Nada technical report: TRITA-NA-0316
Settling the Intractability of Multiple Alignment

Isaac Elias

Dept. of Numerical Analysis and Computer Science,
Royal Institute of Technology, Stockholm, Sweden
isaac@nada.kth.se

Abstract. In this paper some of the most fundamental problems in computational biology are proved intractable. The following problems are shown NP-hard for all binary or larger alphabets under all fixed metrics: MULTIPLE ALIGNMENT with SP-score, STAR ALIGNMENT, and TREE ALIGNMENT (for a given phylogeny). Earlier these problems have only been shown intractable for sporadic alphabets and distances, here the intractability is settled. Moreover, CONSENSUS PATTERNS and SUBSTRING PARSIMONY are shown NP-hard.
The new cases for which intractability is established are of significant practical interest. For example, these are core problems in areas such as protein structure prediction, phylogeny, and gene regulation.

1 Introduction

Multiple sequence alignment is at the very core of many computational problems in molecular biology. Different variations of multiple sequence alignment occur in areas such as protein structure prediction, phylogeny (inference of evolutionary history among species), and localization of functionally important units in biological sequences. As the field of bioinformatics grows these problems, and several of their variations, become increasingly important. Although the results in this paper are not surprising, the significance of the problems make the results both interesting and important.

The evolutionary process is driven by mutation and natural selection. DNA sequence similarity is therefore a good indication of common evolutionary origin and function. With *pairwise alignment* two sequences are aligned while allowing errors such as substitutions, insertions and deletions of symbols. The idea is that these errors model the mutations occurring in DNA sequence replication.

Multiple alignment is the natural extension of pairwise alignment, and also a much more powerful tool. Typically, when sequence similarity is weak, multiple alignment may find similarities which pairwise alignments would not. However, pairwise alignment is solvable in polynomial time and multiple alignment is "not".

Many scoring schemes have been suggested to measure the cost of a multiple alignment. In this paper the focus is on the sum-of-pairs score (SP-score), STAR ALIGNMENT, and TREE ALIGNMENT. The three scoring schemes are in many aspects different and are therefore considered as separate problems. There are numerous papers on practical applications of these problems, e.g. [CWC92,AMS+97,BA86,SC83,AL89] to mention just a few.

In [WJ94] Wang and Jiang gave a short NP-hardness proof for the SP-score under a non-metric distance measure over a 4 symbol alphabet. This result was then improved by Bonizzoni and Vedova [BV01], who showed that the problem is NP-hard for the binary alphabet and a specific metric. The result was extended further by Just [Jus01] to cover many metrics, and also under some non-metrics the problem was proved APX-complete. However, all metrics were not covered and in particular not the unit metric. We build on some of the ideas developed by Bonizzoni and Vedova to show that the problem is intractable for all binary or larger alphabets under any metric.

In [WJ94] STAR ALIGNMENT was proved to be APX-complete over a 7 symbol alphabet, however the symbol distance did not have the property of identity nor that of triangle inequality. In [LMW99] Li et al. gave a PTAS and an NP-hardness result under the unit metric for a version of STAR ALIGNMENT in which there was a restriction on the number of gaps. Moreover, in [SP01] the problem was proved NP-hard for a 6 symbol metric. Wang and Jiang also proved that TREE ALIGNMENT is NP-hard for a specific metric over a 4 symbol alphabet. Later in two companion papers [WJL96,WJG01] they gave a couple of nice PTASs working for all metrics. In this paper both problems are proved intractable for all binary or larger alphabets under any metric, thereby settling the complexity[1] of TREE ALIGNMENT. We emphasize that hardness results for non-metrics are easier to come by and that these do not cover the problems considered in practice. Moreover, by considering metrics in general, this paper covers most, if not all, variations occurring in practice.

Rather than finding a consensus for the strings as a whole it is sometimes of biological interest to focus on the consensus of well conserved regions, e.g. in gene regulation. A well conserved region in biological sequences relates to a functionally important unit. CONSENSUS PATTERNS [PS00,BT02] and SUBSTRING PARSIMONY [BST02] are natural formalizations of the problem of finding the most conserved region. While our NP-hardness result for SUBSTRING PARSIMONY is new, CONSENSUS PATTERNS has earlier been proved NP-hard [LMW01,BST02,Aku98] and W[1]-hard in [FGN02]. Nonetheless, since the construction is similar to that of STAR ALIGNMENT, both proofs are given.

In the following section MULTIPLE ALIGNMENT with SP-score is first proved to be NP-hard for the binary alphabet under the unit metric, at the end the construction is extended to cover all metrics. In Sect. 3, STAR ALIGNMENT is proved to be NP-hard for binary or larger alphabets under any metric. Thereafter, in Sect. 4, we extend the construction to handle the structure of a tree, thereby equivalently showing that TREE ALIGNMENT is NP-hard. Finally, in Sect. 5, the NP-hardness proofs of CONSENSUS PATTERNS and SUBSTRING PARSIMONY are given.

2 MULTIPLE ALIGNMENT with SP-score is NP-hard

In this paper a *string* is a sequence of symbols from an alphabet Σ, typically $\Sigma = \{0, 1\}$. A *pairwise alignment* of two strings s_1 and s_2 is a $2 \times l$ matrix

[1] All problems considered in this paper have polynomially bounded optimal solutions and therefore an FPTAS can not exist unless P=NP.

A, where row one and two contain strings s_1 and s_2 interleaved by spaces, respectively. The spaces are represented by the symbol $'-' \notin \Sigma$. By the cost of A we mean $\mathrm{d}_A(s_1, s_2) = \sum_{i=1}^{l} \mu(r_1[i], r_2[i])$, where r_1 and r_2 are the rows in A and μ a predefined metric for symbols from the extended alphabet $\Sigma \cup \{-\}$. We call the least such cost, denoted $\mathrm{d}(s_1, s_2)$, the *evolutionary distance*.

The most simple of metrics, the *unit metric*, is the metric in which all non-zero distances are exactly 1. The cost of the minimum pairwise alignment under the unit metric is also referred to as the *edit distance* for strings. The edit distance is simply the minimum number of edit operations (substitutions, insertions, and deletions) needed to transform one of the strings into the other. In Table 1 the metric for the extended binary alphabet is depicted (the variables will reappear later).

Table 1.

	0	**1**	-
0	0	α	β
1	α	0	γ
-	β	γ	0

Definition 1 (MULTIPLE ALIGNMENT with SP-score). *A multiple alignment of a set $\mathcal{S}$ of k strings, is a $k \times l$ matrix A where row i contains string s_i interleaved by spaces. The SP-score (sum-of-pairs) for a multiple alignment is the sum of all pairwise distances between rows in the alignment; $\sum_{i=1}^{k} \sum_{j=i}^{k} \mathrm{d}_A(s_i, s_j)$.* MULTIPLE ALIGNMENT *with SP-score is the problem of finding a minimum alignment under the SP-score.*

The main theorem of this section is Theorem 1 which states that MULTIPLE ALIGNMENT with SP-score is NP-hard under all metrics. However, to convey the proof, a restricted case of the problem is shown NP-hard, Corollary 1. The full proof which requires a more detailed analysis is given thereafter. Some of the ideas in the construction[1] is by Bonizzoni and Vedova [BV01].

Theorem 1. *The decision version of* MULTIPLE ALIGNMENT *with SP-score is NP-complete for the binary alphabet under each metric. (Proof on page 6)*

Corollary 1. *The decision version of* MULTIPLE ALIGNMENT *with SP-score is NP-complete for the binary alphabet under the unit metric. (Proof on page 6)*

First the reduction is presented and on page 6 its correctness is proved. The reduction is from INDEPENDENT SET in 3-regular graphs: INDEPENDENT R3 SET. Let $V = \{v_1, \ldots, v_n\}$ be the vertices of the graph and $E \subseteq \{(v_i, v_j) : 1 \leq i < j \leq n\}$ the edges. From now on we reserve n for $|V|$, m for $|E|$, and c for the decision limit of the INDEPENDENT R3 SET instance.

Theorem 2. INDEPENDENT R3 SET *is NP-hard.*

Proof. In [PY91,BF95] INDEPENDENT SET is shown to be APX-complete even for graphs with degree bounded by 3. This result is simply extended to 3-regular graphs by joining vertices of degree 1 and 2 with gadgets as in Fig. 1. □

[1] Bonizzoni and Vedova made a reduction from VERTEX COVER. Although they conjectured that it could be improved to cover an *ultra metric*, they did not consider the unit metric or, more importantly, metrics in general.

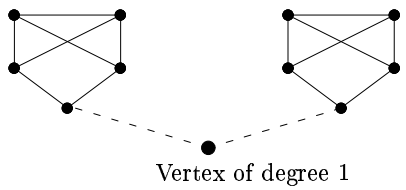

Fig. 1. Gadgets for showing that INDEPENDENT R3 SET is NP-hard. Each gadget has a maximal independent set of size exactly 2. In the picture a vertex of degree 1 is joined with two gadgets.

The decision version of the MULTIPLE ALIGNMENT instance has a set of strings $\mathcal{S} = \mathcal{T} \cup \mathcal{P} \cup \mathcal{C}$ and a decision limit K. It will be shown that there is an independent set of size c if and only if there is an alignment of $\mathcal{S}$ of cost K.

Let $b = 6nm^2$; a number chosen big enough to have a dominating effect in the alignment. In $\mathcal{S}$ there are b *template strings* $\mathcal{T}$, which force every optimum alignment to have a canonical structure, illustrated in Table 2. All template strings are identical to $T = (\mathtt{1}\mathtt{0}^b)^{n-1}\mathtt{1}$. Since they are identical, they are also aligned identically in every optimum alignment. In the canonical alignment the 1's in column $(b+1)(i-1)+1$ play the role of vertex v_i in the graph. For this reason we refer to the column as the *i'th vertex column*.

In $\mathcal{S}$ there are also $b = |\mathcal{P}|$ identical *pick strings* $P = \mathtt{1}^c$. For the same reason as for the template strings, these are aligned identically. Moreover, in any optimum alignment the 1's in the pick strings are aligned in those columns with the most 1's, which are the vertex columns. Thus the pick strings pick c of the n vertices to be part of the independent set (in Table 2 vertex v_2 is picked).

Table 2. A canonical alignment for the complete graph with four vertices. Since v_2 is the only vertex column containing three 1's from the constraint strings, the pick strings are aligned to pick v_2.

		v_1		v_2		v_3		v_4	
$\lvert\mathcal{T}\rvert = b$		1	0...0...	1	0...0...	1	0...0...	1	
		⋮	⋮ ⋮	⋮	⋮ ⋮	⋮	⋮ ⋮	⋮	
		1	0...0...	1	0...0...	1	0...0...	1	
$\lvert\mathcal{P}\rvert = b$				1					
				⋮					
				1					
C_{12}		0	0...10...	1	0...0...	0	0...0...	0	0...
C_{13}	0...	1	0...0...	0	0...10...	0	0...0...	0	
C_{14}		0	0...10...	0	0...0...	0	0...0...	1	0...
C_{23}	0...	0	0...0...	1	0...10...	0	0...0...	0	
C_{24}	0...	0	0...0...	1	0...0...	0	0...10...	0	
C_{34}	0...	0	0...0...	0	0...0...	1	0...10...	0	
	$4n$ 0's								$4n$ 0's

In $\mathcal{S}$ there are m *constraint strings* $\mathcal{C}$, one for each edge. The constraint string for edge $(v_i, v_j) \in E$ is the string

$$C_{ij} = \mathtt{0}^{4n}(\mathtt{0}\mathtt{0}^b)^{i-1}\mathtt{1}\mathtt{0}^{b-4n}(\mathtt{0}\mathtt{0}^b)^{j-i-1}\mathtt{1}\mathtt{0}^b(\mathtt{0}\mathtt{0}^b)^{n-j-1}\mathtt{0}\mathtt{0}^{4n}.$$

That is, C_{ij} has two 1's and is $4n$ longer than the template strings ($|C_{ij}| = 5n + b(n-1)$). The string C_{ij} can be constructed from T by setting all but the i'th and j'th vertex positions to 0, adding $4n$ 0's to the beginning and end and removing $4n$ 0's from in between the i'th and j'th vertex position. This

structure ensures that only one of the two 1's can be aligned in its associated vertex column. The 1 that is not aligned in its vertex column is not part of the independent set. (In Table 2 vertex v_1 is selected not to be part of the independent set by both the alignment of rows C_{12} and C_{14}.)

We now formally define the canononical alignment illustrated in Table 2.

Definition 2 (Canonical Alignment). *A canonical alignment is an alignment in which; (1) the template strings are aligned identically, (2) the pick strings are aligned identically and their 1's are in vertex columns, (3) each constraint string is aligned with the template strings so that $4n$ of its first or last 0's are matched with spaces and the rest with symbols of the template strings.*

We use $\mathrm{d}(\mathcal{T},\mathcal{P})$ to denote the sum of pairwise distances between rows (the alignment matrix is implicit) associated with strings in $\mathcal{T}$ and $\mathcal{P}$, i.e. $\mathrm{d}(\mathcal{T},\mathcal{P}) = \sum_{t\in\mathcal{T}}\sum_{p\in\mathcal{P}}\mathrm{d}(t,p)$. With this notation the total cost of the alignment is $\frac{1}{2}\mathrm{d}(\mathcal{S},\mathcal{S})$.

Since $\mathcal{S} = \mathcal{T}\cup\mathcal{P}\cup\mathcal{C}$ the cost of any alignment is

$$\frac{1}{2}\mathrm{d}(\mathcal{S},\mathcal{S}) = \frac{1}{2}\mathrm{d}(\mathcal{T},\mathcal{T}) + \mathrm{d}(\mathcal{T},\mathcal{P}) + \mathrm{d}(\mathcal{T},\mathcal{C}) + \frac{1}{2}\mathrm{d}(\mathcal{P},\mathcal{P}) + \mathrm{d}(\mathcal{P},\mathcal{C}) + \frac{1}{2}\mathrm{d}(\mathcal{C},\mathcal{C}).$$

Below we consider each pairwise distance to get the value of the decision limit K. That is, K is chosen so that there is an alignment of cost K if and only if there is an independent set of size c.

A1. $\mathrm{d}(\mathcal{T},\mathcal{T}) = 0$ in any optimum alignment and in any canonical alignment.

A2. $\mathrm{d}(\mathcal{T},\mathcal{P}) = (n-c+b(n-1))b^2$ in any optimum alignment and in any canonical alignment. All 0's and $n-c$ 1's are matched with spaces for each pair of template and pick strings.

A3. $\mathrm{d}(\mathcal{T},\mathcal{C}) \geq (4n+n)bm$ is the sum of minimum pairwise costs and also the cost in any canonical alignment. One of the two 1's in each constraint string is aligned in its associated vertex column. Thus $4n$ 0's are matched with spaces and n 1's are matched with 0's for each pair of template and constraint string.

A4. $\mathrm{d}(\mathcal{P},\mathcal{P}) = 0$ in any optimum alignment and in any canonical alignment.

A5. $\mathrm{d}(\mathcal{P},\mathcal{C}) \geq (5n+b(n-1))bm - c3b$ gives a lower bound for all canonical alignments. Remember, the graph is 3-regular and hence there can at most be three 1's from the constraint strings in the columns picked by the strings in $\mathcal{P}$. It is clear that if there is an independent set of size c then there also is a canonical alignment for which equality holds. Moreover, the minimum possible cost of $\mathrm{d}(\mathcal{P},\mathcal{C})$ in any alignment is $(5n+b(n-1))bm - 2bm$, which happens if both 1's of each constraint string are aligned with a 1 from the pick strings.

A6. $\frac{1}{2}\mathrm{d}(\mathcal{C},\mathcal{C}) < (8n+4)m(m-1)/2 < 5nm^2$ in any canonical alignment. In a canonical alignment at most $8n$ 0's are matched with spaces and at most 4 1's are matched with 0's for each pair of constraint strings.

Summing all these we get the value for the decision limit;

$$K = (n-c+b(n-1))b^2 + (4n+n)bm + (5n+b(n-1))bm - c3b + 5nm^2.$$

Note that the equalities in A1-A4 are achieved by every canonical alignment. Equality in A5 is achieved by every canonical alignment describing an independent set of size c. A6 provides an upper bound for the constraint strings in every canonical alignment.

Proof (Corollary 1). Clearly MULTIPLE ALIGNMENT $\in$ NP. Moreover, K was chosen in such a manner that if there is an independent set of size c then there is an alignment of cost K. Below the opposite is proved; if there is no independent set of size c then there is no alignment of cost K.

Assume that there is no independent set of size c. We first show that there can not be a canonical alignment with cost K. Consider a canonical alignment, since there is no independent set of size c, there has to be at least one column, selected by the pick strings, which does not contain three 1's from the constraint strings. Thus, the cost of $\mathrm{d}(\mathcal{P},\mathcal{C}) \geq (5n+b(n-1))bm - c3b + b$, i.e. compared to the lower bound in A5 there is an additional cost of at least b. Thereby the cost of the alignment is at least $K - 5nm^2 + b > K$. Notice that we have disregarded the cost of $\frac{1}{2}\mathrm{d}(\mathcal{C},\mathcal{C}) \geq 0$ and only considered A1-A5. Therefore, the cost of any canonical alignment is more than K.

The proof is completed by showing that there is no alignment of cost K. Recall that the equalities in A1, A2, and A4 are achieved by any optimum alignment and that the contribution of $\mathrm{d}(\mathcal{T},\mathcal{C})$ can be no less than in A3. Thus for an optimum alignment to have cost $\leq K$ the contribution of $\mathrm{d}(\mathcal{P},\mathcal{C})$ has to be made smaller. There are two cases in which this is possible and in each case we show that there exists a better canonical alignment, a contradiction. Essentially; aligning the constraint strings in a non-canonical fashion to improve $\mathrm{d}(\mathcal{P},\mathcal{C})$ is penalized by $\mathrm{d}(\mathcal{T},\mathcal{C})$.

(i) Assume that there are r constraint strings aligned such that one of their 1's is in an unassociated vertex column. That is, if C_{ij} is such a string then one of its 1's is in the k'th vertex column for $k \neq i,j$. Then the cost of $\mathrm{d}(\mathcal{P},\mathcal{C}) + \frac{1}{2}\mathrm{d}(\mathcal{C},\mathcal{C})$ can be at most $2br + 5nm^2$ **less than** in a canonical alignment. However, the cost of $\mathrm{d}(\mathcal{T},\mathcal{C})$ is at least $2(b-4n-1)br$ **more than** in a canonical alignment; due to a gap of $\geq b - 4n$. Since $2(b-4n-1)br > 2br + 5nm^2$ the alignment is not optimum.

(ii) Assume that there are r constraint strings aligned such that both of their 1's are in associated vertex columns. That is, if C_{ij} is such a string then the first of its 1's is in the i'th vertex column and the other in the j'th. As above $\mathrm{d}(\mathcal{P},\mathcal{C}) + \frac{1}{2}\mathrm{d}(\mathcal{C},\mathcal{C})$ is at most $2br + 5nm^2$ less than in a canonical alignment. However, the cost of $\mathrm{d}(\mathcal{T},\mathcal{C})$ becomes $\geq (8n-2)br$ more than in a canonical alignment; due to a gap of $4n$ positions. Since $(8n-2)br > 2br + 5nm^2$ the alignment is not optimum. □

2.1 Proof for All Metrics

Proof (Theorem 1). To generalize the proof of Corollary 1, it is sufficient to change the value of b, K, and the constraint strings such that they depend on the metric. We use the variables in Table 1 on page 3. Moreover, in addition to the metric propetries, we assume w.l.o.g. that $\beta \leq \gamma$, and that $1 \leq \alpha, \beta, \gamma$.

The important property of the constraint strings is that only one of the two 1's can end up in its associated vertex column. This was achieved by adding $4n$ 0's to each end and removing $4n$ 0's from the middle. Here the *seperation*, denoted by s, depends on the metric. Let $s = 5\alpha + 5\gamma + 1$, and modify b so that it continues to have a dominating effect, $b = 2(\alpha+\gamma) + (s\beta + 2\alpha)m^2 + s + \alpha + 1$.

The constraint strings then become

$$C_{ij} = \mathtt{0}^s(\mathtt{00}^b)^{i-1}\mathtt{10}^{b-s}(\mathtt{00}^b)^{j-i-1}\mathtt{10}^b(\mathtt{00}^b)^{n-j-1}\mathtt{00}^s,$$

thus $|C_{ij}| = s + (b+1)(n-1) + 1$.

To get the apropriate value of K we recalculate the pairwise costs (the cases are the same as for A1-A6):

B1. $\mathrm{d}(\mathcal{T},\mathcal{T}) = 0$ in any canonical and optimum alignment.

B2. $\mathrm{d}(\mathcal{T},\mathcal{P}) = (n-c)\gamma b^2 + b(n-1)\beta b^2$ in any canonical and optimum alignment.

B3. $\mathrm{d}(\mathcal{T},\mathcal{C}) \geq (s\beta + n\alpha)bm$ is the minimum pairwise cost and also the cost in any canonical alignment.

B4. $\mathrm{d}(\mathcal{P},\mathcal{P}) = 0$ in any canonical and optimum alignment.

B5. $\mathrm{d}(\mathcal{P},\mathcal{C}) \geq \Big(c\alpha + (s + b(n-1) + n - c - 2)\beta + 2\gamma\Big)bm - 3cb(\alpha+\gamma-\beta)$, in any canonical alignment with equality if there exists an independent set of size c. Similarly to (A5) for each pair (P, C_{ij}) the worst situation is if the c 1's in P are matched with 0's of C_{ij} and all remaining symbols of C_{ij} are matched with spaces. However if there is an independent set of size c then there can be c vertex columns with three 1's. Moreover, a lower bound on the cost in any alignment is $(c\alpha + (s + b(n-1) + n - c - 2)\beta + 2\gamma)bm - 2(\alpha+\gamma-\beta)bm$ (all 1's from $\mathcal{C}$ are aligned with 1's from $\mathcal{P}$).

B6. $\frac{1}{2}\mathrm{d}(\mathcal{C},\mathcal{C}) < (2s\beta + 4\alpha)m(m-1)/2 < (s\beta + 2\alpha)m^2$ in any canonical alignment.

Choose $K = (n-c)\gamma b^2 + b(n-1)\beta b^2 + (s\beta + n\alpha)bm + \Big(c\alpha + (s + b(n-1) + n - c - 2)\beta + 2\gamma\Big)bm - 3cb(\alpha+\gamma-\beta) + (s\beta + 2\alpha)m^2$.

As before we prove that there is an alignment of cost at most K if and only if there is an independent set of size c. Recall that K was chosen so that there is a canonical alignment of cost K if there is an independent set of size c, below the opposite direction is shown.

Assume that there is no independent set of size c. Then in any canonical alignment $\mathrm{d}(\mathcal{P},\mathcal{C}) \geq (c\alpha + (s + b(n-1) + n - c - 2)\beta + 2\gamma)bm - (3c-1)\cdot b(\alpha+\gamma-\beta)$, since one picked column does not contain three 1's. Moreover, since the upper bound in B6 is less than $b(\alpha+\gamma-\beta)$ the canonical alignment can not have cost $\leq K$.

The proof is completed by considering the same two cases as in the previous proof. Assume that there is an optimum alignment of cost $\leq K$:

(i) Moreover, assume that in the alignment r constraint strings are aligned such that each C_{ij} has a 1 in vertex column $k \neq i, j$. Then the cost of $\mathrm{d}(\mathcal{P},\mathcal{C}) + \frac{1}{2}\mathrm{d}(\mathcal{C},\mathcal{C})$ can be at most $2(\alpha+\gamma-\beta)br + (s\beta + 2\alpha)m^2$ less than in a canonical alignment. However, the cost of $\mathrm{d}(\mathcal{T},\mathcal{C})$ is at least $(2(b-s)\beta - 2\alpha)br$ more than in a canonical alignment. Since $\beta \geq 1$ and $b - s - \alpha > 2(\alpha+\gamma-\beta) + (s\beta+2\alpha)m^2$ there is a better canonical alignment contradicting optimality.

(ii) Moreover, assume that in the alignment r constraint strings are aligned such that each 1 is aligned in its associated vertex columns. Then the cost of $\mathrm{d}(\mathcal{P},\mathcal{C}) + \frac{1}{2}\mathrm{d}(\mathcal{C},\mathcal{C})$ can be at most $2(\alpha+\gamma-\beta)br + (s\beta + 2\alpha)m^2$ less than in a canonical alignment. However, the cost of $\mathrm{d}(\mathcal{T},\mathcal{C})$ is at least $(2s\beta - 2\alpha)br$ more than in a canonical alignment. Again since the canonical alignment is better we have a contradiction.

Thus there is no alignment of cost $\leq K$ and the proof is complete. □

3 STAR ALIGNMENT is NP-hard

Here STAR ALIGNMENT is proved NP-hard for all binary or larger alphabets under any metric. In the next section the construction, given here, is extended to cover TREE ALIGNMENT. In addition to the distance measure being the metric depicted in Table 1 on page 3 we assume w.l.o.g. that $\beta \leq \gamma$ and that $1 \leq \min(\alpha, \beta)$. Moreover, we first treat the case when $\alpha < \beta + \gamma$ and at then separately handle the special case $\alpha = \beta + \gamma$. From now on we assume that $\alpha < \beta + \gamma$.

The following lemma is essential to the construction. However, the proof is very simple and therefore left out.

Lemma 1 (Triangle inequality for strings). *If the symbol distance, μ, is a metric then the distance for strings defined by the cost of pairwise alignments w.r.t. μ satisfies the triangle inequality.*

Below we formally define the problem. The reader should notice the relation to Steiner trees and that the Steiner tree is restricted to be a star.

Definition 3 (STAR ALIGNMENT). *Given a set of strings $\mathcal{S}$,* STAR ALIGNMENT *is the problem of finding a string c (called a Steiner string) minimizing the sum of pairwise alignments between c and the strings in $\mathcal{S}$, i.e. $\sum_{s \in \mathcal{S}} \mathrm{d}(c, s)$.*

The reduction is from VERTEX COVER and the construction, given in Table 3 and Fig. 2, has three types of components; *base components*, *selection components*, and one *ground component*. A general outline of the construction is given below (see 1-3). Let $G = (V, E)$ be the graph of the VERTEX COVER instance, $n = |V|$, and $m = |E|$.

(1) There are r base components, definition below, ensuring that the optimum Steiner string is a string in which there are n vertex positions. A `1` in vertex position i corresponds to the i'th vertex being part of the vertex cover. That is, the optimum Steiner string corresponds to a subset V' of the vertices.

(2) There are m selection components, definition below, one for each edge. These components ensure that there for each edge (v_i, v_j) is a `1` in either the i'th or the j'th vertex position of the Steiner string. That is, the optimum Steiner string corresponds to a vertex cover V'.

(3) The ground component, definition below, minimizes the number of `1`'s in vertex positions. That is, the optimum Steiner string corresponds to a minimum vertex cover V'.

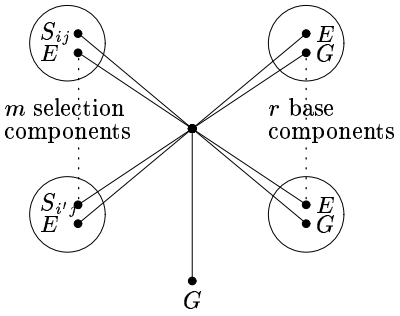

Fig. 2. Construction for STAR ALIGNMENT. See 1-3 above for an outline of the components. The strings are from Table 3.

Table 3. An overview of the strings in the construction (s is from Eq. 1). Each string is formally introduced below. The general idea though is that there is a one-to-one correspondance between the optimum vertex covers and the optimum Steiner strings which are base strings.

Name	Notation	Form	Length
Padding	P	$0^s1^s0^s$	$3s$
1-Block	B_1	$P1P$	$2\|P\|+1$
0-Block	B_0	$P0P$	$2\|P\|+1$
Vertex string	V_i	$B_0^{i-1}B_1B_0^{n-i}$	$\|B_0\|n$
Delimiter string	D	$1^{\|V_i\|}$	$\|B_0\|n$
Cover string	C	$(B_1\|B_0)^n$	$\|D\|$
Selection string	S_{ij}	V_iDV_j	$3\|D\|$
Enforcer string	E	$DD\ B_1^n\ DD$	$5\|D\|$
Ground string	G	$DD\ B_0^n\ DD$	$5\|D\|$
Base string		$DDCDD$	$5\|D\|$

Base Components The optimum Steiner string is *base string* and is of the form $DDCDD$; where C is a *cover string* and the D's are *delimiter strings*, see Table 3. A cover string consists of n consecutive blocks, each being B_0 or B_1. If the i'th block is B_1 then this corresponds to the i'th vertex being part of the cover. In $B_0 = P0P$ and $B_1 = P1P$ the 0 and 1, respectively, are in the so called *vertex position*. For the construction to work for all metrics the size of the paddings, denoted by P, depends on the metric. Let

$$s \geq (n+1) \cdot \left\lceil \frac{\max(\alpha,\gamma)}{\min(\alpha,\beta)} \right\rceil, \tag{1}$$

and let $P = 0^s1^s0^s$. The delimiter strings consists of $|C|$ 1's: $D = 1^{|C|}$.

In the construction there are sufficiently many special pairs of base strings, called *base components*, to ensure that the optimum Steiner string is a base string. The string pair in a base component consists of one *enforcer string* and one *ground string*, defined by $E = DDB_1^nDD$ and $G = DDB_0^nDD$, respectively. The important properties of base components are given in the lemma below and follow from lemmas 5 and 6 in Sect. 3.1.

Lemma 2 (Base components). (1) *The only optimum alignment of an enforcer string and a ground string is the direct match. That is, in the direct match the i'th symbol of E is aligned with the i'th symbol of G, thus* $\mathrm{d}(E,G) = n\alpha$.

(2) *If* $\mathrm{d}(E,x) + \mathrm{d}(G,x) < \mathrm{d}(E,G) + \min(\alpha, \beta, \beta+\gamma-\alpha)$ *and* $\alpha < \beta+\gamma$, *then x is a base string.*

Selection Components Each edge (v_i, v_j) in the vertex cover instance is represented by a selection component. A selection component for the edge (v_i, v_j) consists of two strings, one enforcer string E and one *selection string* $S_{ij} = V_iDV_j$, where $V_i = B_0^{i-1}B_1B_0^{n-i}$. The important properties of the component are given in the lemma below and follow from lemmas 5 and 7 in Sect. 3.1. According to the lemma $\mathrm{d}(E,x) + \mathrm{d}(S_{ij},x)$ is minimized if and only if x is a base string in which block i or j is B_1. Correspondingly, for each edge (v_i, v_j) in the VERTEX COVER instance vertex v_i or v_j have to be part of the cover.

Lemma 3 (Selection component). (1) *The cost of an optimum alignment of an enforcer string and a selection string is* $\mathrm{d}(E,S_{ij}) = 2|D|\gamma + (4ns + 2n - 2)\alpha$.

(2) *If x is a base string and* $\mathrm{d}(E,x)+\mathrm{d}(S_{ij},x)=\mathrm{d}(E,S_{ij})$*, then the i'th or j'th block of x is B_1. Moreover, if x is a base string in which both block i and j are B_0 then* $\mathrm{d}(E,x)+\mathrm{d}(S_{ij},x)=\mathrm{d}(E,S_{ij})+2\alpha$.

Ground Component The ground component is simply one single ground string. For each B_1 block in the Steiner string the ground component adds an additional cost to the alignment. In other words, the fewer vertices that are selected the smaller the cost. The lemma below is a direct consequence of Lemma 5 in Sect. 3.1.

Lemma 4. *If x is a base string and z the number of B_1 blocks in x, then* $\mathrm{d}(G,x)=z\alpha$.

The Completeness Proof

Theorem 3. *The decision version of* STAR ALTHEMENT *is NP-complete for the binary alphabet under all metrics in which $\alpha<\beta+\gamma$. (Remark: The proof for $\alpha=\beta+\gamma$ is on page 14)*

Proof. Clearly STAR ALIGNMENT $\in$ NP. As Fig. 2 indicates, the alignment instance has $2m+2r+1$ strings; one selection component per edge, r base components, and one ground component. Let

$$r=n\left\lceil\frac{\max(\alpha,\gamma)}{\min(\alpha,\beta,\beta+\gamma-\alpha)}\right\rceil \tag{2}$$

and the decision limit

$$K=m\cdot\Big(2|D|\gamma+(4ns+2n-2)\alpha\Big)+r\cdot\alpha n+\alpha c,$$

where c is the decision limit of the VERTEX COVER instance.

We now show that an optimum Steiner string corresponds to a minimum vertex cover, and vice versa. If the Steiner string is a base string corresponding to the cover V' then the cost of the star alignment is

$$\underbrace{m\cdot\mathtt{d}(E,S_{ij})}_{\text{selection comp., Lemma 3}}+\underbrace{r\cdot\mathtt{d}(E,G)}_{\text{base comp., Lemma 2}}+\underbrace{\alpha|V'|}_{\text{ground}}, \tag{3}$$

which is strictly decreasing in the size of the cover, $|V'|$. Clearly, by lemmas 2, 3, and 4, if there exists a cover of size c then there is alignment of cost K. In particular, there is always an alignment of the same cost as above with $|V'|=n-1$.

Let s^* be an optimum Steiner string.

(i) *Assume that s^* is not a base string.* By Lemma 2 the cost of the base components is $r(\mathrm{d}(E,s^*)+\mathrm{d}(G,s^*))\geq r\alpha n+n\max(\alpha,\gamma)$. Moreover, the cost of the selection components is at least $m\cdot\mathtt{d}(E,S_{ij})$. Since this is more than the upper limit of (3) this contradicts the optimality of s^*.

(ii) *Assume that s^* does not correspond to a cover.* Then there is a selection string S_{ij} such that both block i and j of s^* is B_0. By Lemmas 2, 3, and 4, we

could exchange block i or j for B_1 and thereby improve the cost, contradicting the optimality of s^*.

Since (3) is strictly decreasing in the size of the cover we have shown that the optimum Steiner string corresponds to a minimum vertex cover. Clearly, the reduction is polynomial. Hence STAR ALIGNMENT is NP-hard for the binary alphabet under metrics in which $\alpha < \beta + \gamma$. □

3.1 Technical Lemmas for STAR ALIGNMENT

Here the technical lemmas needed above are given.

Lemma 5. *Let a and b be two base strings, and z the number of positions in which the two strings disagree. Then the only pairwise alignment of a and b with cost $< z\alpha + \min(\alpha, \beta, \beta + \gamma - \alpha) = u$ is the direct match. That is, $\mathtt{d}(a, b) = z\alpha$ is the hamming distance.*

Proof. Note that the direct match has cost $z\alpha \leq n\alpha$ and that $s \cdot \min(\alpha, \beta) \geq (n+1)\max(\alpha, \gamma) \geq (n+1)\alpha$. Thus in any alignment of cost less than u there can not be s mismatches. It should be clear that since delimiter strings only consist of 1's the delimiter strings of a and b are directly matched. Thus the delimiter strings can be disregarded in the rest of the proof.

We argue by induction over the number of blocks in the cover strings of a and b, that the strings are directly matched. The induction hypothesis is that for two strings a' and $b' \in \{B_0, B_1\}^i$, $1 \leq i \leq n$, the only alignment of cost $< z'\alpha + \min(\alpha, \beta, \beta + \gamma - \alpha) = u'$, is the direct match. As before $z' \leq i$ is the number of positions in which a' and b' disagree.

The basis step, $i = 1$, is simple. That is, that the only alignment of B_1 and B_0 of cost $< \alpha + \min(\alpha, \beta, \beta + \gamma - \alpha)$ is the direct match.

Inductively let the hypothesis be true $\forall i \leq k - 1 \leq n - 1$. Recall that the paddings in the block strings are $P = \mathtt{0}^s\mathtt{1}^s\mathtt{0}^s$. Since there are k blocks there are also $k - 1$ substrings $\mathtt{1}^s\mathtt{0}^s\mathtt{0}^s\mathtt{1}^s$ in the strings, i.e. the left and right part of two paddings lying next to each other. We show that in any alignment of cost u' there is such a substring in the same position of a' and b' which is directly matched. By induction this implies that a' and b' are directly matched.

Now consider any padding in the same position of a' and b'. Since there can not be more than s mismatches, at least $s/2$ of the 1's from a padding of a' have to line up with 1's from the padding in the same position of b', see Table 4. As described in the table there are three possible types of alignments in which all 1's do not line up. First we conclude that the first two types of alignments can not be of cost u'. Thereafter we use the third type to show that there is a substring, as described above, which is directly matched.

Let $j < s$ of the 1's line up for some padding and let x and y be as in the table. As shown the first two types of alignments can not be of cost $\min(\alpha, \beta, \beta + \gamma - \alpha)$ from the optimum:

(i) *Both x and y are* -. By matching the $j + 1$ of the right most 1's with the $j + 1$ left most 1's the alignment is improved by 2γ.

(ii) *x is* `0` *and y -* *(and vice versa).* By matching the $j+1$ of the right most `1`'s with the $j+1$ left most `1`'s and aligning x with a `-` the alignment is improved by $\gamma + \alpha - \beta \geq \alpha$.

However, if the alignment is of the third type, i.e. both x and y are `0`, then the mismatch of x and y results in a cost of 2α. As described above there are $k-1$ pairs of paddings PP in the strings. If in the alignment both paddings in such a pair are aligned according to the third type the contribution is 4α. Since $4(k-1)\alpha > z'\alpha + \alpha$ it can not be that all pairs are of that type. Since (i) an (ii) are not possible there is a pair PP in which all `1`'s from one padding line up with all `1`'s from the same padding in the other string. Moreover, since there are $2s$ `0`'s in both strings in between the `1`'s these have to be directly matched which implies that the `1`'s from the other padding either are lined up or of type (i) or (ii). Since (i) and (ii) can not occur we have shown that there is a substring $\mathtt{1}^s\mathtt{0}^s\mathtt{0}^s\mathtt{1}^s$ of a' and b' which is directly matched. □

Table 4. Part of an alignment of the two base strings a' and b'. In the table $i \geq s/2$ `1`'s of the paddings line up. There are three cases that can occur; both x and y are `-`, one of x and y is a `-` and the other a `0`, both x and y are `0`.

$$\begin{array}{c|c|c|c|c} \ldots \mathtt{1}^{s-i-1} & \mathtt{1} & \mathtt{1}^i & x & \ldots \\ \ldots & y & \mathtt{1}^i & \mathtt{1} & \mathtt{1}^{s-i-1} \ldots \end{array}$$

Lemma 6. *Let a and b be two base strings. Moreover, let x be a string such that $\mathtt{d}(a,x) + \mathtt{d}(x,b) < \mathtt{d}(a,b) + \min(\alpha, \beta, \beta + \gamma - \alpha) = u$. Then x is a base string.*

Proof. The symbol in each position of x has to agree with the symbol in the same position of either a or b. If this was not the case then, by the triangle inequality, the alignments of (a,x) and (x,b) induce an alignment of (a,b), other than the direct match, of cost $< u$. According to Lemma 5 this is not possible. Therefore x has to be a base string. □

Table 5. The only two types of optimum pairwise alignments for a selection string and a base string.

	D	D	C	D	D
v_j is in the cover	V_i	D	V_j		
v_i is in the cover			V_i	D	V_j

Lemma 7 (Selection string and base string). *In any optimum alignment of a selection string $S_{ij} = V_i D V_j$ and a base string $DDCDD$ either the blocks of V_i or V_j are optimally aligned with the blocks of C (Table 5).*

Proof. The cost of such an alignment is $2|D|\gamma + (n(4s+1)-1)\alpha + z\alpha$, where z is the number of blocks in C that differs from V_i or V_j. Since $z \leq n$ the cost

of the alignment is less than $2|D|\gamma + |D|\alpha$. Below all other types of alignments are shown to be worse.

(i) *It is not possible for substrings of both V_i and V_j to be aligned with the cover string in an optimum alignment.* As shown in Table 6, r of the rightmost symbols of V_i and l of the leftmost symbols of V_j are aligned with the cover string. Then the cost from only the right and left part of the alignment[1] is at least

$$(2|D| + r + l)\gamma + \frac{2|D| - r - l}{2}\alpha \geq 2|D|\gamma + |D|\alpha,$$

where the inequality is given by $\alpha \leq 2\gamma$.

Table 6. Parts from both vertex strings are aligned with the cover string.

D D		C		D D
$V_i[1\ldots(\lvert D\rvert - r)]$	$V_i[(\lvert D\rvert - r)\ldots\lvert D\rvert]$ D		$V_j[1\ldots(\lvert D\rvert - l - 1)]$	$V_j[l+1\ldots\lvert D\rvert]$

(ii) *The delimiter string of S_{ij} is matched with a delimiter string of $DDCDD$ in every optimum alignment.* Trivial since the cover string has `0`'s to the right and left.

(iii) *The only optimum alignment of $DDCDD$ and V_x is the alignment in which C and V_x are directly matched.* Since both V_x and C have s `0`'s to the right and left no part of V_x is aligned with delimiter strings. Moreover, by Lemma 5 the optimum alignment of C and V_x is the direct match. □

3.2 STAR ALIGNMENT is NP-hard when $\alpha = \beta + \gamma$

Above only the metrics in which $\alpha < \beta + \gamma$ were handled, here the special case $\alpha = \beta + \gamma$ is handled. The arising problem is that one substitution can be explained by two insertions. As a result the base components can not force the optimum Steiner string to be a base string. Instead the base components force the optimum Steiner string to be a *semi-base string*. A semi-base string is like a normal base string except that the blocks of the cover string can be on any of the forms: B_1, B_0, PP, $P\mathtt{10}P$, and $P\mathtt{01}P$.

The new lemmas given below are, as their predecessors, consequences of Observation 1 and Lemma 1. Here the observation plays the role of the techincal lemmas above: lemmas 5 and 7. It is a very simple matter to modify the proves of the technical lemmas to prove the observation. Moreover, the proves of the lemmas below are also very simple. Therefore these proofs have been left out.

Observation 1 ($\alpha = \beta + \gamma$). *(1) Let r_1 and r_2 be two semi-base strings. Then the only alignments of cost $< \mathtt{d}(r_1, r_2) + \beta$ are alignments in which all substrings $\mathtt{1}^s$ of r_1 are aligned with the same $\mathtt{1}^s$ substring in r_2.*

(2) Let S_{ij} be a selection string. Then the only alingments of cost $\mathtt{d}(r_1, S_{ij})$ are alignments in which the cover string of r_1 is aligned with one of the vertex strings of S_{ij} such that the $\mathtt{1}^s$ substrings are aligned.

[1] $\mathtt{d}(DD, V[1\ldots(|D|-r)]) + \mathtt{d}(DD, V[l+1\ldots|D|)])$

Lemma 8 (Base components (cont. Lemma 2)). *(1) The cost of an optimum alignment of an enforcer string and a ground string is* $\mathrm{d}(E,G) = n\alpha$.

(2) If $\alpha = \beta + \gamma$, *then the only strings* x *for which* $\mathrm{d}(E,x) + \mathrm{d}(G,x) < \mathrm{d}(E,G) + \beta$ *are* ***semi*** *base strings.*

Proof. (1) is a direct consequence of the observation. (2) follows from the observation, Lemma 1, and the fact that G and E only have blocks that are B_0 or B_1. □

Lemma 9 (Selection components (cont. Lemma 3)). *(1) The cost of an optimum alignment of an enforcer string and a selection string is* $\mathrm{d}(E,S_{ij}) = 2|D|\gamma + (4ns + 2n - 2)\alpha$. *(unchanged)*

(2)If $x = DDCDD$ *is a* ***semi*** *base string in which the* i*'th or* j*'th block is* B_1 *then* $\mathrm{d}(E,x) + \mathrm{d}(S_{ij},x) = \mathrm{d}(E,S_{ij})$. *Moreover, if* ***neither*** *block* i *and* j *is* B_1 *then* $\mathrm{d}(E,x) + \mathrm{d}(S_{ij},x) \geq \mathrm{d}(E,S_{ij}) + z$, *where*

$$z = \begin{cases} 2\alpha \text{ if both blocks are } B_0 \\ 2\beta \text{ if one block is } P10P \text{ or } P01P \\ 2\gamma \text{ if one block is } B_0 \text{ and the other } PP \end{cases}$$

Lemma 10. *The cost of an optimum alignment of a ground string,* G, *and a* ***semi*** *base string,* x, *is* $\mathrm{d}(G,x) = z_1\alpha + z_2\beta + z_3\gamma$, *where* z_1 *is the number of* B_1 *blocks,* z_2 PP *blocks,* z_3 *blocks* $P10P$ *or* $P01P$, *in* x.

Theorem 4. *The decision version of* STAR ALIGNMENT *is NP-complete for the binary alphabet under all metrics in which* $\alpha = \beta + \gamma$.

Proof. The construction is as in Theorem 3 except for r which is

$$r = n\left\lceil \frac{\alpha}{\beta} \right\rceil.$$

We show that an optimum Steiner string corresponds to a minimum vertex cover, and vice versa.

Let s^* be the optimum Steiner string.

(i) *Then* s^* *is a* ***semi*** *base string.* Otherwise by Lemma 8 the cost of the base components is $r(\mathrm{d}(E,s^*) + \mathrm{d}(G,s^*)) \geq r\alpha n + n\alpha$. Moreover, the cost of the selection components is at least as in Equation 3. Therefore, s^* is a semi-base string.

(ii) *For each selection string* S_{ij} *either block* i *or* j *in* s^* *is* B_1. If for some selection string S_{ij} neither block i or j of s^* is B_1. Then block i of s^* is on either of the forms B_0, PP, $P10P$, or $P01P$. By the Lemmas 8, 9, and 10, the i'th block can be exchanged for B_1 and thus the alignment is improved by at least β.

(iii) s^* *is a base string.* If s^* has a block on form PP, $P01P$, or $P10P$, then exchanging that block for a B_0 decreases the contribution of the ground component and leaves the contribution of the other components unchanged.

Since Equation 3 is strictly decreasing in the size of the cover we have shown that the optimum Steiner string corresponds to a minimum vertex cover. Clearly the reduction is polynomial. Hence STAR ALIGNMENT is NP-hard for the binary alphabet under all metrics in which $\alpha = \beta + \gamma$. □

3.3 All Alphabets and All Metrics

As a Corollary of theorems 3 and 4 we get the comprehensive result below.

Corollary 2. *The decision version of* STAR ALIGNMENT *is NP-complete for all binary or larger alphabets under all metrics.*

Proof. The problem that could occur in a larger alphabet is that the optimum Steiner string might contain letters different from `1` and `0`. However, it is not difficult to see that any different letter could be exchanged for a `1` or a `0` without effecting the cost of the alignment. □

4 TREE ALIGNMENT is NP-hard

A *full labeling* of a tree is a function assigning strings to all vertices of the tree. Similarly a *leaf labeling* is a function assigning labels to the leafs. The *length* of an edge (u, v) in a labeled tree is the cost of the minimum pairwise alignment of the labels at u and v.

Definition 4 (TREE ALIGNMENT). *For a leaf labeled tree of bounded degree a* tree alignment *is a full labeling of the tree, such that the leafs are labeled according to the leaf labeling. Given a leaf labeled tree,* TREE ALIGNMENT[1] *is the problem of finding a minimum cost full labeling, where the cost is the sum of all edge lengths.*

The construction for TREE ALIGNMENT (Fig. 3) is very similar to that of STAR ALIGNMENT. However, we need two additional technical lemmas to handle the tree structure, these are given after the proof. Moreover, just as with STAR ALIGNMENT we first assume that $\alpha < \beta + \gamma$ and thereafter handle the case $\alpha = \beta + \gamma$.

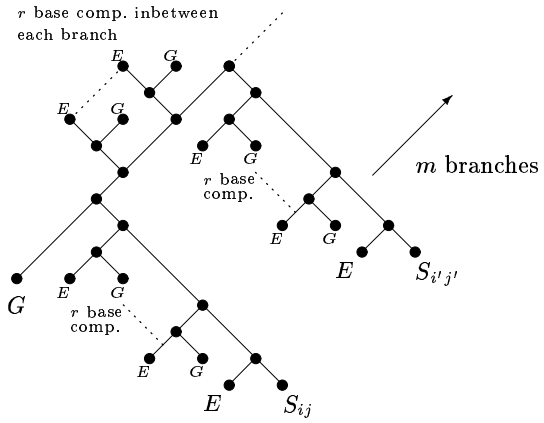

Fig. 3. Construction for TREE ALIGNMENT. There are m branches (one for each edge), each branch has r base components and one selection component. Moreover, in between each branch there are r base components.

[1] "TREE ALIGNMENT" is sometimes used to refer to the problem of finding the structure of the tree in addition to the optimal alignment. This problem is shown to be APX-complete for an ultra metric in [WJ94], the proof is trivial to extend to all metrics. However, for notional convenience we use the term "TREE ALIGNMENT" for the more restricted case.

Theorem 5. *The decision version of* TREE ALIGNMENT *is NP-complete for the binary alphabet under all metrics in which* $\alpha < \beta + \gamma$.

Proof. Clearly TREE ALIGNMENT $\in$ NP. As Fig. 3 indicates, the alignment instance consists of a tree with $2m + 2r(2m-1) + 1$ leaves. Let r be as in (2) in the previous proof. For each edge, in the vertex cover instance, there is a branch with one selection component and r base components (see the figure for details). Moreover, each such branch is separated by r base components. As in the proof for STAR ALIGNMENT there is also one ground component. Moreover, the alignment instance has a decision limit

$$K = m\Big(2|D|\gamma + (4ns + 2n - 2)\alpha\Big) + r(2m-1)\alpha n + \alpha c,$$

where c is the decision limit of the VERTEX COVER instance.

If x is a base string corresponding to a cover V', then the cost of the alignment in which all internal vertices are labeled with x is

$$\underbrace{m \cdot \mathrm{d}(E, S_{ij})}_{\text{selection comp.}} + \underbrace{r(2m-1) \cdot \mathtt{d}(E, G)}_{\text{base comp.}} + \underbrace{\alpha |V'|}_{\text{ground}}. \tag{4}$$

If $|V'| \leq c$ then the alignment has cost $\leq K$ (follows from lemmas 2,3, and 4).

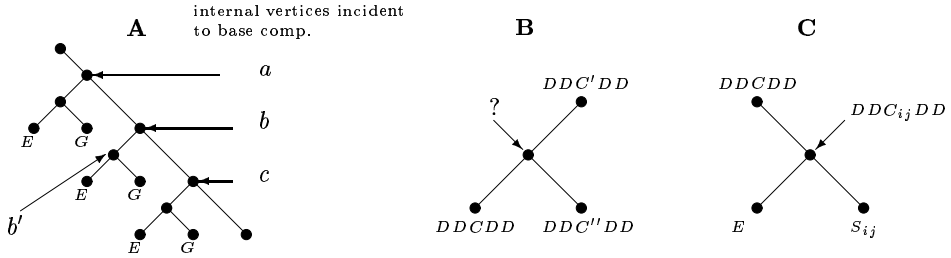

Fig. 4. (A) Part of a branch in the construction. All vertices incident to base components are labeled with the same base string. (B) The vertex connecting a branch is labeled with a base string. (C) The vertex in a selection component is labeled with a base string.

We now show that there is a vertex cover of size c if there is an alignment of cost at most K. This is done by proving that there is an optimum alignment in which each vertex is labeled with the same base string x, corresponding to a cover. Since (4) is strictly decreasing in $|V'|$, x corresponds to a minimum cover. The proof is in two steps: (i) in every optimum alignment all labels are base strings, (ii) using (i) we show that there is an optimum labeling with a base string corresponding to a cover.

(i) *Consider an optimum alignment. We show that if not all internal vertices are labeled with base strings then the alignment is not optimum.* According to Lemma 2, if a vertex incident (e.g. vertex c in Fig. 4A) to a base component is not labeled with a base string, then the cost of that base component is at least $\mathrm{d}(E,G) + \min(\alpha, \beta, \beta + \gamma - \alpha)$. There are r base components on each branch. Thus at least one of the vertices incident to the base components is labeled with a base string. Otherwise, since the contribution of these components is

$$r(\mathrm{d}(E,G) + \min(\alpha, \beta, \beta + \gamma - \alpha)) > r\mathrm{d}(E,G) + n\alpha,$$

the alignment is not optimum, i.e. $n\alpha$ more in (4).

Let the vertices a,b,c, and b', in Fig. 4A be labeled with the strings s_a,s_b,s_c, and $s_{b'}$, respectively. We show that if s_c is a base string then so is s_b, which inductively implies the claim. Assume that s_b is not a base string. Then by exchanging the label at both b and b' for s_c we achieve a better alignment, a contradiction. This is a direct consequence of the triangle inequality (Lemma 1) and the properties of the base component (Lemma 2).

It remains to show that the vertices connecting the branches and the vertices connected with the leafs of a selection component are labeled with base strings, i.e. the vertices in Fig 4B and 4C. This is shown in lemmas 11 and 12 below.

(ii) *We show that there is an optimum alignment in which all internal vertices are labeled with a base string x, corresponding to a cover.* Consider an optimum alignment in which all labels are base strings. Then x is created so that $\forall i \in [1, n]$ if the i'th block is B_1 in any of the labels in the optimum alignment then the i'th block in x is also B_1. The labeling with x is still optimum because: (1) base strings are aligned symbol by symbol (Lemma 5), (2) triangle inequality holds for symbols, and (3) only the cost at the ground is effected by B_1 blocks. Thus with the labeling with x all missmatches occuring in vertex positions have been transfered to the edge incident to the ground. Moreover, by Lemma 12, for each selection string S_{ij} the i'th or j'th block of x is B_1, hence x corresponds to a cover.

Clearly, the reduction is polynomial. Consequently, TREE ALIGNMENT is NP-hard for the binary alphabet under any metric in which $\alpha < \beta + \gamma$. □

Lemma 11. *Let r_1, r_2, and r_3, be three base strings (Fig. 4B). Then there is a base string x for which*

$$\mathrm{d}(r_i, x) + \mathrm{d}(r_j, x) = \mathrm{d}(r_i, r_j),\ 1 \leq i < j \leq 3. \tag{5}$$

Moreover, any x satisfying (5) is a base string.

Proof. The three equations are satisfied if for $i \in [1, |r_1|]$ the symbol in position i in x is given by the majority symbol in position i of the base strings. For example, if the i'th symbol of at least two of the strings is 1 then the i'th symbol of x is 1. Moreover, by Lemma 6 it is indeed the case that x is a base string. □

Lemma 12 (Selection component). *For an enforcer string E, a selection string S_{ij}, and a base string r (Fig. 4C), there is a string x for which: (1)* $\mathrm{d}(E, x) + \mathrm{d}(r, x) = \mathrm{d}(E, r)$, *(2)* $\mathrm{d}(E, x) + \mathrm{d}(S_{ij}, x) = \mathrm{d}(E, S_{ij})$, *(3)* $\mathrm{d}(r, x) + \mathrm{d}(S_{ij}, x) = \mathrm{d}(r, S_{ij})$. *Moreover, any x satisfying the equations is a base string in which block i or j is B_1.*

Proof. By Lemma 6, the first equation is only satisfied if x is a base string. By Lemma 3, the second equation is only satisfied if either the i'th or j'th block of x is B_1. Moreover, by Lemma 7, the third equation holds if x is a base string in which block i or j is B_1. □

4.1 TREE ALIGNMENT is NP-hard when $\alpha = \beta + \gamma$

The special case when $\alpha = \beta+\gamma$ becomes easier when using the base component depicted in Fig. 5. The strings denoted by M_x are on the form $DD\ (PxP)^n\ DD$. We first show some simple properties for these strings and then show the equivalent of Lemma 2 for the new component.

Lemma 13 ($\alpha = \beta + \gamma$). *The optimum alignment of M_- and M_{01} has cost $\mathrm{d}(M_-, M_{01}) = n\alpha$. The only strings x for which $\mathrm{d}(M_-, x)+\mathrm{d}(x, M_{01}) < n\alpha+\beta$, are base strings in which the blocks are either B_1, B_0, PP, or $P01P$.*

Proof. Both properties follow from Observation 1, Lemma 1, and the fact that M_- only contains blocks PP and M_{01} blocks $P01P$. □

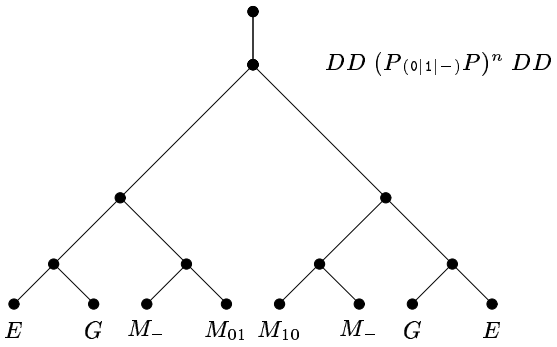

Fig. 5. The base component when $\alpha = \gamma+\beta$. Unless the root of the component is a space-base string, the cost of the component is not optimum. The strings M_x are strings $DD(PxP)^nDD$.

We refer to semi-base strings in which no block is $P01P$ or $P10P$ for *space-base strings.*

Lemma 14 ($\alpha = \beta + \gamma$). *For the base component in Fig. 5 the following properties hold:* **1.** *The minimum cost of the component is $4n\alpha$.*

2. *The only alignments of the component in which the cost $< 4n\alpha + \beta$ are alignments in which the root is labeled with a space-base string, i.e. $DD\ (P_{(0|1|-)}P)^n\ DD$.*

Proof. By Lemmas 8 and 13

$$2 \cdot \mathrm{d}(E, G) + \mathrm{d}(M_-, M_{01}) + \mathrm{d}(M_-, M_{01}) = 4n\alpha.$$

By the same lemmas it is also easy to see that the second property only holds if all internal vertices are labeled with the same space-base string. □

Contrary to the base component when $\alpha < \beta + \gamma$ the new situation allows for base strings in which blocks are PP. Therefore we need to modify lemmas 11 and 12, to handle space-base strings.

Lemma 15 ($\alpha = \beta + \gamma$). *Let r_1, r_2, and r_3, be three space-base strings. Then there is a space-base string x for which*

$$\mathrm{d}(r_i, x) + \mathrm{d}(r_j, x) = \mathrm{d}(r_i, r_j),\ i \neq j.$$

Moreover, the three statements are only satisfied if x is a space-base string.

Proof. By choosing each block in x according to the majority vote of the three strings. For example if block i is B_1 in at least two of the three strings then block i of x is set to B_1. However, if a majority does not exist, i.e. there is one B_1, one B_0, and one PP, then that block in x is set to PP. Moreover, by Observation 1, Lemma 1, and the fact that $r_{1,2,3}$ are space-base strings x has to be a space-base string.

Lemma 16 (Selection component $\alpha = \beta + \gamma$). *For an enforcer string E, a selection string S_{ij}, and a space-base string r, there is a string x for which the three equations in Lemma 12 hold. Moreover, the equations are only satisfied if x is a space-base string with either the i'th or j'th block being B_1.*

Proof. Since each block of E is B_1 and since r is a space-base string x has to be a semi-base string (Observation 1 and Lemma 1). Knowing that x is a semi-base string we note that each block of x is aligned with a block in S_{ij} (again by Observation 1). Thus each block of x is aligned with a B_1 block from E, a B_0, B_1 or PP block from r, and a B_0 or a B_1 block from S_{ij}. Therefore, by the same arguments as in the previous proof each block of x is either B_0, B_1, or PP, and either block i or j is B_1. □

Theorem 6. *The decision version of* TREE ALIGNMENT *is NP-complete for the binary alphabet under all metrics in which $\alpha = \beta + \gamma$.*

Proof. The proof is almost identical to that when $\alpha < \beta + \gamma$. The only parts of the construction that need changing are r and the decision limit:

$$r = n\left\lceil \frac{\alpha}{\beta} \right\rceil,$$

$$K = \underbrace{m\Big(2|D|\gamma + (n(|P| + 2s) + 2n - 2)\alpha\Big)}_{\text{selection comp.}} + \underbrace{r(2m-1)4n\alpha}_{\text{base comp.}} + \underbrace{\alpha c}_{\text{ground}}.$$

As before there is an alignment of cost K if there is a vertex cover of size c. We continue and show the other direction of the reduction. That is, there is a vertex cover of size c if there is an alignment of cost K.

(i) *In every optimum alignment all vertices are labeled with* ***space-base*** *strings.* This follows by the same arguments as in Theorem 5 and by the equivalent lemmas, lemmas 15 and 16.

(ii) *We show that there is an optimum alignment in which all internal vertices are labeled with a base string x, corresponding to a cover.* Consider an optimum alignment in which all labels are space-base strings. Then x is created so that $\forall i \in [1, n]$ if the i'th block is B_1 in any of the labels in the optimum alignment then the i'th block in x is also B_1. The labeling with x is still optimum because: (1) space-base strings are aligned block by block (Observation 1), (2) triangle inequality holds for strings (Lemma 1), and (3) only the cost at the ground is effected by B_1 blocks. Moreover, by the same arguments all other blocks of x can be exchanged for B_0. Thus with the labeling with x all missmatches occuring in vertex positions have been transfered to the edge incident to the ground. Moreover, by Lemma 16, for each selection string S_{ij} the i'th or j'th block of x is B_1, hence x corresponds to a cover.

As before the reduction is polynomial and TREE ALIGNMENT is NP-hard for the binary alphabet when $\alpha = \beta + \gamma$. □

4.2 All Alphabets and All Metrics

From Theorem 5 and Theorem 6 we get the corollary bellow.

Corollary 3. *The decision version of* TREE ALIGNMENT *is NP-complete for all binary or larger alphabets under all metrics.*

Proof. Exactly as the proof of Corollary 2. □

4.3 Distinct Leaf Labels

In the definition of TREE ALIGNMENT (Definition 4) there was no restriction that the leaf labels should be distinct. This is however not a very big problem. To modify the construction such that all leaf labels are distinct we simply add a huge unique prefix to each leaf label.

Let

$$A_x = \mathtt{1}^{5|D|}\ \mathtt{0}^{5|D|} x \mathtt{0}^{5|D|}\ \mathtt{1}^{5|D|}$$
$$\text{Prefix}_i = A_0^{i-1} A_1 A_0^{|L|-i}.$$

and order all leafs $1 \ldots L = (2m + 2r(2m-1) + 1)$. Then add Prefix_i to the label at leaf i. Clearly, all internal labels will have to have the prefix $A_0^{|L|}$.

5 CONSENSUS PATTERNS and SUBSTRING PARSIMONY are NP-hard

In this section CONSENSUS PATTERNS and SUBSTRING PARSIMONY are shown NP-hard. However, CONSENSUS PATTERNS has earlier been proved NP-hard [LMW01,BST02,Aku98] and W[1]-hard in [FGN02].

Definition 5 (CONSENSUS PATTERNS). *Given a set of strings $\mathcal{S} = \{s_1, \ldots, s_n\}$ and a number k.* CONSENSUS PATTERNS *is the problem of finding a string c of length k and substrings t_i of s_i of length k such that the the sum of Hamming distances are minimized, $\sum_{i=1}^{n} \mathtt{d}_H(c, t_i)$.*

Definition 6 (SUBSTRING PARSIMONY). *For a leaf labeled tree of bounded degree and a number k, a* substring parsimony alignment *is a full labeling of the tree such that all internal vertices are labeled with strings of length k and each leaf with a substring of length k of the original leaf label.* SUBSTRING PARSIMONY *is the problem of finding such a labeling of minimum cost. The cost of the labeling is the sum of all edge lengths, where the length of an edge is the Hamming distance between the two labels.*

Theorem 7. CONSENSUS PATTERNS *is NP-hard.*

Proof. The reduction is from VERTEX COVER. Let n and m denote the number of vertices and edges, respectively, then the CONSENSUS PATTERNS instance has the following $2m+1$ strings (see Table 7 for details about the strings): (1) for each edge (v_i, v_j) there is a selection string S_{ij}, (2) there are m identical enforcer strings E, (3) there is one ground string. Moreover, the CONSENSUS PATTERNS instance consists of a substring length $k = 4n^2m + n$.

Table 7. Strings occuring in the reductions.

Name	Notation	Form	Length
Vertex string	V_i	$\mathtt{0}^{i-1}\mathtt{1}\mathtt{0}^{n-i}$	n
Delimiter string	D	$(\mathtt{0}^n\mathtt{1}^n)^{nm}$	$2n^2m$
Selection string	S_{ij}	DV_iDV_jD	$6n^2m + 2n$
Enforcer string	E	$D\mathtt{1}^nD$	$4n^2m + n$
Ground string	G	$D\mathtt{0}^nD$	$4n^2m + n$
Base string	DXD	$D(\mathtt{0}/\mathtt{1})^nD$	$4n^2m + n$

We now show that there is a one-one correspondance between the optimum patterns and the minimum vertex covers. Each optimum pattern is a base string DXD, where X is of length n and a `1` in position i corresponds to the i'th vertex being part in a minimum vertex cover. Note that if X represents a cover V' then the cost of the pattern DXD is

$$\underbrace{m(n-|V'|)}_{\text{enforcer strings}} + \underbrace{m(|V'|-1)}_{\text{selection strings}} + \underbrace{|V'|}_{\text{ground}} = m(n-1)+|V'|. \tag{6}$$

This is strictly decreasing with the size of the cover $|V'|$ and no more than $m(n-1)+n$.

(i) *The optimum pattern is a base string, DXD.* Let s be the substring of length k selected from S_{ij}, then due to the triangle inequality $\mathrm{d}_H(G,c) + \mathrm{d}_H(s,c) \geq \mathrm{d}_H(G,s)$. If s is not DV_iD or DV_jD (i.e. the prefix or the suffix), then the delimiter strings of G and s are unaligned and $\mathrm{d}_H(G,s) \geq 4nm-2$. This is more than $m(n-1)+n$ and therefore not optimum. Thus, all selected substrings are base strings and hence also the optimum consensus pattern.

(ii) *X corresponds a vertex cover.* There are m enforcer strings so whenever a substring DV_iD is selected from a selection component then, by majority vote, X also has a `1` in position i. Thus X must correspond to a cover. Moreover, since (6) is strictly decreasing in the size of $|V'|$, we have shown that the optimum pattern corresponds to a minimum vertex cover, and vice versa.

Clearly the reduction can be done in polynomial time, so CONSENSUS PATTERNS is NP-hard. □

Theorem 8. *The decision version of* SUBSTRING PARSIMONY *is NP-complete for binary trees.*

Proof. Clearly SUBSTRING PARSIMONY $\in$ NP. As Fig. 6 indicates, the binary tree in the SUBSTRING PARSIMONY instance consists of a "grounded backbone" with m branches; one for each edge in the VERTEX COVER instance. Each

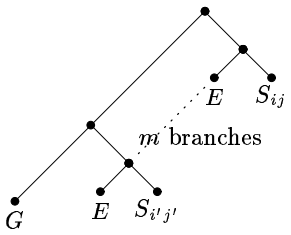

Fig. 6. The construction for SUBSTRING PARSIMONY.

branch has one enforcer string and one selection string, related to an edge in the VERTEX COVER instance. Consequently there are a total of $2m+1$ leafs in the tree. Moreover, the SUBSTRING PARSIMONY instance has a substring length $k = 4n^2m+n$ and a decision limit $K = m(n-1)+c$, where c is the limit of the VERTEX COVER instance.

If there is a vertex cover of size c, then there is a substring parsimony alignment of cost K. Let V' be a vertex cover of size c. Then the alignment in which all labels are identical to the base string DXD corresponding to V' has cost $m(n-1)+|V'| = K$ (same cost as before: (6)).

If there is a substring parsimony alignment of cost K, then there is a vertex cover of size c. We show that there exists a minimum substring parsimony alignment in which all labels are identical and that the labels correspond to a vertex cover. Therefore, since (6) is strictly decreasing in the size of $|V'|$, such a label corresponds to a minimum vertex cover.

(i) *In the minimum substring parsimony alignment the label at every vertex is a base string DXD.* As above, let s be the substring selected from S_{ij}. Then as before, due to the triangle inequality $\mathrm{d}_H(E,c)+\mathrm{d}_H(c,s) \geq \mathrm{d}_H(E,s)$, where c is the label at the vertex connecting E and S_{ij}. Hence s is either the prefix DV_iD or the suffix DV_jD, otherwise $\mathrm{d}_H(E,s) \geq 4nm-2$. In a minimum alignment, since all leaf labels are base strings, all internal labels are base strings.

(ii) *There is a minimum substring parsimony alignment in which all labels are identical and correspond to a vertex cover.* Whenever a substring DV_iD is selected from a selection string then the parent of the selection vertex is labeled with DXD, where X by majority vote has a `1` in the i'th vertex position. Without effecting the cost of the alignment, all other cover strings can be changed so that they have a `1` in position i. Moreover, since the alignment is optimum, all cover strings have `0` in all remaining positions (positions not selected by the selection strings). By this argument all internal vertices are labeled with an identical base string. In addition, since the base string has a `1` from each selection string the base string corresponds to a vertex cover.

The reduction is polynomial and thus SUBSTRING PARSIMONY is NP-hard. □

6 Acknowledgments

I am very grateful for the help and support that I have received from my supervisor Prof. Jens Lagergren. Without him this paper would not have been written. Moreover, I would like to thank an anonymous referee of an earlier version of this paper for many valuable comments.

References

[Aku98] T. Akutsu. Hardness results on gapless local multiple sequence alignment. Technical Report 98-MPS-24-2, 1998.

[AL89] S. F. Altschul and D. J. Lipman. Trees, stars and multiple biological sequence alignment. *SIAM Journal of Applied Mathematics*, 49:197–209, 1989.

[AMS+97] S. F. Altschul, T. L. Madden, A. A. Schaffer, J. Zhang, Z. Zhang, W. Miller, and D. J. Lipman. Gapped blast and psi-blast: a new generation of protein database search programs. *Nucleic Acids Res*, 25:3389–3402, 1997.

[BA86] D. Baconn and W. Anderson. Multiple sequence alignment. *Journal of Molecular Biology*, 191:153–161, 1986.

[BF95] P. Berman and T. Fujito. On approximation properties of the independent set problem in degree 3 graphs. *Lecture Notes in Computer Science*, 955:449–??, 1995.

[BST02] M. Blanchette, B. Schwikowski, and M. Tompa. Algorithms for phylogenetic footprinting. *J. Comput. Bio.*, 9(2):211–223, 2002.

[BT02] J. Buhler and M. Tompa. Finding motifs using random projections. *J. Comput. Bio.*, 9(2):225–242, 2002.

[BV01] P. Bonizzoni and G. D. Vedova. The complexity of multiple sequence alignment with SP-score that is a metric. *TCS*, 259(1–2):63–79, 2001.

[CWC92] S. C. Chan, A. K. C. Wong, and D. K. Y. Chiu. A survey of multiple sequence comparison methods. *Bulletin of Mathematical Biology*, 54(4):563–598, 1992.

[FGN02] M. Fellows, J. Gramm, and R. Niedermeier. On the parameterized intractability of CLOSEST SUBSTRING and related problems. In *STACS*, 2002.

[Jus01] W. Just. Computational complexity of multiple sequence alignment with sp-score. *Journal of Computational Biology*, 8(6):615–623, 2001.

[LMW99] M. Li, B. Ma, and L. Wang. Finding similar regions in many strings. In *Proceedings of the Thirty-First Annual ACM Symposium on Theory of Computing (STOC'99)*, pages 473–482, New York, May 1999. Association for Computing Machinery.

[LMW01] M. Li, B. Ma, and L. Wang. Finding similar regions in many sequences. *accepted by Journal of Computer and System Sciences*, July 2001.

[PS00] P. A. Pevzner and S. Sze. *Combinatorial approaches to finding subtle signals in DNA sequences.* Proc. 8th Int. Conf. on Intell. Sys. for Mol. Biol., August 2000.

[PY91] C. H. Papadimitriou and M. Yannakakis. Optimization, approximation, and complexity classes. *Journal of Computer and System Sciences*, 43(3):425–440, 1991.

[SC83] D. Sankoff and R. J. Cedergren. *Time Warps, String Edits, and Macromolecules: the Theory and Practice of Sequence Comparison*, chapter Simultaneous comparison of three or more sequences related by a tree, pages 253–264. Addison-Wesley, 1983.

[SP01] J. S. Sim and K. Park. The consensus string problem for a metric is np-complete. *Journal of Discrete Algorithms*, 2(1), 2001.

[WJ94] L. Wang and T. Jiang. On the complexity of multiple sequence alignment. *J. Comput. Bio.*, 1:337–348, 1994.

[WJG01] L. Wang, T. Jiang, and D. Gusfield. A more efficient approximation scheme for tree alignment. *SIAM Journal on Computing*, 30(1):283–299, February 2001.

[WJL96] L. Wang, T. Jiang, and E. L. Lawler. Approximation algorithms for tree alignment with a given phylogeny. *Algorithmica*, 16(3):302–315, September 1996.