Hardness of MSA with Selective Column Scoring*

Andrea Caucchiolo and Ferdinando Cicalese[1][0000−0003−1652−0599]

Department of Computer Science, University of Verona, Italy
{andrea.caucchiolo, ferdinando.cicalese}@univr.it

Abstract. Multiple Sequence Alignment (MSA for short) is a well known problem in the field of computational biology. In order to evaluate the quality of a solution, many different scoring functions have been introduced, the most widely used being the Sum-of-pairs score (SP-score). It is known that computing the best MSA under the SP-score measure is NP-hard.
In this paper, we introduce a variant of the Column score (defined in Thompson et al. 1999), which we refer to as Selective Column score: Given a symbol $a \in \Sigma$, the score of the i-th column is one if and only if all symbols of the same column are a, and otherwise zero. The a-column score of an alignment is then the number of columns made of only character a.
We show that finding the optimal MSA under the Selective Column Score is NP-hard for all alphabets of size $|\Sigma| \geq 2$ by reducing from MIN-2-SAT.

Keywords: Multiple Sequence Alignment · Column score · NP-completeness

1 Introduction

Alignments are fundamental methods for the comparison of biological sequences. They are extensively employed for assessing similarity among sequences as well as in subprocedures of more complex operations, like the computation of approximate overlaps in the context of sequencing.

Definition 1 (Multiple alignment). *Let $s_1, \ldots, s_m$, be m strings over some alphabet Σ. Let - $\notin \Sigma$ be a gap symbol and $\Sigma' = \Sigma \cup \{\text{-}\}$. A multiple alignment $\tilde{S}$ of $s_1, \ldots, s_m$ of length ℓ is a tuple of strings $\tilde{s}_1, \ldots, \tilde{s}_m \in \Sigma'$ such that*

- $|\tilde{s}_1| = |\tilde{s}_2| = \cdots |\tilde{s}_m| = \ell$,
- *for each $i = 1, \ldots, m$, if we remove from $\tilde{s}_i$ all and only the gap characters we obtain the string s_i,*
- *there does not exist a position j where a gap occurs in all $\tilde{s}_i$, i.e., for all $j = 1, \ldots, \ell$ there exists an $i \in \{1, \ldots, m\}$, such that*[1] *$\tilde{s}_i[j] \neq$ -.*

[1] For a string s and integer i, we denote by $s[i]$ the ith character of s.

It is useful to visualize the alignment as a matrix, where rows are the strings $\tilde{s}_i$. Therefore, for each $j = 1, \ldots, \ell$ we refer to the sequence $\tilde{s}_1[j], \ldots, \tilde{s}_m[j]$ as the jth column of the alingment.

Many proposals exist regarding the choice of the most appropriate objective function that should be optimized when computing an alignment of a set of strings [1, 11, 12].

In this paper, we focus on the so called *column score* analyzed in [12], where the alignment is assessed by counting the number of columns consisting of a single character.

Definition 2 (Column Score). *Let $s_1, \ldots, s_m$ be a sequence of strings from some alphabet Σ. Given an alignment $\tilde{S} = \tilde{s}_1, \ldots, \tilde{s}_m$ of length ℓ of $s_1, \ldots, s_m$, and a character $a \in \Sigma$, the* a-column score *of $\tilde{S}$, denoted $cs_a(\tilde{S})$ is the number of columns made of only character a, i.e.,*

$$cs_a(\tilde{S}) = |\{j \in \{1, \ldots, \ell\} \mid \forall i = 1, \ldots, m,\ \tilde{s}_i[j] = a\}|.$$

The column score *of $\tilde{S}$, denoted $cs(\tilde{S})$ is the number of columns made of a single character a, i.e.,*

$$cs(\tilde{S}) = |\cup_{a \in \Sigma} \{j \in \{1, \ldots, \ell\} \mid \forall i = 1, \ldots, m,\ \tilde{s}_i[j] = a\}| = \sum_{a \in \Sigma} cs_a(\tilde{S}).$$

In particular we prove the NP-completeness of the following problem.

Selective Column Score Alignment (SCS-Align)

Input: Strings $s_1, \ldots, s_m$, over some alphabet Σ, an integer $M \geq \max_i |s_i|$, a non-negative integer κ, and a *selected* character $a \in \Sigma$.

Question: Is there an aligment $\tilde{S} = \tilde{s}_1, \ldots, \tilde{s}_m$ of $\tilde{s}_1, \ldots, \tilde{s}_m$ such that the length of $\tilde{S}$ is at most M and the a-column score of $\tilde{S}$ is at least κ?

In contrast with other objective functions for multiple sequence alignment, to the best of our knowledge, before our result, nothing was known about the tractability of computing the best possible alignment under the (selective) a-column score.

Main Result. In this paper, we show that the Selective Column Score Alignment problem (and several of its variants) is NP-complete.

Related work. It is not hard to see that without the upper bound M on the alignment length, computing an alignment of maximum possible column score is equivalent to finding the longest common subsequence of the input strings, which is known to be an NP-hard problem [9]. However, in contrast to our SCS-Align, the Longest Common Subsequence problem is easily solvable in polynomial time when the output is restricted to subsequences made of a single given character—note that a a-column score alignment asks for a subsequence of

this type. Another important issue/difference with respect to modelling column score alignment as an instance of LCS is that an alignment obtained starting from a longest common subsequence might have an uncontrollable number of gaps per row. We show that the hardness result on SCS holds for an equivalent variant of the problem where we impose a bound of 1 on the number of gaps per row. In this respect, the length constraint on the solutions of SCS-Align have some similarity with the gapped common subsequences considered in [4, 5]. In these papers, however, the authors consider only the special case of two input strings and do not discuss complexity issues of the general instances.

The most commonly considered objective function for optimizing multiple sequence alignments is the so called Sum of Pairs score (SP-score) introduced by Carrillo and Lipman [3]. The SP-score generalizes the edit distance based score for pairwise alignments (equivalently multiple alignments of only two strings). An optimal pairwise alignment can be computed in quadratic time by the Needleman and Wunsh dynamic programming approach [10]. In contrast, computing an optimal SP-score based multiple sequence alignment is known to be NP-hard [13].

Alternative scoring measures, based on the distance to a consensus string, are investigated in Li et al. [8]. In its simplest form, given an alignment a consensus string can be obtained by selecting for each column a majority symbol. The score of the alignment is the sum of the Hamming distance between each input string and the consensus sequence. The authors of [8] investigate several variants of this approach showing that they are in general NP-hard. They also consider a sort of inverse procedure, where a median string of the input string is computed that minimizes the sum of gap-bounded edit distance of each string from the median. Once this string has been found the alignment can be constructed by optimally aligning each input string to the median.

Both the above approaches (SP-score and distance from consensus) are based on pairwise comparisons of strings. More precisely, in the SP-score case each possible pairwise comparison of the input strings is computed and the corresponding scores are added up to obtain the final score. In the case of distance to consensus, a median string is computed and the alignment score is defined as the sum of pairwise alignment scores of the input strings to the median. The column score is a basic model of scoring functions that are obtained by evaluating the set of elements in a column, rather than composing evaluations of single pairs of elements in a column.

Organization of the paper. In section 2, we prove the NP-completeness of our basic version of the column score, Selective Column Score Alignment, by reducing from MIN-2-SAT. In section 3, we show how the reduction can be extended to variants where there is no selected character, and the bound on the length of the alignment is substituted with a very sharp bound on the number of gaps per row of the alignment. Section 4 concludes the paper with some open questions.

2 The NP-completeness of Selective Column Score Alignment

In this section, we establish the complexity of the SCS-Align problem defined in the introduction. We provide a reduction from the NP-complete problem MIN-2-SAT [7].

Min-2-Sat
Input: A 2-CNF formula ϕ and an integer k
Question: Is there an assignment to the variables of ϕ that satisfies at most k clauses?

The Reduction. Let $I = (\phi, k)$ be an instance of MIN-2-Sat. Let $x_1, \ldots, x_n$ be the variables of ϕ and $C_1, \ldots, C_m$ be the clauses of ϕ. Then, $\phi = C_1 \wedge C_2 \wedge \cdots \wedge C_m$, where for each $i = 1, \ldots, m$, $C_i = (l_1^i \vee l_2^i)$, with $l_j^i \in \{x_1, \ldots x_n, \overline{x_1}, \ldots, \overline{x_n}\}$.

From I, we now build an instance $J = (\{s_0, \ldots, s_{n+1}\}, M, \kappa, a)$ for SCS-Align, where each s_i is a binary string, M is a bound on the alignment length, a is a character in the alphabet $\Sigma = \{0, 1\}$, and κ is the bound on the number of columns of the alignments that are expected to contain only the selected character a.

We set $M = 8mn+3m+2$, $a = 0$ and build $n+2$ binary strings: $s_0, s_1, \ldots, s_n, s_{n+1}$ where $|s_0| = |s_{n+1}| = M$ and $|s_i| = M - 1$ for each $i = 1, \ldots, n$.

More precisely we define:

$$s_0 = 1(01)^{2mn}(101)^m(10)^{2mn}1, \tag{1}$$

and for each $i = 1, \ldots, n$,

$$s_i = 0(00)^{(i-1)2m}(10)^{2m}(00)^{(n-i)2m}c_i^{(1)} \ldots c_i^{(m)}(00)^{(i-1)2m}(10)^{2m}(00)^{(n-i)2m}, \tag{2}$$

where, for each $j = 1, \ldots, m$,

$$c_i^{(j)} = \begin{cases} 100 \text{ if } \overline{x_i} \in C_j \\ 010 \text{ if } x_i \in C_j \\ 000 \text{ else} \end{cases} \tag{3}$$

Finally, we define

$$s_{n+1} = 0^M, \tag{4}$$

and set

$$\kappa = 2mn + m - k.$$

Example 1: Let $\phi = (\overline{x_1} \vee \overline{x_2}) \wedge (x_2 \vee x_3)$ be a 2-CNF formula with $m = 2$ and $n = 3$. The strings $s_0, \ldots, s_4$ obtained by the above process are then:

$$
\begin{aligned}
s_0 &: 1(01)^4(01)^4(01)^4101101(10)^4(10)^4(10)^41\\
s_1 &: 0(10)^4(00)^4(00)^4100000(10)^4(00)^4(00)^4\\
s_2 &: 0(00)^4(10)^4(00)^4100010(00)^4(10)^4(00)^4\\
s_3 &: 0(00)^4(00)^4(10)^4000010(00)^4(00)^4(10)^4\\
s_4 &: 0(00)^4(00)^4(00)^4000000(00)^4(00)^4(00)^40
\end{aligned}
$$

with s_1, s_2, s_3 respectively representing the variables x_1, x_2, x_3.

Remark 1. Let $\tilde{S} = \tilde{s}_0, \tilde{s}_1, \ldots, \tilde{s}_{n+1}$ be an alignment of length $\ell \leq M$ of the strings $s_0, \ldots, s_{n+1}$. The following three facts are easy implied by the structure of the instance produced by our reduction.

1. Since $|s_0| = |s_{n+1}| = M$ we have that it must be $\ell = M$, hence $\tilde{s}_0 = s_0$ and $\tilde{s}_{n+1} = s_{n+1}$. Moreover, for each $i = 1, \ldots, n$, because of $|s_i| = M - 1$ it follows that $\tilde{s}_i$ contains exactly one gap. We will use $gap(i)$ to denote the position of the single gap in $\tilde{s}_i$, i.e., the index $j = 1, \ldots, M$, such that $\tilde{s}_i[j] =$ -.
2. Following from Fact 1. and since $\tilde{s}_{n+1}$ is made of only zeroes there exist no columns with only 1, and the only columns that can contain only 0 are those corresponding to the positions of the zeroes in s_0. In other words, for the instances produced by our reduction, we have $cs_0(\tilde{S}) = cs(\tilde{S})$.
3. For each $i = 1, \ldots, n$ and $j = 1, \ldots, M$, we say that $\tilde{s}_i$ *blocks* position j if $\tilde{s}_0[j] = s_0[j] = 0$ and $\tilde{s}_i[j] = 1$. A column j is made of only zeroes if and only if there is no $i = 1, \ldots, n$ such that $\tilde{s}_i$ blocks position j.

For each $i = 1, \ldots, n$, let $s_i^T = s_i$- be the string obtained by adding a gap at the end of s_i. Let $s_i^F =$ -s_i be the string obtained by adding a gap at the beginning of s_i. Then, we have

$$
\begin{aligned}
s_i^T &= 0(00)^{(i-1)2m}(10)^{2m}(00)^{(n-i)2m}c_{i,T}^{(1)}\ldots c_{i,T}^{(m)}(00)^{(i-1)2m}(10)^{2m}(00)^{(n-i)2m}\text{-},\\
s_i^F &= \text{-}(00)^{(i-1)2m}(01)^{2m}(00)^{(n-i)2m}c_{i,F}^{(1)}\ldots c_{i,F}^{(m)}(00)^{(i-1)2m}(01)^{2m}(00)^{(n-i)2m}0,
\end{aligned}
$$

where, for each $j = 1, \ldots, m$, $c_{i,T}^{(j)}$ is equal to $c_i^{(j)}$ (defined in (3)) and

$$
c_{i,F}^{(j)} = \begin{cases} 010 \text{ if } \overline{x_i} \in C_j \\ 001 \text{ if } x_i \in C_j \\ 000 \text{ else} \end{cases}
$$

Let us define *the head of* s_0 to be the prefix $s_0^h = s_0[1]\ldots s_0[4mn+1] = 1(01)^{2mn}$.

Let us define *the core of* s_0 to be the substring $s_0^p = s_0[4mn+2]\dots s_0[4mn+3m+1] = (101)^m$.

Finally, let us define *the tail of* s_0 to be the suffix $s_0^t = s_0[4mn+3m+2]\dots s_0[8mn+3m+2] = (10)^{2mn}1$.

By a direct comparison with the definition of s_0 in (1), we have the following proposition, stating useful properties of strings s_i^F, s_i^T.

Proposition 1. *For each $i = 1, \dots, n$, the string s_i^T blocks*

- *exactly $2m$ positions of the head of s_0;*
- *no position of the tail of s_0;*
- *the position of the jth zero in the core of s_0 if and only if $x_i \in C_j$, i.e., if the assignment $x_i = True$ satisfies clause C_j.*

Analogously, we have that the string s_i^F blocks

- *exactly $2m$ positions of the tail of s_0;*
- *no position of the head of s_0;*
- *the position of the jth zero in the core of s_0 if and only if $\overline{x_i} \in C_j$, i.e., if the assignment $x_i = False$ satisfies clause C_j.*

Moreover, for each $i \neq i'$ the strings $s_i^T, s_i^F, s_{i'}^T, s_{i'}^F$ block distinct positions in the head and tail of s_0.

The following lemma proves that our reductions maps *yes* instances of MIN-2-Sat to *yes*-instances of SCS-Align.

Lemma 1. *If there exists an assignment A for ϕ that satisfies at most k clauses then there exists an alignment $\tilde{S} = \tilde{s}_0, \dots, \tilde{s}_{n+1}$ of $s_0, \dots, s_{n+1}$ such that the length of $\tilde{S}$ is M and there are at least $\kappa = 2mn + m - k$ columns made only of zeroes.*

Proof. Given an assignment A that satisfies at most k clauses of ϕ, we define the alignment $\tilde{S}$ as follows: for each $i = 0, \dots, n+1$ we let

$$\tilde{s}_i = \begin{cases} s_i & \text{if } i = 0 \text{ or } i = n+1 \\ s_i^T & \text{if } A \text{ sets } x_i = True \\ s_i^F & \text{if } A \text{ sets } x_i = False \end{cases} \tag{5}$$

By the definition of s_i^F and s_i^T and Proposition 1, we have that strings $\tilde{s}_1, \dots, \tilde{s}_n$ block exactly $2mn$ of the positions in the head and the tail of s_0. Moreover, the position of the jth zero in the core of s_0 is blocked by some string $\tilde{s}_i$ if and only if the truth value of x_i in A satisfies C_j. Since A satisfies at most k clauses, we have that at most k positions of the core of s_0 are blocked by some $\tilde{s}_i$. It total there are $4mn+m$ positions in s_0 corresponding to a zero character and at most $2mn+k$ are blocked by the strings of the alignment. Hence, there are at least $2mn+m-k$ positions which are not blocked, equivalently $\geq 2mn+m-k$ columns made of only zeros.

Example 2: Using the same CNF formula as Example 1 with the assignment $A = \{x_1 = True, x_2 = False, x_3 = False\}$, by Lemma 1 we get this alignment:

$$\begin{aligned}
\tilde{s_0} = s_0 &: 1(01)^4(01)^4(01)^4101101(10)^4(10)^4(10)^41\\
\tilde{s_1} = s_1^T &: 0(10)^4(00)^4(00)^4100000(10)^4(00)^4(00)^4\text{-}\\
\tilde{s_2} = s_2^F &: \text{-}(00)^4(01)^4(00)^4010001(00)^4(01)^4(00)^40\\
\tilde{s_3} = s_3^F &: \text{-}(00)^4(00)^4(01)^4000001(00)^4(00)^4(01)^40\\
\tilde{s_4} = s_4 &: 0(00)^4(00)^4(00)^4000000(00)^4(00)^4(00)^40
\end{aligned}$$

We can see that the strings s_1^T, s_2^F, s_3^F block a total of 13 columns, and knowing that the assignment A satisfies only one clause we have $k = 1$, therefore it holds $13 \geq 2mn + k = 13$.

Lemma 2. *Let κ^* be the maximum column score achievable by an alignment of length M of the strings $s_0, \ldots, s_{n+1}$. Then, there exists an alignment $\tilde{S}$ of $s_0, \ldots, s_{n+1}$, such that*

1. *the length of $\tilde{S}$ is M,*
2. *$cs(\tilde{S}) = \kappa^*$,*
3. *for each $i = 1, \ldots, n$, we have $gap(i) \in \{1, M\}$, where $gap(i)$ denotes the position of the gap in $\tilde{s}_i$, i.e., the index $j \in \{1, \ldots, M\}$ such that $\tilde{s}_i[j] =$ -.*

Proof. Let $\tilde{S}$ be an alignment that, among those satisfying conditions 1. and 2., has the minimum number of strings $\tilde{s}_i$ violating condition 3. And assume, by contradiction, that such number is greater than 0.

Then there exists $i \in \{1, 2, \ldots, n\}$ such that the position of the gap in $\tilde{s}_i$ is not in $\{1, M\}$. We are going to show that we can modify $\tilde{s}_i$ moving the gap in position 1 or M and obtain another alignment satisfying condition 1. and 2. The existence of such an alignment contradicts the minimality of $\tilde{S}$ with respect to condition 3, implying that in fact in $\tilde{S}$ for all $i = 1, \ldots, n$ we must have $gap(i) \in \{1, M\}$ as desired.

Let n_i be the number of positions that $\tilde{s}_i$ blocks altogether in the tail and the head of s_0. We have

$$n_i = \begin{cases}
2m & \text{if } gap(i) < 4m(i-1) + 2,\\
2m & \text{if } gap(i) \geq 4mi + 2 + 4mn + 3m,\\
4m & \text{if } 4mi + 2 \leq gap(i) \leq (i-1)4m + 1 + 4mn + 3m,\\
2m + \lceil \frac{x}{2} \rceil & \text{if } gap(i) = (i-1)4m + 1 + x,\ 1 \leq x \leq 4m,\\
2m + \lceil \frac{4m-x+1}{2} \rceil & \text{if } gap(i) = 4m(i-1) + 1 + 4mn + 3m + x,\ 1 \leq x \leq 4m.
\end{cases}$$

Note that these numbers refer to positions that can be blocked only by $\tilde{s}_i$. Hence, unblocking any of such positions implies a net increase in the column score of the resulting alignment.

We apply the following three cases:
Case 1. $gap(i) \leq 4mi + 1$. In this case $\tilde{s}_i$ blocks $2m$ positions in the tail of s_0 and possibly some positions in the head of $\tilde{s}_0$. If we modify $\tilde{s}_i$ by moving the

gap to the first position (equivalently, we set $\tilde{s}_i = s_i^F$) we reduce the number of positions blocked in the tail and head of $\tilde{s}_0$ to $2m$. Moreover, the characters of the new $\tilde{s}_i$ which are aligned to the core of $\tilde{s}_0$ do not change. Then, the new alignment has not decreased the number of columns made of only zeros, i.e., it satisfies conditions 1 and 2 and has one more string satisfying condition 3.

Case 2. $gap(i) \geq 4mn + 3m + 4m(i-1) + 2$. In this case $\tilde{s}_i$ blocks $2m$ positions in the head of s_0 and possibly some positions in the tail of $\tilde{s}_0$. If we modify $\tilde{s}_i$ by moving the gap to the last position (equivalently, we set $\tilde{s}_i = s_i^T$) we reduce the number of positions blocked altogether in the tail and head of $\tilde{s}_0$ to only $2m$. Moreover, the characters of the new $\tilde{s}_i$ which are aligned to the core of $\tilde{s}_0$ do not change. Then, the new alignment has not decreased the number columns made of only zeros, i.e., it satisfies conditions 1 and 2 and has one more string satisfying condition 3.

Case 3. $4mi + 2 \leq gap(i) \leq 4m(i-1) + 1 + 4mn + 3m$. In this case, $\tilde{s}_i$ blocks $2m$ positions in the head of s_0 and $2m$ positions in the tail of s_0. It is possible that the position of the gaps allows to align the central zeros of $\tilde{s}_i$ (those belonging to the substrings $c_i^{(j)}$) to the zeroes of the core of s_0. If we modify $\tilde{s}_i$ by moving the gap to the first position (equivalently, we set $\tilde{s}_i = s_i^F$) we reduce the number of positions blocked altogether in the tail and head of $\tilde{s}_0$ to $2m$. Even if in the resulting new alignment the new $\tilde{s}_i$ blocked all the positions in the core of s_0, which are m, the net gain in the number of columns made of only zeroes would be m. Then, the new alignment has strictly increased the number of columns made of only zeros, i.e., it satisfies conditions 1 and 2 and has one more string satisfying condition 3.

We can see that the strings s_1^T, s_2^F, s_3^F block a total of 13 columns, and knowing that the assignment A satisfies only one clause we have $k = 1$, therefore it holds $13 \geq 2mn + k = 13$.

Lemma 3. *If there exists an alignment $\tilde{S}$ of $s_0, \ldots, s_{n+1}$ that has length M and such that $cs(\tilde{S}) \geq \kappa$, then there exists an assignment A for ϕ that satisfies at most k clauses.*

Proof. By Lemma 2, we can assume w.l.o.g., that in $\tilde{S}$ for each $i = 1, \ldots, n$, we have $gap(i) \in \{1, M\}$. Equivalently, for each $i = 1, \ldots, n$, we have that $\tilde{s}_i \in \{s_i^T, s_i^F\}$.

We now create the assignment A by setting for each $i = 1, \ldots, n$,

$$x_i = \begin{cases} True & \text{if } \tilde{s}_i = s_i^T \\ False & \text{if } \tilde{s}_i = s_i^F \end{cases} \tag{6}$$

By the definition of s_i^F and s_i^T and Proposition 1, we have that strings $\tilde{s}_1, \ldots, \tilde{s}_n$ block exactly $2mn$ of the positions in the head and the tail of s_0. Hence, exactly $2mn$ columns corresponding to the non-blocked positions in the head and tail of s_0 are made of only zeroes. Since $cs(\tilde{S}) \geq \kappa = 2mn + m - k$,

we have that at least $m - k$ columns corresponding to positions in the core of s_0 must be made of only zeroes. Equivalently, at most k positions in the core of s_0 are blocked by the strings $\tilde{s}_1, \ldots, \tilde{s}_n$.

By Proposition 1, we have that the position of the jth zero in the core of s_0 is blocked by some string $\tilde{s}_i$ if and only if the truth value of x_i in A satisfies C_j. Since at most k positions of the core of s_0 are blocked by some $\tilde{s}_i$, it follows that A satisfies at most k clauses.

As a result of Lemmas 1 and 3, we have the following.

Theorem 1. *The problem* Selective Column Alignment *is NP-complete already over an alphabet of size* ≥ 2.

3 Variants and Extensions

In this section we extend the NP-completeness of the SCS-Align to show the NP-completeness of other variants or reformulations of the problem that are also of interest. We start by the simple observation that the problem remains NP-complete even if we consider as objective function the column score, rather than the a-column score with respect to a specific character a.

Bounded Column Score Alignment (BCS-Align)

Input: Strings $s_1, \ldots, s_m$ over some alphabet Σ, an integer $M \geq \max_i |s_i|$, and a non-negative integer κ.

Question: Is there an aligment $\tilde{S} = \tilde{s}_1, \ldots, \tilde{s}_m$ of $s_1, \ldots, s_m$ such that the length of $\tilde{S}$ is at most M and the column score of $\tilde{S}$ is at least κ?

In fact, due to the presence of the string s_{n+1} being made of only zeroes as observed in point 2 of Remark 1, the proof in the previous section also shows the result in the following corollary. It is enough to observe that s_{n+1} forces only columns entirely made of 0 to be accepted.

Corollary 1. *The* Bounded Column Score Alignment *problem is NP-complete*

We can also reformulate the problem by interpreting the bound on the length of the alignment as a bound on the number of gaps. In fact, the following formulation is also easily shown to be NP-complete

gap-Bounded Column Score Alignment (gap-CS-Align)

Input: Strings $s_1, \ldots, s_m$ over some alphabet Σ, an integer bound g on the number of allowed gaps per string, and a non-negative integer bound κ on the number of uniform columns to be found.

Question: Is there an aligment $\tilde{S} = \tilde{s}_1, \ldots, \tilde{s}_m$ of $\tilde{s}_1, \ldots, \tilde{s}_m$ such that for each $i \in [m]$, $\tilde{s}_i$ contains at most g gaps, and the column score of $\tilde{S}$ is at least κ?

Corollary 2. *The* GAP-BOUNDED COLUMN SCORE ALIGNMENT *problem is NP-complete*

Proof. We use exactly the same reduction employed for the NP-completeness of SCS-ALIGN. We show that the particular case of GAP-CS-ALIGN with $g = 1$ is NP-complete. The simple observation needed is that in any alignment $\tilde{S} = \tilde{s}_0, \ldots, \tilde{s}_{n+1}$ of the strings $s_0, \ldots, s_{n+1}$ no gap can be present in $\tilde{s}_0$, (nor in $\tilde{s}_{n+1}$); otherwise, at least 2 gaps should be present in each $\tilde{s}_i$ $(i = 1, \ldots, n)$. Hence, we have that Remark 1 (and in particular point 1.) holds also on the basis of the bound on the number of gaps.

4 Final Remarks

We proved that computing the best multiple sequence alignment with respect to the column score is NP-hard even in a very restricted model where: strings are binary, we only try to optimize the number of columns of the alignment made of only zeroes, and we bound the number of gaps per row to 1.

It would be interesting to better understand the relationships of this problem to parameterized variants of the LONGEST COMMON SUPERSEQUENCE (LCS) problem [4, 5, 2, 6], since LCS is the natural model of column score based multiple sequence alignment in the absence of any bounds on the number of gaps or restrictions on the uniform columns. Another natural open problem regards the limits of approximability of our problems.

References

1. Berkemer, S.J., Höner zu Siederdissen, C., Stadler, P.F.: Compositional properties of alignments. Mathematics in Computer Science (2020)
2. Blin, G., Bulteau, L., Jiang, M., Tejada, P.J., Vialette, S.: Hardness of longest common subsequence for sequences with bounded run-lengths. In: Kärkkäinen, J., Stoye, J. (eds.) Combinatorial Pattern Matching. pp. 138–148. Springer Berlin Heidelberg, Berlin, Heidelberg (2012)
3. Carrillo, H., Lipman, D.: The multiple sequence alignment problem in biology. SIAM Journal on Applied Mathematics **48**(5), 1073–1082 (1988)
4. Iliopoulos, C.S., Kubica, M., Rahman, M.S., Waleń, T.: Algorithms for computing the longest parameterized common subsequence. In: Ma, B., Zhang, K. (eds.) Combinatorial Pattern Matching. pp. 265–273. Springer Berlin Heidelberg, Berlin, Heidelberg (2007)
5. Iliopoulos, C.S., Sohel Rahman, M.: Algorithms for computing variants of the longest common subsequence problem. Theoretical Computer Science **395**(2), 255–267 (2008)
6. Keller, O., Kopelowitz, T., Lewenstein, M.: On the longest common parameterized subsequence. Theoretical Computer Science **410**(51), 5347–5353 (2009), combinatorial Pattern Matching
7. Kohli, R., Krishnamurti, R., Mirchandani, P.: The minimum satisfiability problem. SIAM Journal on Discrete Mathematics **7**(2), 275–283 (1994)

8. Li, M., Ma, B., Wang, L.: Finding similar regions in many sequences. Journal of Computer and System Sciences **65**(1), 73–96 (2002)
9. Maier, D.: The complexity of some problems on subsequences and supersequences. J. ACM **25**(2), 322?336 (1978)
10. Needleman, S.B., Wunsch, C.D.: A general method applicable to the search for similarities in the amino acid sequence of two proteins. Journal of Molecular Biology **48**(3), 443–453 (1970)
11. Notredame, C.: Recent progress in multiple sequence alignment: a survey. Pharmacogenomics **3**(1), 131–144 (January 2002)
12. Thompson, J.D., Plewniak, F., Poch, O.: A comprehensive comparison of multiple sequence alignment programs. Nucleic Acids Research **27**(13), 2682–2690 (07 1999)
13. Wang, L., Jiang, T.: On the complexity of multiple sequence alignment. Journal of Computational Biology **1**(4), 337–348 (1994)