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, the construction can be extended to prove NP-hardness results for Consensus Patterns and Substring Parsimony.

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.

In [1] 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

T. Ibaraki, N. Katoh, and H. Ono (Eds.): ISAAC 2003, LNCS 2906, pp. 352–363, 2003.

improved by Bonizzoni and Vedova [2], who showed that the problem is NP-hard for the binary alphabet and a specific metric. The result was extended further by Just [3] 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 [1] 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 [4] 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 [5] 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 [6,7] 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 and Substring Parsimony 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 [8,9,10] and W[1]-hard in [11]. The constructions in both proofs are similar to that of Star Alignment and can be found in [12].

In the following section Multiple Alignment with SP-score is proved to be NP-hard for binary or larger alphabets under the unit metric, the result for all metrics is achieved by a slight modification and is given in [12]. Then in Sect. 3 Star Alignment and Tree Alignment are proved to be NP-hard for binary or larger alphabets under any metric.

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 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

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

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 requires only a slight modification and is given in [12]. Some of the ideas in the construction[2] is by Bonizzoni and Vedova [2].

Theorem 1. *The decision version of* Multiple Alignment *with SP-score is NP-complete for the binary alphabet under each metric. (Proof in [12])*

Corollary 1. *The decision version of* Multiple Alignment *with SP-score is NP-complete for the binary alphabet under the unit metric. (Proof below)*

First the reduction is presented and on page 356 its correctness is proved. The reduction is from Independent Set in 3-regular graphs: Independent R3 Set[3]. 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.

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

[2] Bonizzoni and Vedova made a similar 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.

[3] In [13,14] Independent Set is shown to be NP-complete even for graphs with degree bounded by 3. It is a simple matter to extend the result to 3-regular graphs.

strings are identical to $T = (\mathtt{10}^b)^{n-1}\mathtt{1}$. 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

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

$$C_{ij} = \mathtt{0}^{4n}(\mathtt{00}^b)^{i-1}\mathtt{10}^{b-4n}(\mathtt{00}^b)^{j-i-1}\mathtt{10}^b(\mathtt{00}^b)^{n-j-1}\mathtt{00}^{4n}.$$

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. □

3 Star Alignment and Tree Alignment Are NP-Hard

In this section Star Alignment and Tree Alignment are proved NP-hard for all binary or larger alphabets under any metric. Both reductions are from Vertex Cover and have very similar constructions. The metric is depicted in Table 1. We assume w.l.o.g. that $\beta \leq \gamma$ and that $1 \leq \min(\alpha,\beta)$. Due to space limitations we have, in this version of the paper, omitted the proof of the special case $\alpha = \beta + \gamma$. From now on we assume that $\alpha < \beta + \gamma$. Moreover, we have omitted the proofs of several technical lemmas, these can be found in [12].

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.*

3.1 Star Alignment Is NP-Hard

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. 1, 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'.

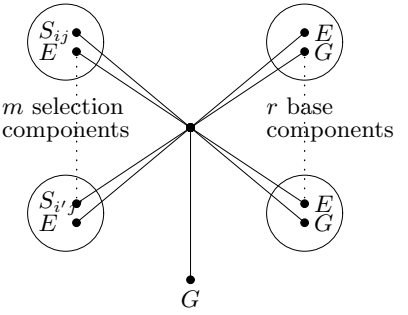

Fig. 1. Construction for Star Alignment. See 1–3 above for an outline of the components. The strings are from Table 3.

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 $B_0 = P\mathtt{0}P$ and $B_1 = P\mathtt{1}P$ 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 $P = \mathtt{0}^s\mathtt{1}^s\mathtt{0}^s$. The delimiter strings consists of $|C|$ `1`'s: $D = \mathtt{1}^{|C|}$.

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

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	$\mathtt{1}^{\|V_i\|}$	$\|B_0\|n$
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		$DDCDD$	$5\|D\|$

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 the base components are given in the lemma below.

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. 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.

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 2. *The decision version of* Star Alignment *is NP-complete for the binary alphabet under all metrics in which* $\alpha < \beta + \gamma$.

Proof. Clearly Star Alignment $\in$ NP. As Fig. 1 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

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

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

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

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 $\alpha < \beta + \gamma$. □

3.2 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 *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. 2) is very similar to that of Star Alignment. Only some additional arguments to handle the tree structure are required.

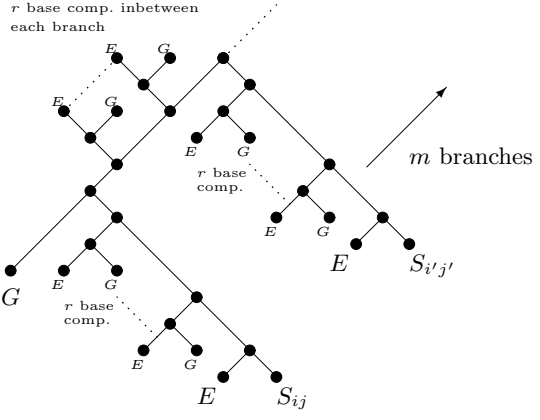

Fig. 2. Construction for Tree Alignment. There are m branches (one for each edge), each branch has r base components and one selection component. Moreover, in between each branch there are r base components.

Theorem 3. *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. 2 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

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

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).

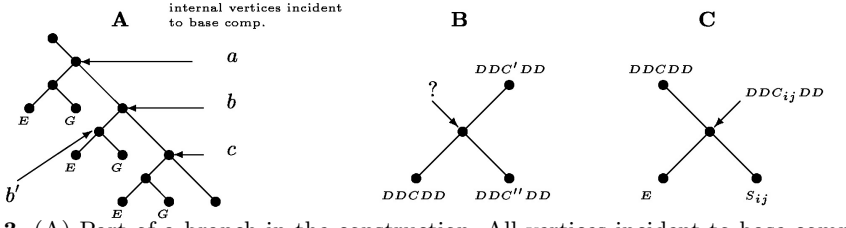

Fig. 3. (A) Part of a branch in the construction. All vertices incident to base components are labeled with the same base string. (B) The vertex connecting a branch is labeled with a base string. (C) The vertex in a selection component is labeled with a base string.

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 Lem. 2, if a vertex incident (e.g. vertex c in Fig. 3A) 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 than in (4).

Let the vertices a,b,c, and b', in Fig. 3A 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 (Lem. 1) and the properties of the base component (Lem. 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 3B and 3C. A detailed proof of this fact is given in [12].

(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, (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, 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 $\alpha < \beta + \gamma$. □

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

1. L. Wang and T. Jiang. On the complexity of multiple sequence alignment. *J. Comput. Bio.*, 1:337–348, 1994.
2. 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.
3. W. Just. Computational complexity of multiple sequence alignment with sp-score. *Journal of Computational Biology*, 8(6):615–623, 2001.
4. 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.
5. J. S. Sim and K. Park. The consensus string problem for a metric is np-complete. *Journal of Discrete Algorithms*, 2(1), 2001.
6. L. Wang, T. Jiang, and E. L. Lawler. Approximation algorithms for tree alignment with a given phylogeny. *Algorithmica*, 16(3):302–315, September 1996.
7. 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.
8. M. Li, B. Ma, and L. Wang. Finding similar regions in many sequences. *accepted by Journal of Computer and System Sciences*, July 2001.
9. M. Blanchette, B. Schwikowski, and M. Tompa. Algorithms for phylogenetic footprinting. *J. Comput. Bio.*, 9(2):211–223, 2002.
10. T. Akutsu. Hardness results on gapless local multiple sequence alignment. Technical Report 98-MPS-24-2, 1998.
11. M. Fellows, J. Gramm, and R. Niedermeier. On the parameterized intractability of CLOSEST SUBSTRING and related problems. In *STACS*, 2002.
12. I. Elias. Settling the intractability of multiple alignment. Technical Report TRITA-NA-0316, Nada, KTH, 2003.
13. C. H. Papadimitriou and M. Yannakakis. Optimization, approximation, and complexity classes. *Journal of Computer and System Sciences*, 43(3):425–440, 1991.
14. P. Berman and T. Fujito. On approximation properties of the independent set problem in degree 3 graphs. *Lecture Notes in Computer Science*, 955:449ff, 1995.