Complexities of the Centre and Median String Problems[⋆]

François Nicolas and Eric Rivals

L.I.R.M.M., CNRS U.M.R. 5506
161 rue Ada, F-34392 Montpellier Cedex 5, France
{nicolas, rivals}@lirmm.fr

Abstract. Given a finite set of strings, the MEDIAN STRING problem consists in finding a string that minimizes the sum of the distances to the strings in the set. Approximations of the median string are used in a very broad range of applications where one needs a representative string that summarizes common information to the strings of the set. It is the case in Classification, in Speech and Pattern Recognition, and in Computational Biology. In the latter, MEDIAN STRING is related to the key problem of Multiple Alignment. In the recent literature, one finds a theorem stating the NP-completeness of the MEDIAN STRING for unbounded alphabets. However, in the above mentioned areas, the alphabet is often finite. Thus, it remains a crucial question whether the MEDIAN STRING problem is NP-complete for finite and even binary alphabets. In this work, we provide an answer to this question and also give the complexity of the related CENTRE STRING problem. Moreover, we study the parametrized complexity of both problems with respect to the number of input strings.

1 Introduction

Given an alphabet Σ, a set W of strings over Σ, and the Levenshtein distance between strings, the problem of finding a string over Σ that minimizes the sum of distances to the strings of W is called the MEDIAN STRING problem. Alternative terminologies include the GENERALIZED MEDIAN STRING problem [1], the STAR ALIGNMENT problem [13] and also the STEINER STRING problem [7].

The MEDIAN STRING problem is of major significance in several areas of research: Pattern Recognition, Speech Recognition and Computational Biology. Its importance is reflected by the wide use of a polynomial time approximation, the *set median* string ([9, 18, 17, 7, 6, 8]). In this restricted version of the problem, the solution string must be taken in the input set W (it is also termed the "center string" in [7, p. 349]). One class of applications, encountered in all three areas, looks for a string (or a language) that models the input set of strings.

⋆ Published in LNCS Vol. 2676, p. 315 - 327, Springer Verlag, ISSN: 0302-9743, Combinatorial Pattern Matching: 14th Annual Symposium, CPM 2003, Morelia, Michoacan, Mexico, June 25-27, 2003.

In other words, this string summarizes the information shared by strings of W. Depending on the application, it then serves as an index for W (in databases and data mining), as a pattern that is searched for in longer texts (in computational biology [7, 19]) or used for classification purposes (in speech recognition [9], classification [7] and computational biology [7, 19]).

In [1], it is shown that CENTRE STRING and MEDIAN STRING are NP-hard for alphabets of size at least 4 and for unbounded alphabets, respectively. In [21], it is shown that MEDIAN STRING is NP-hard for alphabet of size 7 and a particular weighted edit distance. In many practical situations, the alphabet is of fixed constant size. In computational biology, the DNA and protein alphabets are respectively of size 4 and 20. However, other alphabet sizes are also used. Indeed, for some applications, one needs to encode the DNA or protein sequences on a binary alphabet that expresses only a binary property of the molecule, e.g., hydrophoby. For instance, it is the case in some protocols to identify similar DNA sequences [22]. The important practical question is whether CENTRE STRING and MEDIAN STRING are NP-hard for finite and even binary alphabets. In the abovementioned article, these questions remain open [1, p. 48]. These conjectures are solved in this paper. Additionnally, an interesting issue concerns the existence of fast exact algorithms when the number of input strings is fixed. We provide an answer to this issue for both CENTRE STRING and MEDIAN STRING.

1.1 Definitions

We denote by $\mathbb{N}$ the set of non negative integers and by $\mathbb{N}^*$ the set of positive integers. For all $n \in \mathbb{N}^*$, we denote by $\mathbb{N}_n^*$ the set $\{1, 2, \ldots, n\}$ and for every finite set X we denote by $\#X$ the cardinality of X.

Words. An *alphabet* is a non empty set of *letters*. In the sequel, Σ always denotes an alphabet. A *word* over Σ is a finite sequence of elements of Σ. The set of all words over Σ is denoted by $\Sigma^\star$. A *language* over Σ is any subset of $\Sigma^\star$. The empty sequence, denoted by ε, is called the *empty word*. For a word w, $|w|$ denotes the length of w. For all $a \in \Sigma$, $|w|_a$ denotes the number of occurences of the letter a in w. For all $i \in \mathbb{N}_{|w|}^*$, $w[i]$ denotes the i-th letter of w. Given two words x and y, we denote by xy the *concatenation* of x and y. For all $n \in \mathbb{N}$, we denote by x^n the *n-th power* of x that is, the concatenation of n copies of x (note that $x^0 = \varepsilon$). For all $L \subseteq \Sigma^\star$ and for all $w \in \Sigma^\star$, we denote $Lw := \{xw : x \in L\}$.

Edit Distance. Let x, y be words over Σ. The *edit distance* between x and y is the smallest number of single letter deletions, insertions and substitutions needed to transform x into y. It is also called LEVENSHTEIN *distance* [11] and denoted by $\mathrm{Lev}(x, y)$. For exemple, we have, for all $n \in \mathbb{N}^*$, $\mathrm{Lev}\left((\mathtt{01})^n, (\mathtt{10})^n\right) = 2$. WAGNER & FISHER's algorithm [23] compute the edit distance $\mathrm{Lev}(x, y)$ in polynomial time $\mathrm{O}\left(|x|\,|y|\right)$.

Radius, Centre String and Median String.

Definition 1. *For all languages W over Σ, we denote:*

$$\mathcal{R}(W) := \inf_{\gamma \in \Sigma^\star} \left(\sup_{w \in W} \operatorname{Lev}(\gamma, w) \right)$$

$$\mathcal{S}(W) := \inf_{\mu \in \Sigma^\star} \left(\sum_{w \in W} \operatorname{Lev}(\mu, w) \right)$$

and we call $\mathcal{R}(W)$ the radius *of W.*

A centre *of W is a word γ over Σ such that* $\sup_{w \in W} \operatorname{Lev}(\gamma, w) = \mathcal{R}(W)$.

A median *of W is a word μ over Σ such that* $\sum_{w \in W} \operatorname{Lev}(\mu, w) = \mathcal{S}(W)$.

Our goal is to prove the intractability of the following two problems.

Definition 2. *The* CENTRE STRING *(resp.* MEDIAN STRING*) problem is the decision problem: given a non empty finite language W over Σ and $K \in \mathbb{N}$, is $\mathcal{R}(W) \leq K$ (resp. $\mathcal{S}(W) \leq K$)?*

1.2 Related works

Related problems Computational biology exhibits numerous problems related to MEDIAN STRING. In the more studied ones, the computationally less demanding HAMMING distance replaces the edit distance. One often uses a closest representative of a set of constant length strings that share a biological function. Its computation is known to be NP-hard and is called the CONSENSUS STRING problem (see [19], [13] and references therein). The CONSENSUS PATTERNS and its variants, like the CLOSEST SUBSTRING problem, aim at finding common substrings of a given length in a set of strings, and a model for them. Li et al. [13, 14, 12] exhibit PTAS for all of these, while [5, 4] give exact polynomial time algorithms for some special cases and studied their parameterized complexities. When the Levenshtein distance is used, finding common substrings is termed pattern discovery or motif extraction (see [16, 19]).

Another interesting problem is the DISTINGUISHING STRING SELECTION problem. Given two sets, one of "positive" and the other of "negative" example strings, one has to find a string that is close to the positive, and far from the negative strings (see [10, 5, 2]).

MEDIAN STRING is also important because of its relation with the *Multiple Alignment* problems. Indeed, once given a median string, one can compute an approximate multiple alignment from the pairwise alignments between the median and any string in the input set [7]. Thus, an algorithm for the set median string is used by several authors as an approximation of the MEDIAN STRING. First, Gusfield [6] provides an approximation algorithm for the SUM-OF-PAIRS MULTIPLE ALIGNMENT problem. In this problem, one wishes to minimise the sum of all pairwise alignment costs, hence the name Sum-of-Pairs. Second, Jiang et al. [8] also give an approximation for the TREE MULTIPLE ALIGNMENT problem. Given a tree and sequences associated with its leaves, one has to find sequences for the internal nodes, such that the sum of distances between adjacent strings/nodes over all edges is minimal. They show that associating the set median string to each internal node provides a good approximation scheme. This result was further improved in [24].

Known results. CENTRE STRING and MEDIAN STRING are polynomial in some trivial setup, e.g., with two sequences. Hence, we can deduce from WAGNER & FISCHER's algorithm a dynamic programming algorithm that computes for every non empty, finite language W over Σ a median and a centre of W in $\mathrm{O}\left(\prod_{w \in W} |w|\right)$ time. Thus, for all fixed $n \in \mathbb{N}^*$, the restrictions of CENTRE STRING and MEDIAN STRING to the instances such that $\#W = n$ are polynomial.

In [13], a theorem states that the STAR c-ALIGNMENT problem is NP-hard but no proof is given. The STAR c-ALIGNMENT problem consists in the MEDIAN STRING problem where the number of gaps between any two sequences is constrained to be at most c. Nevertheless, it would not imply that MEDIAN STRING is NP-hard in its general setup. [1]

In [1], it is shown that if Σ is unbounded (resp. $\#\Sigma$ is at least 4) then MEDIAN STRING (resp. CENTRE STRING) is NP-complete. Above, we argue already that especially the NP-completeness of MEDIAN STRING for finite alphabet is an important conjecture. In this work, we demonstrate that both problems are NP-complete even if Σ is binary. Both proofs consist in reducing a well known NP-complete problem, LONGEST COMMON SUBSEQUENCE (LCS), to CENTRE STRING and MEDIAN STRING.

Note that the main difficulty of the NP-completess proof of MEDIAN STRING is that in the instances (W, K), W is a *set* and not a *family*. Hence we do not allow repetitions of words in W, e.g., $\mathcal{S}(\{\varepsilon, \varepsilon, \texttt{01011000}, \texttt{10}, \texttt{10}\}) = \mathcal{S}(\{\varepsilon, \texttt{01011000}, \texttt{10}\})$. Otherwise, a little modification of the proof in [1] shows the NP-completeness of MEDIAN STRING problem for families of words.

We also demonstrate that both CENTRE STRING and MEDIAN STRING are hard in the sense of parametrized complexity with respect to the number of strings. These are important results from a practical point of view since they imply that the existence of an exact polynomial time algorithm is unprobable even if the number of strings is fixed.

Organisation of the paper. We conclude this section with some definitions about parameterized complexity and some known results about the LCS problem. In Section 2, we prove that CENTRE STRING is NP-complete and $W[1]$-hard. In Section 3 we prove that MEDIAN STRING is NP-complete and $W[1]$-hard. We conlude the paper in Section 4 with some open problems.

[1] The authors mention on p. 172 that MEDIAN STRING is NP-hard but without any reference nor proof.

1.3 Parameterized complexity

We give a short introduction to parameterized complexity and the $W[1]$-class (see [3] for a definition of the whole W-hierarchy).

Let L, $L' \subseteq \{\mathtt{0},\mathtt{1}\}^\star \times \mathbb{N}$ be two parameterized binary languages.

We say that L is *fixed parameter tractable* if there exists an algorithm that decides for all $(x,k) \in \{\mathtt{0},\mathtt{1}\}^\star \times \mathbb{N}$ wether $(x,k) \in L$ in time $f(k)\,|x|^c$ where $f : \mathbb{N} \to \mathbb{N}$ is an arbitrary fonction and c an integer constant. We denote FPT the set of all fixed parameter tractable parameterized languages.

We says that L *reduces to L' by a standard parameterized (many to one) reduction* if there are functions f, $m : \mathbb{N} \to \mathbb{N}$, $M : \{\mathtt{0},\mathtt{1}\}^\star \times \mathbb{N} \to \{\mathtt{0},\mathtt{1}\}^\star$ and a constant $c \in \mathbb{N}$ such that for all $(x,k) \in \{\mathtt{0},\mathtt{1}\}^\star \times \mathbb{N} : M(x,k)$ is computable in time $f(k)\,|x|^c$ and $(M(x,k), m(k)) \in L'$ iff $(x,k) \in L$.

We say that a parameterized language L belongs to W[1] if there exists a standard parameterized reduction from the k-Step Halting problem[2] to L. A language L is *$W[1]$-hard* if there exists a standard parameterized reduction from L to the k-Step Halting problem.

The k-Step Halting problem is the parameterized analog of the Turing Machine Acceptance problem, which is the basic generic NP-complete problem. The conjecture FPT $\neq W[1]$ is to parameterized complexity what P $\neq$ NP is to classical computational complexity. Hence, from a practival point of view, $W[1]$-hardness gives a concrete indication that a paramerized problem is fixed parameter untracktable.

1.4 The Longest Common Subsequence Problem.

Let w be a word. A *subword* of w is any word obtained from w by deleting between 0 and $|w|$ letters. We denote by $\mathrm{Sub}(w)$ the set of all subwords of w. For every language L, we denote by $\mathrm{CSub}(L)$ the set of all the words which are common subwords of all the words in L and by $\mathrm{lcs}(L)$ the length of the longest words in $\mathrm{CSub}(L)$. Formally, we have:

$$\mathrm{CSub}(L) = \bigcap_{x \in L} \mathrm{Sub}(x) \quad \text{and} \quad \mathrm{lcs}(L) = \max_{s \in \mathrm{CSub}(L)} |s| \ .$$

For example, for all $n \in \mathbb{N}$, we have, $\mathrm{CSub}\left(\{\mathtt{0}^n\mathtt{1}^n, \mathtt{1}^n\mathtt{0}^n\}\right) = \bigcup_{i=0}^n \left\{\mathtt{0}^i, \mathtt{1}^i\right\}$ and therefore $\mathrm{lcs}\left(\{\mathtt{0}^n\mathtt{1}^n, \mathtt{1}^n\mathtt{0}^n\}\right) = n$.

Definition 3 (Longest Common Subsequence problem (lcs)). *Given a non empty finite language L over Σ and $k \in \mathbb{N}$, is $\mathrm{lcs}(L) \geq k$?*

The intractability of lcs was studied firstly by Maier [15], and later by Pietrzak [20] who slightly improved Maier's results in terms of parameterized complexity :

Theorem 1 (Maier). *If $\#\Sigma$ is at least 2, then* lcs *is* NP*-complete.*

Theorem 2 (Pietrzak). *If $\#\Sigma$ is at least 2, then* lcs *problem parameterized in $\#L$ is $W[1]$-hard.*

[2] also known as Short Turing Machine Acceptance problem

2 NP-completeness of CENTRE STRING

In order to reduce LCS to CENTRE STRING we introduce, like in [1], the following intermediate problem, LCS0, which consists in the restriction of LCS to the instances in which strings have length $2k$, i.e., such that $L \subseteq \Sigma^{2k}$.

Before stating our theorems, we need the following lemma. In substance, it says that if one concatenates a letter a to all words in a language L then the lcs of L increases by one. Indeed, by doing this, one "adds" an a to any maximal common subword of L (one changes $\mathrm{CSub}(L)$ into $\mathrm{CSub}(L) \cup \mathrm{CSub}(L)a$). Thus, the lcs increases by one. The formal proof is left to the reader.

Lemma 1. *For every language L and for every letter a, we have* $\mathrm{lcs}(La) = \mathrm{lcs}(L) + 1$.

It is shown in [1] that if $\#\Sigma$ is at least 4 then LCS0 is NP-complete (note that the proved result is stronger than the one stated in their proposition). We improve this result.

Theorem 3. *The LCS0 problem is NP-hard even if Σ is binary. Moreover, the LCS0 problem parameterized in $\#L$ is $W[1]$-hard.*

Proof. Suppose that Σ is the binary alphabet $\{\mathtt{0}, \mathtt{1}\}$. By Theorem 2, it is sufficient to reduce LCS (parameterized in $\#L$) to LCS0 (parameterized in $\#L$). Let (L, k) be an instance of LCS, L being a non empty finite language and k a positive integer. We construct $\left(\tilde{L}, \tilde{k}\right)$ such that it is an instance of LCS0. Let

$$n := \max_{x \in L} |x|, \quad N := 2k + n, \quad \tilde{k} := k + n,$$
$$L' := \bigcup_{x \in L} \left\{x\mathtt{0}^{N-|x|}, x\mathtt{1}^{N-|x|}\right\} \quad \text{and} \quad \tilde{L} := L'\mathtt{0}^n.$$

We have $L' \subseteq \{\mathtt{0}, \mathtt{1}\}^N$. Therefore, $\tilde{L}$ is a subset of $\{\mathtt{0}, \mathtt{1}\}^{2\tilde{k}}$ and $\left(\tilde{L}, \tilde{k}\right)$ is an instance of LCS0. The transformation of an instance (L, k) of LCS into the instance $\left(\tilde{L}, \tilde{k}\right)$ of LCS0 is polynomial and parameter preserving ($\#\tilde{L} = \#L' = 2\#L$). It remains to prove that

$$\mathrm{lcs}(L) \geq k \iff \mathrm{lcs}\left(\tilde{L}\right) \geq \tilde{k}. \tag{1}$$

Note that for all words u, v, w, $\mathrm{Sub}(wu) \cap \mathrm{Sub}(wv) = \mathrm{Sub}(w)$ if and only if u and v do not share any letter. We have

$$\mathrm{CSub}(L') = \bigcap_{x \in L} \underbrace{\mathrm{Sub}\left(x\mathtt{0}^{N-|x|}\right) \cap \mathrm{Sub}\left(x\mathtt{1}^{N-|x|}\right)}_{\mathrm{Sub}(x)} = \mathrm{CSub}(L)$$

and therefore $\mathrm{lcs}(L) = \mathrm{lcs}(L')$ (the polynomial transformation of (L, k) into (L', k) shows that the restriction of LCS to the instances such that all words in L share the same length is NP-complete).

On the other end, Lemma 1 assures that $\operatorname{lcs}\left(\tilde{L}\right) = \operatorname{lcs}(L') + n$ and therefore:

$$\operatorname{lcs}\left(\tilde{L}\right) = \operatorname{lcs}(L) + n$$

which implies (1). Moreover, as our reduction preserves the parameter $\#L$, the $W[1]$-hardness of LCS0 follows from the $W[1]$-hardness of LCS. $\square$

Now, we have to link the edit distance and the notion of subword to complete the reduction of LCS0 to CENTRE STRING.

Lemma 2. *For all x, $y \in \Sigma^\star$ we have:*

1. $\operatorname{Lev}(x, y) \geq |x| - |y|$,
2. $\operatorname{Lev}(x, y) = |x| - |y|$ *if and only if y is a subword of x.*

Proof. Let $x, y \in \Sigma^\star$ and w.l.o.g. assume x is longer than y. The first statement says that the edit distance is larger than or equal to the length difference of x and y. Clearly, any transformation of x into y has to delete $|x| - |y|$ supernumerary symbols. The second statement says that the equality holds iff y is a subword of x. Again, once the transformation has deleted the $|x| - |y|$ supernumerary symbols, if the resulting subword is y, it means that y is a subword of x, and conversely. $\square$

Theorem 4. *The* CENTRE STRING *problem is* NP*-complete even if Σ is binary. Moreover,* CENTRE STRING *problem parameterized in $\#W$ is $W[1]$-hard.*

Proof. The proof is the same as in [1]. It consists in reducing LCS0 to CENTRE STRING: we transform an instance (L, k) of LCS0 in the instance $(W, K) := (L \cup \{\varepsilon\}, k)$ of CENTRE STRING. The transformation is clearly polynomial and parameter preserving ($\#W \in \{\#L, \#L + 1\}$) and to check the equivalence $\operatorname{lcs}(L) \geq k \iff \mathcal{R}(W) \leq K$, we only need the properties of Lev stated in Lemma 2.

Suppose that Σ is binary. Since in this case LCS0 is NP-complete according to Theorem 3, our reduction shows that CENTRE STRING (parameterized in $\#W$) is $W[1]$-hard too. $\square$

3 NP-completeness of MEDIAN STRING

In order to reduce LCS to MEDIAN STRING, we need to link edit distance and subwords by a tighter inequality than the one provided by Lemma 2. Let x, $y \in \Sigma^\star$ and w.l.o.g. assume $|x| \geq |y|$. The lemma shows that any transformation of x into y contains at least as much operations as the difference between the lengths of x and of its longest common subwords with y. An explanation is as follows. Consider the positions of x that do not belong to a fixed maximal common subword of x and y. All these are either supernumerary and have to be deleted, or differ from the corresponding position in y and need to be substituted.

Lemma 3. *For all x, $y \in \Sigma^\star$, we have*

$$\mathrm{Lev}(x,y) \geq \max\{|x|, |y|\} - \mathrm{lcs}(\{x,y\})$$

Proof. Let $((x_1,y_1),(x_2,y_2),\ldots,(x_n,y_n))$ be an alignment of x and y with cost $\mathrm{Lev}(x,y)$. Remember that $x[i]$ and not x_i denotes the ith symbol of x. We have:

- for all $i \in \mathbb{N}_n^*$, we have $(x_i,y_i) \in ((\Sigma \cup \{\varepsilon\}) \times \Sigma) \cup (\Sigma \times (\Sigma \cup \{\varepsilon\}))$, i.e., in other words a symbol in the alignment can be a single letter or the empty word,
- $x = x_1x_2\ldots x_n$,
- $y = y_1y_2\ldots y_n$,
- denoting by J the set of all $i \in \mathbb{N}_n^*$ such that $x_i \neq y_i$, we have $\mathrm{Lev}(x,y) = \#J$.

As any alignment symbol can be the empty word, we have $n \geq |x_1x_2\ldots x_n| = |x|$ and $n \geq |y_1y_2\ldots y_n| = |y|$, and thus:

$$n \geq \max\{|x|, |y|\}\,.$$

On the other hand, denote $k := \#(\mathbb{N}_n^* \setminus J) = n - \mathrm{Lev}(x,y)$ and let i_1, i_2, $\ldots$, i_k be indexes such that: $\mathbb{N}_n^* \setminus J = \{i_1,i_2,\ldots,i_k\}$ and $i_1 < i_2 < \cdots < i_k$. For all $j \in \mathbb{N}_k^*$, $i \notin J$ means that $x_j = y_j$ and therefore $x_{i_1}x_{i_2}\ldots x_{i_k} = y_{i_1}y_{i_2}\ldots y_{i_k}$ is a subword of x and of y. From that we deduce:

$$\mathrm{lcs}(\{x,y\}) \geq k = n - \mathrm{Lev}(x,y) \geq \max\{|x|, |y|\} - \mathrm{Lev}(x,y)\,.$$

The inequality stated in our lemma follows. □

The inequality stated in Lemma 3 involved only two words. In order to generalize it to many words (Lemma 5), we need the following lemma.

Lemma 4. *For all $\mu \in \Sigma^\star$ and for all X, $Y \subseteq \Sigma^\star$, we have*

$$\mathrm{lcs}\left(\{\mu\} \cup X \cup Y\right) \geq \mathrm{lcs}\left(\{\mu\} \cup X\right) + \mathrm{lcs}\left(\{\mu\} \cup Y\right) - |\mu| \qquad (2)$$

Proof. Let $p := \mathrm{lcs}(\{\mu\} \cup X)$ and $q := \mathrm{lcs}(\{\mu\} \cup Y)$. By hypothesis for $\{\mu\} \cup X$, there exist indexes i_1, i_2, $\ldots$, i_p satisfying $1 \leq i_1 < i_2 < \cdots < i_p \leq |\mu|$ such that $u := \mu[i_1]\mu[i_2]\ldots\mu[i_p] \in \mathrm{CSub}(\{\mu\} \cup X)$. Similarly, for $\{\mu\} \cup Y$, there exist indexes j_1, j_2, $\ldots j_q$ satisfying $1 \leq j_1 < j_2 < \cdots < j_q \leq |\mu|$ such that $v := \mu[j_1]\mu[j_2]\ldots\mu[j_q] \in \mathrm{CSub}(\{\mu\} \cup Y)$.

Setting $I := \{i_1,i_2,\ldots,i_p\}$ and $J := \{j_1,j_2,\ldots,j_q\}$, we see that u and v share a common subword of length $\#(I \cap J)$. It is also a common subword of all words in $\{\mu\} \cup X \cup Y$. From which we deduce

$$\mathrm{lcs}(\{\mu\} \cup X \cup Y) \geq \#(I \cap J) \qquad (3)$$

On the other hand, since I and J are subsets of $\mathbb{N}_{|\mu|}^*$, we have $\#(I \cup J) \leq |\mu|$ and therefore

$$\#(I \cap J) = p + q - \#(I \cup J) \geq p + q - |\mu| \qquad (4)$$

Combining (3) and (4) gives (2) and concludes the proof. □

Lemma 5. *For all $\mu \in \Sigma^\star$ and for all finite languages X over Σ, we have:*

$$\sum_{x \in X} \mathrm{Lev}(\mu, x) + (\#X - 1)\,|\mu| \geq \sum_{x \in X} |x| - \mathrm{lcs}(\{\mu\} \cup X)$$

Proof. We proceed by induction on $\#X$. Assume $\#X = 0$; the inequality holds since both members are equal to $-|\mu|$. When $\#X = 1$, the statement follows from Lemma 3.

Now suppose that $\#X \geq 1$. Let $x_0 \in X$ and let $X' := X \setminus \{x_0\}$. We have

$$\mathrm{Lev}(\mu, x_0) \geq |x_0| - \mathrm{lcs}(\{\mu, x_0\})\,, \tag{5}$$

$$\sum_{x' \in X'} \mathrm{Lev}(\mu, x') + (\#X' - 1)\,|\mu| \geq \sum_{x' \in X'} |x'| - \mathrm{lcs}\left(\{\mu\} \cup X'\right)\,, \tag{6}$$

$$\mathrm{lcs}(\{\mu\} \cup X) \geq \mathrm{lcs}\left(\{\mu\} \cup X'\right) + \mathrm{lcs}(\{\mu, x_0\}) - |\mu|\,. \tag{7}$$

The inequalities (5) and (6) result respectively from Lemma 3 and from the induction hypothesis. Lemma 4 applied with $(X, Y) := (X', \{x_0\})$ yields (7). Adding (5), (6) and the trivial inequality $|\mu| \geq |\mu|$ we obtain

$$\begin{aligned}\sum_{x \in X} \mathrm{Lev}(\mu, x) + (\#X')\,|\mu| &\geq \sum_{x \in X} |x| - \mathrm{lcs}\left(\{\mu\} \cup X'\right) - \mathrm{lcs}(\{\mu, x_0\}) + |\mu| \\ &\geq \sum_{x \in X} |x| - \mathrm{lcs}(\{\mu\} \cup X)\end{aligned}$$

where the last inequality deduces from (7). Since $\#X' = \#X - 1$, this concludes the proof. □

We can now prove the main theorem of this section.

Theorem 5. *The* MEDIAN STRING *problem is* NP*-complete even if Σ is binary. Moreover, the* MEDIAN STRING *problem parameterized in $\#W$ is $W[1]$-hard.*

Proof. Since the edit distance can be computed in polynomial time, it is easy to check that MEDIAN STRING is NP. Now, suppose Σ is the binary alphabet $\{0, 1\}$. The schema of the proof is the following: we reduce LCS to MEDIAN STRINGin order to apply Theorem 2 and conclude. The $W[1]$-hardness of MEDIAN STRING parameterized in $\#W$ is deduced from the one of LCS parameterized in $\#L$ since our reduction is parameter preserving.

Let (L, k) be an instance of LCS, L being a non empty finite language over $\{0, 1\}$ and k a positive integer. We transform (L, k) into the instance (W, K) of MEDIAN STRING, as described below. Let

$$\begin{aligned}&n := \#L\,, \quad C := \textstyle\sum_{x \in L} |x|\,, \quad N := \max\left\{C + \frac{n(n-1)}{2} - k, n - 1\right\}\,, \\ &K := C + (n-1)N - k - \tfrac{n(n-1)}{2}\,, \quad W := L0^N \cup \left\{0^i : i \in \mathbb{N}^*_{n-1}\right\}\,.\end{aligned}$$

This transformation is polynomial and parameter preserving since $\#W = 2(\#L) - 1$. Hence, it remains to prove

$$\mathrm{lcs}(L) \geq k \iff \mathcal{S}(W) \leq K.$$

($\Rightarrow$) Suppose that $\text{lcs}(L) \geq k$. We want to prove that $\mathcal{S}(W) \leq K$. By hypothesis, it exists $s \in \text{CSub}(L)$ such that $|s| = k$. Let $\mu := s\mathtt{0}^N$.

For all $i \in \mathbb{N}^*_{n-1}$, we have $i \leq n-1 \leq N \leq |\mu|_{\mathtt{0}}$, so $\mathtt{0}^i$ is a subword of μ. Hence, by Lemma 2, we have

$$\text{Lev}(\mu, \mathtt{0}^i) = |\mu| - \left|\mathtt{0}^i\right| = k + N - i$$

Moreover, for all $x \in L$, μ is a subword of $x\mathtt{0}^N$; again Lemma 2 applies and we obtain

$$\text{Lev}(\mu, x\mathtt{0}^N) = \left|x\mathtt{0}^N\right| - |\mu| = |x| - k.$$

Using these equalities, we compute the sum

$$\begin{aligned}\sum_{w \in W} \text{Lev}(\mu, w) &= \sum_{x \in L} \text{Lev}(\mu, x\mathtt{0}^N) + \sum_{i=1}^{n-1} \text{Lev}(\mu, \mathtt{0}^i) \\ &= \sum_{x \in L} (|x| - k) + \sum_{i=1}^{n-1} (k + N - i) \\ &= C - nk + (n-1)k + (n-1)N - \frac{n(n-1)}{2} \\ &= K\end{aligned}$$

and we obtain $\mathcal{S}(W) \leq K$.

($\Leftarrow$) Conversely, assume $\mathcal{S}(W) \leq K$. We will show that $\text{lcs}(L) \geq k$. By hypothesis, it exists $\mu \in \{\mathtt{0}, \mathtt{1}\}^\star$ such that $\sum_{w \in W} \text{Lev}(\mu, w) \leq K$. First, we prove that $|\mu|_{\mathtt{0}} \geq n-1$. For this, we note that for all words u, v, $\text{Lev}(u, v)$ is greater or equal to $|v|_{\mathtt{0}} - |u|_{\mathtt{0}}$. Hence, for all $x' \in L\mathtt{0}^N$, we have

$$N - |\mu|_{\mathtt{0}} \leq |x'|_{\mathtt{0}} - |\mu|_{\mathtt{0}} \leq \text{Lev}(\mu, x')$$

so by summing over $x' \in L\mathtt{0}^N$, we get

$$nN - n\,|\mu|_{\mathtt{0}} \leq \sum_{x' \in L\mathtt{0}^N} \text{Lev}(\mu, x') \leq K$$

and:

$$n\,|\mu|_{\mathtt{0}} \geq nN - K \geq n(n-1)$$

which is equivalent to $|\mu|_{\mathtt{0}} \geq n-1$. This implies that for all $i \in \mathbb{N}^*_{n-1}$, $\mathtt{0}^i$ is a subword of μ, and so $\text{Lev}(\mu, \mathtt{0}^i) = |\mu| - i$. Thus,

$$\sum_{i=1}^{n-1} \text{Lev}(\mu, \mathtt{0}^i) = \sum_{i=1}^{n-1} |\mu| - \sum_{i=1}^{n-1} i = (n-1)\,|\mu| - \frac{n(n-1)}{2}.$$

We can now write

$$\begin{aligned}\sum_{w \in W} \text{Lev}(\mu, w) &= \sum_{x' \in L\mathtt{0}^N} \text{Lev}(\mu, x') + (n-1)\,|\mu| - \frac{n(n-1)}{2} \\ &\geq \sum_{x' \in L\mathtt{0}^N} |x'| - \text{lcs}\left(\{\mu\} \cup L\mathtt{0}^N\right) - \frac{n(n-1)}{2} \qquad (8)\end{aligned}$$

where the application of Lemma 3 with $X := L\mathtt{0}^N$ yields the last inequality.

On the other hand, we have:

$$\sum_{x' \in L\mathtt{0}^N} |x'| = \sum_{x \in L} \left|x\mathtt{0}^N\right| = \sum_{x \in L} (|x| + N) = C + nN \tag{9}$$

and by Lemma 1

$$\operatorname{lcs}(\{\mu\} \cup L\mathtt{0}^N) \leq \operatorname{lcs}(L\mathtt{0}^N) = \operatorname{lcs}(L) + N\,. \tag{10}$$

By hypothesis, $K \geq \mathcal{S}(W) \geq \sum_{w \in W} \operatorname{Lev}(\mu, w)$; combining this with (8), (9) and (10) yields

$$\begin{aligned} K \geq \textstyle\sum_{w \in W} \operatorname{Lev}(\mu, w) &\geq (C + nN) - (\operatorname{lcs}(L) + N) - \tfrac{n(n-1)}{2} \\ \Rightarrow \operatorname{lcs}(L) &\geq C + (n-1)N - \tfrac{n(n-1)}{2} - K = k\,. \end{aligned}$$

This concludes the proof. □

4 Open problems

Possible improvements of our results We prove in this paper that CENTRE STRING and MEDIAN STRING are NP-complete, according to the "non weighted" edit distance. Of course, in general for any distance function, the problem remains NP-complete since the edit distance is a particular case. Now, for a fixed alphabet-weighted edit distance the problem may not be NP-complete. To prove the NP-completeness in such setup is not trivial since, since, if $\Sigma = \{\mathtt{0}, \mathtt{1}\}$ and if the scores of insertions/deletions of `0` and of `1` are not equal then our reductions do not hold.

Approximation Although CENTRE STRING and MEDIAN STRING are NP-complete, there exist approximation algorithms with bounded errors [7] and heuristic algorithms [9], [18]. For example, given a finite language W over Σ, *set center* (resp. set median) of W can be found in polynomial time and is an approximate centre (resp. median) of W with performance ratio 2 (resp. $2 - \frac{2}{\#W}$). Note that we call set center of W any word that minimizes the maximum of the distances to strings in the set W *and* belongs to W. An open question subsists: do these problems admit Polynomial Time Approximation Schemes?

Acknowledgment

The authors thank Colin De La Higuera for reading a preliminary version of this work and Jens Gramm for introducing us to parameterized complexity. This is supported by the CNRS STIC Specific Action "Algorithmes et Séquences", by the ACI Jeunes Chercheurs "Combinatoire des mots multidimensionnels, pavages et numération" and by the Math STIC project 2001 "Mots : de la combinatoire la dynamique symbolique".

References

1. C. de la Higuera and F. Casacuberta. Topology of strings: Median string is NP-complete. *Theoretical Computer Science*, 230:39–48, 2000.
2. X. Deng, G. Li, Z. Li, B. Ma, and L. Wang. A ptas for distinguishing (sub)string selection. In *ICALP*, pages 740–751, 2002.
3. R. G. Downey and M. R. Fellows. *Parameterized Complexity.* Springer, 1999.
4. Michael R. Fellows, Jens Gramm, and Rolf Niedermeier. On the parameterized intractability of CLOSEST SUBSTRING and related problems. In *Symposium on Theoretical Aspects of Computer Science*, pages 262–273, 2002.
5. Jens Gramm, Rolf Niedermeier, and Peter Rossmanith. Exact solutions for CLOSEST STRING and related problems. In *ISAAC*, volume 2223 of *LCNS*, pages 441–453, 2001.
6. Dan Gusfield. Efficient methods for multiple sequence alignment with guaranteed error bounds. *Bull. Math. Biol.*, 55:141–154, 1993.
7. Dan Gusfield. *Algorithms on strings, trees, and sequences: computer science and computational biology.* Cambridge University Press, 1997.
8. Tao Jiang, Eugene L. Lawler, and Lusheng Wang. Approximation algorithms for tree alignment with a given phylogeny. *Algorithmica*, 16(3):302–315, 1996.
9. T. Kohonen. Median strings. *Pattern Recognition Letters*, 3:309–313, 1985.
10. J. Lanctot, M. Li, B. Ma, S. Wang, and L. Zhang. Distinguishing string selection problems. In *SODA: ACM-SIAM Symposium on Discrete Algorithms*, 1999.
11. V. I. Levenshtein. Binary codes capable of correcting deletions, insertions and Reverseals. *Cybernetics and Control Theory*, 10(8):707–710, 1966.
12. M. Li, B. Ma, and L. Wang. On the closest string and substing problems. *Journal of the ACM*, 49(2):157–171, 2002.
13. Ming Li, Bin Ma, and Lusheng Wang. Finding similar regions in many strings. In *Proceedings of the 31st Annual ACM Symposium on Theory of Computing (STOC'99)*, pages 473–482, 1999.
14. Bin Ma. A polynomial time approximation scheme for the closest substring problem. In *CPM*, volume 1848 of *LNCS*, pages 99–107, 2000.
15. D. Maier. The complexity of some problems on subsequences and supersequences. *Journal of the Association for Computing Machinery*, 25:322–336, 1978.
16. L. Marsan and M. F. Sagot. Algorithms for extracting structured motifs using a suffix tree with an application to promoter and regulatory site consensus identification. *J Comput Biol*, 7(3-4):345–62, 2000.
17. C. D. Martinez, A. Juan, and F. Casacuberta. Improving classification using median string and nn rules. In *Spanish Symp. on Pattern Recognition and Image Analysis*, pages 391–395, 2001.
18. C. D. Martinez-Hinarejos, A. Juan, and F. Casacuberta. Use of median string for classification. In *15th International Conference on Pattern Recognition*, volume 2, pages 907–910, september 2000.
19. Pavel Pevzner. *Computational Molecular Biology.* MIT Press, 2000.
20. Krzysztof Pietrzak. On the parameterized complexity of the fixed alphabet shortest common supersequence and longest common subsequence problems. *Journal of Computer and System Sciences*, 2003. to appear.
21. J. S. Sim and K. Park. The consensus string problem for a metric is NP-complete. In R. Raman and J. Simpson, editors, *Proceedings of the 10th Australasian Workshop On Combinatorial Algorithms*, pages 107–113, Perth, WA, Australia, 1999.

22. David J. States and Pankaj Agarwal. Compact encoding strategies for DNA sequence similarity search. In *Proceedings of the Fourth International Conference on Intelligent Systems for Molecular Biology*, pages 211–217. AAAI Press, 1996.
23. Robert A. Wagner and Michael J. Fischer. The string-to-string correction problem. *Journal of the ACM (JACM)*, 21(1):168–173, 1974.
24. L. Wang and D. Gusfield. Improved approximation algorithms for tree alignment. *J. Algorithms*, 25(2):255–273, 1997.