = \sum_{k=B_j[first]}^{B_j[last]} L_{jk}$.

All possible B's are considered except for:

a) Whenever there are gaps in different sequences at the same position of the same length, we assume that they are placed correctly, because such gaps already *are* in a block, and we do not need to disrupt them (e.g. if $L_{1,j_1} = L_{2,j_1} = L_{3,j_1}$ and the three gaps start at the same place in the MSA A, we treat them as one gap).

b) If the gaps start more than W characters away from each other in the MSA, then we discard them.

c) We only consider r_{max} gaps at a time: $B_i[last] - B_i[first] \leq r_{max}$

Problem 3.1 (Find all gap blocks) *Given is an MSA* $\mathrm{A} = \langle a_1, .., a_n \rangle$. *The problem is to find all possible gap blocks* B *in* A *(see Definition 3.2).*

Example 3.1: The MSA in Figure 6 is transformed into a list $L = \langle L_1, .., L_n \rangle$. From this list, many possible gap blocks can be formed. Two valid gap blocks are shown on the left in Figure 7 (empty lists are not shown). In both, B^1 and B^2, each B_i^1, B_i^2 is a subsequence of L_i, and the sum of the integers in each B_i^1, B_i^2 is the same. One invalid gap block is shown on the right in Figure 7. In B^3, the sum of the lengths in one B_i^3 is not the same.

a_1	PDCVCPTLKQAAKAV____RLQG_QHQ____PMQ_______VRKIYQTAKHLPNVC	$L_1 = \langle 4, 1, 4, 8 \rangle$
a_2	PVCVCPTLRQAARAV____SLQG_QHG____PFQ_______SRKIYKTAKYLPNIC	$L_2 = \langle 4, 1, 4, 8 \rangle$
a_3	PVCVCPTLKQAARAV____SLQG_QHG____PFQ_______SRKIYQSAKYLPNIC	$L_3 = \langle 4, 1, 4, 8 \rangle$
a_4	QVCVCPTLKQAAKSVRVQGQHG________________PFQSTRIYQIAKNLPNVC	$L_4 = \langle 16 \rangle$
a_5	PLCVCPTLKGASKAVKQQVRQQLEQQG____Q____QGPHVISRIYQTATHLPKVC	$L_5 = \langle 4, 4 \rangle$
a_6	PLCVCPTLKGASKAVKQQVRQ___QQGQQGQQLQQV_____ISRIYQTATHLPKVC	$L_6 = \langle 3, 5 \rangle$
a_7	PLCVCPTLKGASKAVRQQVRQ___QQG___QQMQGQQMQQVISRVYQTATHLPRVC	$L_7 = \langle 3, 3 \rangle$
a_8	PLCVCPTLKGASKAVKQQIRQQGQQQGQQGQQLQHE_____ISRIYQTATHLPRVC	$L_8 = \langle 5 \rangle$
a_9	PLCVCPTLKGAAKAVKQQIQQQGQQHGQQGQQLQHE_____IRRIYQTATHLPKVC	$L_9 = \langle 5 \rangle$
a_{10}	PLCVCPTLKGASKAVKQQIQQQGQQQG__KLQM________VSRIYQTATHLPKVC	$L_{10} = \langle 2, 8 \rangle$
a_{11}	PLCVCPTLKGASKAVKQQIQQQGQQQG__KQQM________VSRIYQTATHLPKVC	$L_{11} = \langle 2, 8 \rangle$
a_{12}	PLCVCPTLRGASKAVKQQIQQQEQ__QQGKQQM________VNRIYQTATHLPKVC	$L_{12} = \langle 2, 8 \rangle$

Figure 6: Each sequence of the MSA is transformed to a list L of integers

$B^1 = \langle B_1^1, B_2^1, B_3^1 \rangle$	$B^2 = \langle B_2^2, B_3^2, B_5^2, B_6^2 \rangle$	$B^3 = \langle B_2^3, B_3^3, B_5^3, B_6^3 \rangle$
$B_1^1 = (2, 4)$	$B_2^2 = (4, 4)$	$B_2^3 = (4, 4)$
$B_2^1 = (2, 4)$	$B_3^2 = (4, 4)$	$B_3^3 = (3, 4)$
$B_3^1 = (2, 4)$	$B_5^2 = (1, 2)$	$B_5^3 = (1, 2)$
	$B_6^2 = (1, 2)$	$B_6^3 = (1, 2)$

Figure 7: On the left are two valid gap blocks B^1 and B^2 on the left that correspond to the alignment in Figure 6 (empty lists are not shown). The gap block B^3 on the right is invalid: in B^3 the sum of the gap lengths in each B_i is not the same.

Most proteins are composed of *domains*, which are usually at most 70 amino acids long. In order to find all possible gap blocks, we restrict the problem to the size of a domain, because it does not make sense biologically to combine gaps which are further from each other than a domain size. Usually we don't know where a domain starts and where it ends, therefore we define a window of width W and height n that slides along the MSA. W is the maximum number of characters in one sequence of the MSA we look at. Within this window, we want to find all combinations of gaps that build blocks. In the following, when we use L, we mean an L restricted by such a window.

Definition 3.3 *The parameter* W *is the maximum number of positions we analyze at in a sequence of an MSA. Let* w_i *be the maximum possible number* *of gaps that can appear in L_i for a window of size W. It is clear that $w_i \leq (W-1)/2$ and $w_i \leq |L_i|$.*

In addition, we restrict the maximum number of gaps in each sequence we will combine to a block.

Definition 3.4 *The parameter r_{max} is the maximum number of gaps that can be contained in any B_i of a gap block B.*

Let us now determine the number of possible gap blocks that can be formed. To build a gap block B, for each row we can either choose gaps or not. Since there are at most w gaps in each row for each position, we can build $m(\min(r_{max}, \frac{W-1}{2}+1))^n$ blocks in the worst case, or more precisely, $m\prod_{i=1}^{n}\min(r_{max}, w_i)+1$, when m is the length of a sequence a_i in the MSA A.

Procedure 3.1: *FindAllBlocks*
Input: L, r_{max}
Output: a list of gap blocks P
$P = \langle\rangle$;
for $r = 1$ **to** r_{max} **do**
  for $i = 1$ **to** n **do**
    compute the list P' of all possible blocks such that for each $B \in P'$:
      • $B_i = (1, r)$ and
      • for all $k > i$, $B_k[last] - B_k[first] \leq r$ and
        $B_k = (1, s)$ s.t.$\sum_{t=B_i[first]}^{B_i[last]} L_{it} = \sum_{t=B_k[first]}^{B_k[last]} L_{kt}$;
      for all $B' \in P'$ **do**
        if B' contains more than one gap **then** add B' to P;
        $L' = copy(L)$;
        remove all elements from L' that correspond to B';
        if L' is not empty **then** $P = P \cup FindAllBlocks(L', r_{max})$;
      od;
    od;
  od;
  return P;

In real cases, the number of gaps is small enough so that the problem is solvable in a reasonable amount of time for r up to 5 and for W up to around 70 (see Section 3.3).

3.2 Improving Alignments With Gap Blocks

Definition 3.5 *The score $\underline{S(B)}$ of a block $B \in$ A is the maximum possible score of the MSA* A *that can be obtained by shifting the gaps in* A *that correspond to the gaps within the block B.*

Once we have all the gap blocks we can optimize the MSA A by shifting the gaps in each B that correspond to the real gaps in the alignment A. The problem is that some gap blocks may overlap, that is two different gap blocks B^i and B^j may share the same gaps. Of course, only either B^i or B^j can be chosen in the final alignment. Whenever this happens, we want to select the block with the best score. For any two blocks B^i, B^j in P, if the block B^i overlaps with some other block B^j, score both according to Definition 3.5 and remove the block with the lower score from P.

For each block B in P, shift and combine the gaps in the MSA A that correspond to the block B^i and keep the change if the score increases.

Example 3.2: After applying the above procedures to the MSA in Figure 6, we end up with an MSA that has a much better score and only has 4 indel events. In addition, we can use the gaps of the MSA to construct the corresponding evolutionary tree (see Section 4).

```
Score of the alignment: 2400.081
Maximum possible score: 2472.632
 a1    PDCVCPTLKQAAKAVRLQG___________QHQPMQVRKIYQTAKHLPNVC
 a2    PVCVCPTLRQAARAVSLQG___________QHGPFQSRKIYKTAKYLPNIC
 a3    PVCVCPTLKQAARAVSLQG___________QHGPFQSRKIYQSAKYLPNIC
 a4    QVCVCPTLKQAAKSVRVQG___________QHGPFQSTRIYQIAKNLPNVC
 a5    PLCVCPTLKGASKAVKQQVRQQLEQQG___QQGPHVISRIYQTATHLPKVC
 a6    PLCVCPTLKGASKAVKQQVRQQQGQQG___QQLQQVISRIYQTATHLPKVC
 a7    PLCVCPTLKGASKAVRQQVRQQQGQQM_QGQQMQQVISRVYQTATHLPRVC
 a8    PLCVCPTLKGASKAVKQQIRQQGQQQGQQGQQLQHEISRIYQTATHLPRVC
 a9    PLCVCPTLKGAAKAVKQQIQQQGQQHGQQGQQLQHEIRRIYQTATHLPKVC
 a10   PLCVCPTLKGASKAVKQQIQQQGQQQG_____KLQMVSRIYQTATHLPKVC
 a12   PLCVCPTLKGASKAVKQQIQQQGQQQG_____KQQMVSRIYQTATHLPKVC
 a13   PLCVCPTLRGASKAVKQQIQQQEQQQG_____KQQMVNRIYQTATHLPKVC
```

3.3 Experimental Results

We did a large simulation where more than 1000 MSAs with 16, 20 and 30 sequences were randomly generated by simulating evolution. The length of each sequences was around 200. The score of the generated MSA was noted. The unaligned sequences were then given to two different algorithms, ClustalW [11] and ProbModel [15]. The resulting MSAs were scored and then given to our new algorithm to improve the alignments. The difference in score of the final alignment and the original alignment was noted. We also counted how often our algorithm improved the MSAs. The results are depicted in Table 3.3. In all cases, our gap algorithm could improve at least 64% of the MSAs.

4 Tree construction using gaps

Gap blocks can be viewed as binary state characters: each event partitions the tree into two groups, the sequences with the gap and the sequences without the gap. Each event tells us which sequences belong into the same subtree,

No Seq	ProbModel	ClustalW	before	after
16 %	65 %	71 %	44 %	71 %
20 %	64 %	74 %	29 %	41%
30 %	70 %	78 %	22 %	40 %

Figure 8: The columns ProbModel and ClustalW show the percentage of MSAs that have improved. The column *before* is the percentage of MSAs that scored at least as well as the real alignment before the improvement, the column *after* is the percentage after the gap algorithm was applied.

because when several sequences have the same gap, we know that the indel event happened in an *ancestor* of those sequences. In addition, it is not very likely that two equal indel events happen at exactly the same place of same length, therefore we ignore this case. We can use indel events to reconstruct the tree or at least parts of the tree. We want to build a minimum parsimony tree [7,8,9], i.e. minimum number of errors according to gap blocks. Gaps have been used before to reconstruct trees [24], but this has never been stated explicitly. We give a practical approach on how to use gaps for tree construction.

Example 4.1: The MSA in Figure 9 has four gap blocks. We use those blocks to partition the sequences (see Table 9). Event 1 partitions the tree into the sequences $\{A, B\}$ and $\{C, D, E\}$. Events 2 and 4 do not add any information. Event 3 completes the tree (see Figure 10).

```
Event      1           2          3          4
A: RPCVCP___VLRQAAQ__QVLQRQIIQGPQQLRRLF_AA
B: RPCACP___VLRQVVQ__QALQRQIIQGPQQLRRLF_AA
C: KPCVCPRQLVLRQAAHLAQQLYQGQ____RQVRRAF_VA
D: KPCVCPRQLVLRQAAH__QQLYQGQ____RQVRRLF_AA
E: KPCLCPKQAAVKQAAH__QQLYQGQLQGPKQVRRAFRLL
```

Event	Seqs with gaps	Seqs w/o gaps
1	{A, B}	{C, D, E}
2	{A, B, D, E}	{C}
3	{C, D}	{A, B, E}
4	{A, B, C, D}	{E}

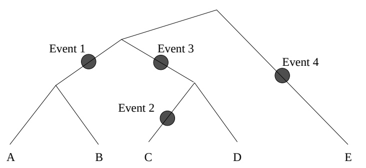

Figure 9: Partitioning of sequences using gaps. The indel events are depicted in the tree.

To reconstruct a binary tree with n leaves completely, we need $n-3$ different partitions, where each partition must contain at least two elements (two leaves), because an indel in just one sequence does not supply any information.

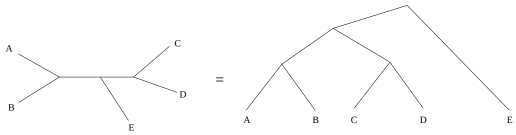

Figure 10: Tree constructed using all four events. On the left is an unrooted version, on the right a rooted version of the same tree.

4.1 Resolving conflicts

Usuall we do not have any guarantee that our MSA is correct. Hence we don't know if all the gaps appear at the right places. This leads to conflicts: assume that the gap in sequence B of event 4 in Figure 9 is shifted to the left by e.g. 3 positions. Let us consider events 1 and 4 only. Because of event 1 we conclude that sequences $\{A, B\}$ and $\{C, D, E\}$ are grouped together. But at the same time sequences $\{A, C, D\}$ and $\{B, E\}$ should also be grouped together, due to event 4, which leads to a contradiction. Generally we define a conflict as follows:

Definition 4.1 (Conflict) *Given an MSA* $\mathrm{A} = \langle a_1, .., a_n\rangle$ *and two indel events (gap blocks) that split the sequences into the sets* A_1, B_1 *(event 1) and* A_2, B_2 *(event 2). Then event 1 and event 2 have a conflict, if* $A_1 \not\subseteq A_2$, $B_1 \not\subseteq A_2$, $A_1 \not\subseteq B_2$ *and* $B_1 \not\subseteq B_2$ *(see Figure 9).*

Finding the minimum number of events to ignore such that we have no conflicts is called the "vertex cover problem" in graph theory.

Problem 4.1 (Vertex Cover) *Given is a graph* $G = (V, E)$ *with vertices* V *and edges* E*. The problem is to find the smallest subset* $V' \in V$ *such that for each edge* $(a, b) \in G : a \in V'$ *or* $b \in V'$*.*

In our case each indel event (or gap block) is a vertex, and whenever there is a conflict between two events an edge is drawn that connects the two vertices. A minimum vertex cover corresponds to a minimum number of events such that the MSA ignoring those events is conflict free. Once all conflicts are removed, we can build an evolutionary tree from the remaining events (if we have enough events).

The problem of finding the minimum vertex cover is NP-complete [25]. But usually the number of indel events is small (smaller than 50), when the size of the MSA is restricted to the size of a domain (which is usually at most 70 amino acids long). Vertex cover problems of this magnitude can be readily solved, there is a fixed parameter tractable algorithm [27] to solve the problem deterministically, when parameterized by the size of the vertex cover [26,29]. With this algorithm, conflict graphs with minimum vertex covers of size larger than 150 can be resolved in reasonable time. A more detailed version of the

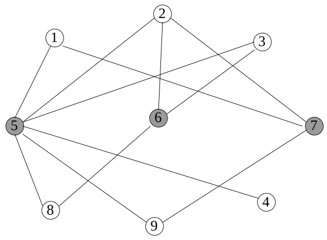

Figure 11: Gap conflict graph: each vertex corresponds to an indel event, each edge means a conflict between two events. Here an optimal solution is to remove vertices 5, 6 and 7.

paper can be obtained at *http://www.inf.ethz.ch/personal/korosten/papers/gaps.ps*. All algorithms described here have been implemented into the Darwin system [28] and are available upon request. The programs can also be used via our computational biochemistry server: *http://cbrg.ethz.ch*.

5 Example

With the following we show with a real example how an evolutionary tree can be constructed using gaps.

Example 5.1:

```
Score of the alignment: 1431.442
Maximum possible score: 1905.438

Event:       1   2    3      4      5   6   7       8    9
1       PDCVCPTLK_AAK___LQGKQQIQQQKAPQQG_QHQPMPNICQVRKIYQTAKHL
2       PVCV__TLR_AARAVKLQGKQQGQQQKAP_QQGQHLPFPNICQSRKIYKTAKYL
3       PVCV__TLK_AARIVKLQIRQQGQQQKAPQQGQHGP_FPNICQSRKIYQSAKYL
4       QVCVCPTLK_AAKAVRVQGRQQGQQQKAPQQA_QHGPFPNICQ____YQIAKNL
5       PLCVCPTLK_ASKVARQQVRQQLEQQKAP_QQGQQAPHPNICV____YQTATHL
6       PLCVCPTLKGASK___QQVR____QQKAPQQGQQLQ_QPNICVISRIYQTATHL
7       PLCVCPTLKGASK___QQVR____QQKAP_QQGQQMQQPNICVISRVY__ATHL
8       PLCVCPTLKGASK___QQIR____QQKAPQQAQQLQ_HPNICEISRIY__ATHL
9       PLCVCPTLKGAAK___QQIQQQGQQHKAPGQQGQQLQHPNICEIRRIYQTATHL
10      PLCVCPTLKGASK___QQIQQQGQQQKAPGQQGQQKLQPNICMVSRIYQTATHL
```

First we build a table that partitions the events from the MSA above:

In Table 12 we can see that events 1 and 2 do not have a conflict, because $a_1 = \{2, 3\} \in a_2 = \{1, 2, 3, 4, 5\}$, but events 1 and 5 have a conflict, because none of the sets is contained in the other. The problem is now to resolve those conflicts. To do this we build a conflict graph: each event corresponds to a vertex, and each edge represents a conflict between two events.

We want to remove the smallest number of events to resolve the conflicts. This can be done via vertex cover. The minimal solution is {5, 6, 7}. So the conflicts are most likely caused by events {5, 6, 7}.

Event	Seqs with gaps	Seqs without gaps	conflicts with other events
1	{2, 3}	{1, 4, 5, 6, 7, 8, 9, 10}	{5, 7}
2	{1, 2, 3, 4, 5}	{6, 7, 8, 9, 10}	{5, 6, 7}
3	{1, 6, 7, 8, 9, 10}	{2, 3, 4, 5}	{5, 6, 7}
4	{6, 7, 8}	{1, 2, 3, 4, 5, 6, 10}	{5, 7}
5	{2, 5, 7}	{1, 3, 4, 6, 8, 9, 10}	{1, 2, 3, 4 ,8 , 9}
6	{1, 4}	{2, 3, 5, 6, 7, 8, 9, 10}	{2, 3, 8}
7	{3, 6, 8}	{1, 2, 4, 5, 7, 9, 10}	{1, 2, 3, 4,9}
8	{4, 5}	{1, 2, 3, 6, 7, 8, 9, 10}	{5, 6}
9	{7, 8}	{1, 2, 3, 4, 5, 6, 9. 10}	{5, 7}

Figure 12: This table shows the conflicts that arise when gaps are used to partition the sequences in the MSA

If we go back to the alignment we can see that the gaps of events {5, 6, 7} can be grouped into a block that partitions the tree into {9, 10} and {1, 2, 3, 4, 5, 6, 7, 8}. There are no more conflicts left, and a tree can now be constructed (see Figure 13).

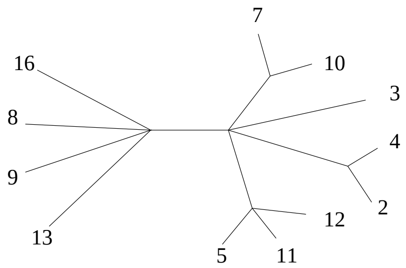

Figure 13: Tree construction of the conflict free partitions

```
Score of the alignment: 1905.438
Maximum possible score: 1905.438

Event:      1    2    3       4   {5,6,7}=5’       8   9
1       PDCVCPTLK_AAK___LQGKQQIQQQKAP_QQGQHQPMPNICQVRKIYQTAKHL
2       PVCV__TLR_AARAVKLQGKQQGQQQKAP_QQGQHLPFPNICQSRKIYKTAKYL
3       PVCV__TLK_AARIVKLQIRQQGQQQKAP_QQGQHGPFPNICQSRKIYQSAKYL
4       QVCVCPTLK_AAKAVRVQGRQQGQQQKAP_QQAQHGPFPNICQ____YQIAKNL
5       PLCVCPTLK_ASKVARQQVRQQLEQQKAP_QQGQQAPHPNICV____YQTATHL
6       PLCVCPTLKGASK___QQVR____QQKAP_QQGQQLQQPNICVISRIYQTATHL
7       PLCVCPTLKGASK___QQVR____QQKAP_QQGQQMQQPNICVISRVY__ATHL
8       PLCVCPTLKGASK___QQIR____QQKAP_QQAQQLQHPNICEISRIY__ATHL
9       PLCVCPTLKGAAK___QQIQQQGQQHKAPGQQGQQLQHPNICEIRRIYQTATHL
10      PLCVCPTLKGASK___QQIQQQGQQQKAPGQQGQQKLQPNICMVSRIYQTATHL
```

1. D. Gerloff, T. Jenny, L. Knecht, G. Gonnet, and S. Benner. *FEBS*, 318(2):118–124, 1993.
2. D. Gerloff, C. Cohen, C. Korostensky, M. Turcotte, G. Gonnet, and S.A. Benner. *Proteins: Struct. Funct. Genet.*, 27:450–458, 1997.
3. B. Rost and C. Sander. *Proc. Natl. Acad. Sci. U.S.A.*, 90:7558 – 7562, 1993.
4. D. Frishman and P. Args. *Protein Engineering, in press*, 256, 1996.
5. Lachlan H. Bell, John R. Coggins, and James E. Milner-White. *Protein Engineering*, 6(7):683–690, 1993.
6. Robert B. Russell and Geoffrey J. Barton. *J. Molecular Biology*, 234:951–957, 1993.
7. G. Estabrook, C. Johnson, and F. McMorris. *Math. Biosciences*, 29:181 – 87, 1976.
8. D. Sankoff. *SIAM J. Appl. Math.*, 28(35 – 42), 1975.
9. A. Dress and M. Steel. *Appl. Math. Lett.*, 5:3 – 6, 1993.
10. T. Jiang and L. Wang *J. Comp. Biol.*, 1:337 – 48, 1994.
11. J. Thompson, D. Higgins, and T Gibson. *Nucleic Acids Research*, 22:4673–4680, 1994.
12. Xiaoqiu Huang. *CABIOS*, 10(3):227–235, 1994.
13. Adam Godzik and Jeffrey Skolnick. *CABIOS*, 10(6):587–596, 1994.
14. Sandeep K. Gupta, John Kececioglu, and Alejandro A. Schaffer. In *J. Computational Biology*, 1996.
15. Gaston H. Gonnet and Steven A. Benner. In *SWAT96*, 1996.
16. M. Dayhoff, R. Schwartz, and B. Orcutt. In M. Dayhoff, *Atlas of Protein Sequence and Structure*, volume 5, pages 345–352. 1978.
17. Gaston H. Gonnet, Mark A. Cohen, and Steven A. Benner. *Science*, 256:1443–1445, 1992.
18. Temple F. Smith and Michael S. Waterman. *J. Mol. Biol.*, 147:195–197, 1981.
19. O. Gotoh. *J. Mol. Biol.*, 162:705–708, 1982.
20. S. Benner, M. Cohen, and G. Gonnet. *J. Molecular Biology*, 229:1065–1082, 1993.
21. Humberto Carillo and David. Lipman. *SIAM J. Appl. Math.*, 48(5):1073–1082, 1988.
22. Gaston H. Gonnet and C. Korostensky. *J. Comp. Biol.*, 1999. accepted.
23. S. Benner. personal communication.
24. S. Baldauf and J. Palmer. *Proc. Natl. Acad. Sci. USA*, 90(11):558 – 62, 1993.
25. Richard M. Karp. Plenum Press, NY, 1972.
26. U. Stege and M. Fellows. *Technical Report, CS, ETH Zuerich*, 318, 1999.
27. Rodney G. Downey and Michael R. Fellows *Springer-Verlag*, 1998.
28. Gaston H. Gonnet, Darwin Tutorial, CS, *ETH Zuerich* 1994.
29. U. Stege, Ph.D. thesis draft, CS, *ETH Zuerich*, June 1999. 1994.