Available online at www.sciencedirect.com
SCIENCE DIRECT®
ELSEVIER
Journal of Discrete Algorithms 3 (2005) 390–415
JOURNAL OF DISCRETE ALGORITHMS
www.elsevier.com/locate/jda

Hardness results for the center and median string problems under the weighted and unweighted edit distances

François Nicolas *, Eric Rivals

L.I.R.M.M., CNRS U.M.R. 5506, 161 rue Ada, F-34392 Montpellier Cedex 5, France

Available online 17 September 2004

Abstract

Given a finite set of strings, the MEDIAN STRING problem consists in finding a string that minimizes the sum of the edit 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 bounded and even binary alphabets. In this work, we provide an answer to this question and also give the complexity of the related CENTER STRING problem. Moreover, we study the parameterized complexity of both problems with respect to the number of input strings. In addition, we provide an algorithm to compute an optimal center under a weighted edit distance in polynomial time when the number of input strings is fixed.

Keywords: Concensus string; Multiple alignment; Tree alignment; NP-complete; Parameterized complexity; LCS

* Corresponding author.
E-mail addresses: nicolas@lirmm.fr (F. Nicolas), rivals@lirmm.fr (E. Rivals).

1570-8667/$ – see front matter
doi:10.1016/j.jda.2004.08.015

1. Introduction

Given an alphabet Σ, a set W of strings over Σ, and an edit distance between strings, the problem of finding a string over Σ that minimizes the sum of edit distances to the strings of W is called the MEDIAN STRING problem. Alternative terminologies include the GENERALIZED MEDIAN STRING problem [2], the STAR ALIGNMENT problem [12], the CONSENSUS ALIGNMENT problem [13] and also the STEINER STRING problem [8].

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 [7–10,19,20]. 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 [8, 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. In other words, this string summarizes the information shared by the 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 [8,22]) or used for classification purposes (in speech recognition [10], classification [8] and computational biology [8,22]).

In computational biology, computing a median string of a set W is equivalent to solving a MULTIPLE ALIGNMENT problem, which is one of the most important and difficult problems in this area [8]. In practice, MULTIPLE ALIGNMENT may be easier when the evolutionary relationships of the species bearing the sequence are known. The input of the so called TREE ALIGNMENT problem is then a set of strings W and a tree whose leaves are labeled by the strings of W. The objective is to find strings for the internal nodes, such that the sum of edit distances between adjacent strings/nodes over all edges is minimal. A special case of tree of theoretical importance is the *star tree*: there the computed internal string is a median string [8,24]. As already mentioned, the median string can serve as consensus of the strings in W, especially if the strings occupy homogeneously the metric space around the median. Unfortunately, in practical applications, the strings in W are not a uniform sample of the evolutionary diversity: some evolutionary families of strings in W are more represented than others. In such cases, minimizing the sum of the edit distances results in a biased alignment and consensus (see [1] for a discussion about this matter). Minimizing the maximum edit distance, i.e., computing a center string, produces solutions that reflect more faithfully the variations in W. For this purpose, Ravi and Kececioglu introduced in 1995 a variant of the TREE ALIGNMENT problem with this objective. It is called the BOTTLENECK TREE ALIGNMENT. When the input tree is a star, the BOTTLENECK TREE ALIGNMENT problem is equivalent to the CENTER STRING problem.

In [2], it is shown that CENTER STRING and MEDIAN STRING are NP-hard for alphabets of size at least 4 and for unbounded alphabets, respectively. 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 proteins or purine-pyrimidine composition for nucleic acids. For instance, it is the case in some protocols to identify similar DNA sequences [27]. The important practical ques-

tion is whether CENTER STRING and MEDIAN STRING are NP-hard for finite and even binary alphabets. In the above-mentioned article, these questions remain open [2, p. 48]. These conjectures are solved in this paper. Additionally, 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 CENTER 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 $m, n \in \mathbb{N}$, we denote by $[m, n]$ the set $\{k \in \mathbb{N}: m \leqslant k \leqslant n\}$. For every finite set X we denote by $\#X$ the cardinality of X.

1.1.1. Strings

An *alphabet* is a non-empty set of *letters*. In the sequel, Σ always denotes an alphabet. A *string* over Σ is a finite sequence of elements of Σ. The set of all strings 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 string*. Given two strings x and y, we denote by xy the *concatenation* of x and y. For all $L \subseteq \Sigma^\star$ and for all $w \in \Sigma^\star$, we denote $\{xw: x \in L\}$ by Lw. For all $n \in \mathbb{N}$, we denote by x^n the *nth power* of x, i.e., the concatenation of n copies of x (note that $x^0 = \varepsilon$). For a string w, $|w|$ denotes the length of w. For a language $L \subseteq \Sigma^\star$, $|L|$ denotes $\sum_{x \in L} |x|$. For all $i \in [1, |w|]$, $w[i]$ denotes the ith letter of w: $w = w[1]w[2] \ldots w[|w|]$. For all $a \in \Sigma$, $|w|_a := \#\{i \in [1, |w|] : w[i] = a\}$ denotes the number of *occurrences* of the letter a in w.

1.1.2. Edit distance

Definition 1 (*Metric*). Let E be a set and d be a mapping from $E \times E$ onto $\mathbb{R}$. We say that d is a *metric* over E iff for any $x, y, z \in E$, d fulfills the following conditions

- $d(x, y) \geqslant 0$ (positivity),
- $d(x, y) = 0$ if and only if $x = y$ (separation),
- $d(x, y) = d(y, x)$ (symmetry),
- $d(x, z) \leqslant d(x, y) + d(y, z)$ (triangular inequality).

The *edit operations* are single letter deletions, insertions and substitutions. Let δ be an integer valued metric over $\Sigma \cup \{\varepsilon\}$: δ can be viewed as a cost function over the edit operations (a *penalty matrix*). Hence, for all $a, b \in \Sigma$, the substitution from a into b costs $\delta(a, b)$, the deletion of an a costs $\delta(a, \varepsilon)$ and the insertion of a b costs $\delta(\varepsilon, b)$. The cost of a sequence of edit operations is the sum of the costs of its terms.

The *δ-weighted edit distance* between two strings x and y is the cost of the cheapest sequence of edit operations needed to transform x into y. It is also the cost the cheapest alignment of x and y. Let us denote by $d_E : \Sigma^\star \times \Sigma^\star \to \mathbb{N}$ the δ-weighted edit distance: since δ is a metric, d_E is also a metric and for all $a, b \in \Sigma \cup \{\varepsilon\}$, we have $d_E(a, b) = \delta(a, b)$.

Wagner and Fisher's algorithm [28] computes the weighted edit distance $d_E(x, y)$ in polynomial time $O(|x||y|)$. It proceeds by dynamic programming and can be easily deduced from Theorem A.1 below.

If δ is such that:

$$\forall a, b \in \Sigma \cup \{\varepsilon\} \quad \delta(a,b) = \begin{cases} 1 & \text{if } a \neq b, \\ 0 & \text{otherwise} \end{cases}$$

(that is if each edit operation has unitary cost) then the δ-weighted edit distance is called *Levenshtein distance* (or sometimes *unweighted edit distance*) and is denoted by d_L.

1.1.3. Radius, center string and median string

Definition 2. Let d_E a weighted edit distance. For all languages W over Σ, we denote:

$$\mathcal{R}(W) := \inf_{\gamma \in \Sigma^\star} \Big(\sup_{w \in W} d_E(\gamma, w) \Big),$$

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

and we call $\mathcal{R}(W)$ the *radius* of W (under d_E). A *center* of W (under d_E) is a string γ over Σ such that $\sup_{w \in W} d_E(\gamma, w) = \mathcal{R}(W)$. A *median* of W (under d_E) is a string μ over Σ such that $\sum_{w \in W} d_E(\mu, w) = \mathcal{S}(W)$.

If W is infinite then the radius of W and $\mathcal{S}(W)$ are infinite. We study only the finite case. Let us consider the case $\#W = 2$. Let $x, y \in \Sigma^\star$.

- Under any weighted edit distance d_E, $\mathcal{S}(\{x, y\}) = d_E(x, y)$. Any string on an optimal alignment path between x and y is a median of $\{x, y\}$, including x and y themselves.
- Under Levenshtein distance, $\mathcal{R}(\{x, y\}) = \lceil d_L(x, y)/2 \rceil$. Any string γ on an optimal alignment path between x and y such that $d_L(\gamma, x)$ or $d_L(\gamma, y)$ equals $\lceil d_L(x, y)/2 \rceil$ is a center of $\{x, y\}$. In this case, a center is always a median. Given an optimal alignment between x and y, a center can be computed in linear time.

Example 1. Let $\Sigma := \{\mathtt{0}, \mathtt{1}\}$ and $W := \{\mathtt{0}^N, \mathtt{1}^N\}$ where N denotes an even integer. Under Levenshtein distance,

- $\mathcal{S}(W) = d_L(\mathtt{0}^N, \mathtt{1}^N) = N$ and the medians of W are the strings $\mu \in \{\mathtt{0}, \mathtt{1}\}^\star$ such that $|\mu|_{\mathtt{0}} + |\mu|_{\mathtt{1}} = N$ and,
- $\mathcal{R}(W) = N/2$ and the centers of W are the strings $\gamma \in \{\mathtt{0}, \mathtt{1}\}^\star$ such that $|\gamma|_{\mathtt{0}} = |\gamma|_{\mathtt{1}} = N/2$.

Example 2. Let $\Sigma := \{\mathtt{0}, \mathtt{1}\}$ and $W := \{(\mathtt{01})^N, (\mathtt{10})^N\}$ where N denotes a positive integer. Under Levenshtein distance, both strings $(\mathtt{01})^{N-1}\mathtt{0}$ and $\mathtt{1}(\mathtt{01})^{N-1}$ are centers and medians of W ($\mathcal{S}(W) = d_L((\mathtt{01})^N, (\mathtt{10})^N) = 2$ and $\mathcal{R}(W) = 1$).

Example 3. Let

$$\Sigma := \{a_0, a_1, \ldots, a_\sigma, b, c\} \quad \text{and} \quad W := \{a_0 b^N, a_1 c^N, a_2 c^N, \ldots, a_\sigma c^N\}$$

where $\sigma \in \mathbb{N} \setminus \{0, 1\}$, N denotes an even positive integer, and $a_0, a_1, \ldots, a_\sigma$, b and c are distinct letters. Under Levenshtein distance,

– $a_i c^N$ is a median of W for all $i \in [0, \sigma]$ ($\mathcal{S}(W) = N + \sigma$), and
– $b^{N/2} c^{N/2}$ is a center of W for all $i \in [0, \sigma]$ ($\mathcal{R}(W) = N/2 + 1$).

In this example, no word is both a center and a median of W.

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

Definition 3. The CENTER STRING (resp. MEDIAN STRING) problem is the decision problem: "given a non-empty finite language W over Σ and $K \in \mathbb{N}$, is $\mathcal{R}(W) \leqslant K$ (resp. $\mathcal{S}(W) \leqslant K$)?"

1.2. Related works

1.2.1. Related problems

Computational biology exhibits numerous problems related to MEDIAN STRING and CENTER 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. For instance, under the Hamming distance MEDIAN STRING is polynomial, while CENTER STRING is known to be NP-hard [11] and is called CLOSEST STRING.

The CONSENSUS PATTERN problem (also called the CONSENSUS STRING problem in [22]) 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. [12,14,16] exhibit PTAS for all of these, while [5,6] give exact polynomial time algorithms for some special cases and study their parameterized complexities. Another interesting problem is the DISTINGUISHING SUBSTRING 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 [3,6,11]).

When the edit distance is used, finding common substrings is termed pattern discovery or motif extraction (see [18,22]).

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 [8]. Thus, an algorithm for the set median string is used by several authors as an approximation of the *Multiple Alignment* problem. First, Gusfield [7] provides an approximation algorithm for the SUM-OF-PAIRS MULTIPLE ALIGNMENT problem. In this problem, one wishes to minimize the sum of all pairwise alignment costs, hence the name Sum-of-Pairs. Second, Jiang et al. [9] also give an approximation for the TREE ALIGNMENT problem. They show that associating the set median string to each internal node provides a good approximation scheme. This result is further improved in [29].

An approximation algorithm for BOTTLENECK TREE ALIGNMENT is given in [24].

1.2.2. Known results

MEDIAN STRING is polynomial for two strings (any of the input string is a median). Moreover, a dynamic programming algorithm computes for every non-empty, finite lan-

guage W over Σ a median of W in $O((\#W)2^{\#W}\prod_{w\in W}|w| + (\#W)(\#\Sigma)^{\#W+1})$ time [25]. Thus, if the number of input strings is fixed, MEDIAN STRING is polynomial. In [24], an exact dynamic programming algorithm is sketched for BOTTLENECK TREE ALIGNMENT under Levenshtein distance.

In [13], it is shown that the CONSENSUS c-ALIGNMENT problem is NP-hard for alphabet size 4. The CONSENSUS c-ALIGNMENT problem consists in the MEDIAN STRING problem where the number of gaps between any two strings is constrained to be at most c. Nevertheless, it would not imply that MEDIAN STRING is NP-hard in its general setup.

1.2.3. Our contribution

In [2], it is shown that if Σ is unbounded (resp. $\#\Sigma$ is at least 4) then MEDIAN STRING (resp. CENTER STRING) is NP-complete. In [26], it is shown that MEDIAN STRING is NP-hard for alphabet of size 7 and under a conveniently weighted edit distance. Above, we argue already that the NP-completeness of MEDIAN STRING for finite alphabet is an important conjecture. In this work, we demonstrate that both problems are NP-complete under Levenshtein distance even if Σ is binary. Both proofs consist in reducing a well-known NP-complete problem, LONGEST COMMON SUBSEQUENCE (LCS), to CENTER STRING and MEDIAN STRING.

We also demonstrate that both CENTER STRING and MEDIAN STRING are hard in the sense of parameterized complexity with respect to the number of input strings. These are important results from a practical point of view since they rule out the existence of an exact algorithm solving one of our problems in time $O(f(\#W)|W|^c)$ where $f:\mathbb{N}\to\mathbb{N}$ is an *arbitrary* function and c is a constant. Unlike CLOSEST STRING that is FPT with respect to the number of input strings, CENTER STRING is W[1]-hard for this parameter.

Moreover, we extend the intractability results for CENTER STRING for a large class of weighted edit distances satisfying natural assumptions.

1.2.4. Organization 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 under Levenshtein distance, CENTER STRING and MEDIAN STRING over binary alphabets are NP-complete and W[1]-hard with respect to the number of input strings. In Section 3 we generalize the results for CENTER STRING obtained in Section 2: we prove that under *any* weighted edit distance satisfying Property 1, CENTER STRING over binary alphabets is NP-complete and W[1]-hard with respect to the number of input strings. In Appendix A, we show that CENTER STRING is polynomial if the number of input strings and the weighted edit distance are fixed. Note that in the same case, the counterpart for MEDIAN STRING has been known for a long time [25]. We conclude the paper in Section 4 with some open problems.

1.3. Parameterized complexity

We give a short introduction to parameterized complexity and the W[1]-class (see [4] for a definition of the whole W-hierarchy).

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

We say that L is *fixed parameter tractable* if there exists an algorithm that, for all $(x,k) \in \{0,1\}^\star \times \mathbb{N}$, decides whether $(x,k) \in L$ in time $f(k)|x|^c$ where $f : \mathbb{N} \to \mathbb{N}$ is an arbitrary function and c an integer constant. We denote by 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 $f, m : \mathbb{N} \to \mathbb{N}$, $M : \{0,1\}^\star \times \mathbb{N} \to \{0,1\}^\star$ and a constant $c \in \mathbb{N}$ such that for all $(x,k) \in \{0,1\}^\star \times \mathbb{N}$: $M(x,k)$ 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 L to the k-STEP HALTING problem.[1] A language L is W[1]-*hard* if there exists a standard parameterized reduction from the k-STEP HALTING problem to L.

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 practical point of view, W[1]-hardness gives a concrete indication that a parameterized problem is fixed parameter intractable.

1.4. The longest common subsequence problem

Let w be a string. A *subsequence* of w is any string obtained from w by deleting between 0 and $|w|$ letters. We denote by $\mathrm{Sub}(w)$ the set of all subsequences of w. For every non-empty language L, we denote by $\mathrm{CSub}(L)$ the set of all the strings which are common subsequences of all the strings in L, and by $\mathrm{lcs}(L)$ the length of the longest strings 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, $\mathrm{CSub}(\{0^n1^n, 1^n0^n\}) = \bigcup_{i=0}^{n} \{0^i, 1^i\}$ and therefore $\mathrm{lcs}(\{0^n1^n, 1^n0^n\}) = n$.

Definition 4 (*Longest Common Subsequence problem* (*LCS*)). Given a non-empty finite language L over Σ and $k \in \mathbb{N}$, is $\mathrm{lcs}(L) \geqslant k$?

The intractability of LCS was studied firstly by Maier and later by Pietrzak, who improves Maier's results in terms of parameterized complexity:

Theorem 1. *Suppose Σ is a binary alphabet.*

The LCS *problem is* NP*-complete* [17].

The LCS *problem, parameterized in #L, is* W[1]*-hard* [23].

[1] Also known as SHORT TURING MACHINE ACCEPTANCE problem.

2. Hardness of CENTER STRING and MEDIAN STRING over binary alphabets under Levenshtein distance

In this section, we demonstrate the NP-completeness and W[1]-hardness with respect to the number of input strings of CENTER STRING and MEDIAN STRING under Levenshtein distance. These results already appear in [21]. We start with the less technical of the two proofs, the one concerning CENTER STRING (Theorem 3). Similar ideas are used for MEDIAN STRING (Theorem 4). As a by-product, we also show the hardness of a restriction LCS to instances in which all input strings share the same length.

2.1. Hardness of CENTER STRING under Levenshtein distance

In order to reduce LCS to CENTER STRING we introduce, like in [2], the following intermediate problem, LCS0, which consists in the restriction of LCS to the instances in which input strings have length $2k$.

Before stating our theorems, we need the following lemmas. In substance, the first lemma says that if one concatenates a letter a to all strings in a language L, then the lcs of L increases by one. Indeed, by doing this, one "adds" an a to any maximal common subsequence 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* $\mathrm{lcs}(La) = \mathrm{lcs}(L) + 1$.

The following lemma shows that given a language L and two different letters of Σ, one can design another language L' by associating to each string x of L two strings such that their only common subsequence is x. This is made by concatenating to x two suffixes sharing no common letters. It follows that the lcs of L and of L' have equal length although the strings of L' are arbitrarily longer than the ones of L. This lemma is novel compared to the proof in [2] and allows us to exhibit a reduction of LCS0 to LCS that remains valid in the case of binary alphabets, and preserves the parameter.

Lemma 2. *Let L be a language over Σ, $a, b \in \Sigma$ such that $a \neq b$, and $(m_x)_{x\in L}$ and $(n_x)_{x\in L}$ two lists of positive integers each associated with a string of L. Let us define the language L' by $L' := \bigcup_{x\in L}\{xa^{m_x}, xb^{n_x}\}$. Then, the longest common subsequences of L and of L' share the same length, i.e.,* $\mathrm{lcs}(L) = \mathrm{lcs}(L')$.

Proof. For all strings $u, v, w \in \Sigma^\star$, $\mathrm{Sub}(wu) \cap \mathrm{Sub}(wv) = \mathrm{Sub}(w)$ if and only if u and v do not share any letter. We have

$$\mathrm{CSub}(L') = \bigcap_{x\in L} \underbrace{\mathrm{Sub}(xa^{m_x}) \cap \mathrm{Sub}(xb^{n_x})}_{\mathrm{Sub}(x)} = \mathrm{CSub}(L)$$

and therefore $\mathrm{lcs}(L) = \mathrm{lcs}(L')$. □

It is shown in [2] that if $\#\Sigma$ is at least 4 then LCS0 is NP-complete. (The result they prove is stronger than the one stated in [2, Proposition 1].) We improve this result.

Theorem 2. *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 1, 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 $(\tilde{L},\tilde{k})$ such that it is an instance of LCS0. In our construction, we introduce an intermediate language L' whose strings all have intermediate length N. L' is constructed from L as in Lemma 2 with appropriate m_x's and n_x's in order to obtain strings of the same length N. With n set as the length of the longest string in L, the final language $\tilde{L}$ is made by concatenating $\mathtt{0}^n$ to all strings in L'. This forces the lcs of $\tilde{L}$ to be larger than or equal to n. We set $\tilde{k} := k+n$ such that if (L,k) has a solution of length k, $(\tilde{L},\tilde{k})$ has a solution of length $\tilde{k}$. Let

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

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

$$\mathrm{lcs}(L) \geqslant k \quad \Leftrightarrow \quad \mathrm{lcs}(\tilde{L}) \geqslant \tilde{k}. \tag{1}$$

First, Lemma 2 applied to L and L' yields $\mathrm{lcs}(L) = \mathrm{lcs}(L')$. As a corollary, the polynomial reduction of (L,k) to (L',k) shows that the restriction of LCS to the instances such that all strings in L share the same length is NP-complete and W[1]-*hard* with respect to $\#L'$.

On the other end, Lemma 1 implies that $\mathrm{lcs}(\tilde{L}) = \mathrm{lcs}(L') + n$ and therefore: $\mathrm{lcs}(\tilde{L}) = \mathrm{lcs}(L) + n$ which implies (1). □

Now, we have to relate the Levenshtein distance and the notion of subsequence to complete the reduction of LCS0 to CENTER STRING.

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

(i) $d_L(x,y) \geqslant |x| - |y|$,
(ii) $d_L(x,y) = |x| - |y|$ *if and only if* y *is a subsequence 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 subsequence of x. Again, once the transformation has deleted the $|x| - |y|$ supernumerary symbols, if the resulting subsequence is y, it means that y is a subsequence of x, and conversely. □

Theorem 3. *The* CENTER STRING *problem under Levenshtein distance is* NP*-complete even if* Σ *is binary. Moreover,* CENTER STRING *under Levenshtein distance, parameterized in* $\#W$*, is* W[1]*-hard.*

Proof. The proof is the same as in [2]. It consists in reducing LCS0 to CENTER STRING: we transform an instance (L, k) of LCS0 into the instance $(W, K) := (L \cup \{\varepsilon\}, k)$ of CENTER STRING. The transformation is polynomial and parameter preserving (since $\#W \in \{\#L, \#L + 1\}$). The equivalence

$$\operatorname{lcs}(L) \geqslant k \quad \Leftrightarrow \quad \mathcal{R}(W) \leqslant K,$$

follows from the properties of d_L stated in Lemma 3.

($\Rightarrow$) First, assume $\operatorname{lcs}(L) \geqslant k$. We show that $\mathcal{R}(W) \leqslant K$. By hypothesis, it exists $s \in \operatorname{CSub}(L)$ such that $|s| = k$. Statement (ii) of Lemma 3 implies that for all $x \in L$: $d_L(x, s) = |x| - |s| = 2k - k = k$. As $d_L(\varepsilon, s) = |s| = k = K$, for any $x \in W$, it follows that $d_L(x, s) = K$ and thus, $\mathcal{R}(W) \leqslant k = K$.

($\Leftarrow$) Now, assume that $\mathcal{R}(W) \leqslant K$. We show $\operatorname{lcs}(L) \geqslant k$, i.e., it exists an s in $\operatorname{CSub}(L)$ such that $|s| \geqslant k$. By hypothesis, it exists $s \in \{\mathtt{0}, \mathtt{1}\}^\star$ such that for any $w \in W$ we have $d_L(w, s) \leqslant K = k$. As ε belongs to W, we know by hypothesis that $|s| = d_L(\varepsilon, s) \leqslant k$. By definition of the radius, for any $x \in L$:

$$\begin{aligned} k &\geqslant d_L(x, s) \\ &\geqslant |x| - |s| \quad \text{by statement (i) of Lemma 3} \\ &= 2k - |s| \quad \text{because } x \in L \\ &\geqslant 2k - k \quad \text{since } |s| \leqslant k \\ &= k. \end{aligned}$$

It follows that the previous inequalities are in fact equalities. Thus, by statement (ii) of Lemma 3, $d_L(x, s) = |x| - |s|$ implies that s is a subsequence of x. Moreover, $2k - |s| = k$ and therefore $|s| = k$, which completes the proof. □

2.2. *Hardness of* MEDIAN STRING *under Levenshtein distance*

In order to reduce LCS to MEDIAN STRING, we need to link Levenshtein distance and subsequences by a tighter inequality than the one provided by Lemma 3. Let $x, y \in \Sigma^\star$ and w.l.o.g. assume $|x| \geqslant |y|$. Lemma 4 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 subsequences with y. An explanation is as follows. Consider the positions of x that do not belong to a fixed maximal common subsequence 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 4. *For all* $x, y \in \Sigma^\star$*, we have*

$$d_L(x, y) \geqslant \max\bigl\{|x|, |y|\bigr\} - \operatorname{lcs}\bigl(\{x, y\}\bigr).$$

Proof. Let $((x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n))$ be an alignment of x and y with optimal cost $d_L(x, y)$. For a definition of an alignment, we refer the reader to [8]. Remember that $x[i]$ and not x_i denotes the ith symbol of x. We have

- $(x_i, y_i) \in ((\Sigma \cup \{\varepsilon\}) \times \Sigma) \cup (\Sigma \times (\Sigma \cup \{\varepsilon\}))$ for all $i \in [1, n]$ (a symbol in the alignment is a single letter or the empty string),
- $x = x_1 x_2 \ldots x_n$,
- $y = y_1 y_2 \ldots y_n$,
- $d_L(x, y) = \#J$ where J is the set of all $i \in [1, n]$ such that $x_i \neq y_i$.

Denote $k := \#([1, n] \setminus J)$. Let $i_1, i_2, \ldots, i_k$ be indexes such that: $[1, n] \setminus J = \{i_1, i_2, \ldots, i_k\}$ and $i_1 < i_2 < \cdots < i_k$. For all $i \in [1, n]$, $i \notin J$ means that $x_i = y_i$ and therefore $x_{i_1} x_{i_2} \ldots x_{i_k} = y_{i_1} y_{i_2} \ldots y_{i_k}$ is a subsequence of x and of y. From that we deduce:

$$\operatorname{lcs}\bigl(\{x, y\}\bigr) \geqslant k = n - d_L(x, y). \tag{2}$$

On the other hand, as any alignment symbol can be the empty string, we have

$$n \geqslant |x_1 x_2 \ldots x_n| = |x| \quad \text{and} \quad n \geqslant |y_1 y_2 \ldots y_n| = |y|,$$

and thus:

$$n \geqslant \max\bigl\{|x|, |y|\bigr\}. \tag{3}$$

Combining Eqs. (2) and (3), we obtain the inequality stated in our lemma. □

The inequality stated in Lemma 4 involves only two strings. In order to generalize it to many strings (Lemma 6), we need the following lemma. For any two finite sets A, B, we have $\#(A \cup B) = \#A + \#B - \#(A \cup B)$. Lemma 5 states an analogous result for the length of the longest common subsequence. Indeed, the lcs of the union of $\{\mu\} \cup X$ and $\{\mu\} \cap Y$ is larger than or equal to the lcs of each minus the lcs of their intersection, which contains $\{\mu\}$.

Lemma 5. *For all $\mu \in \Sigma^\star$ and for all $X, Y \subseteq \Sigma^\star$, we have*

$$\operatorname{lcs}\bigl(\{\mu\} \cup X \cup Y\bigr) \geqslant \operatorname{lcs}\bigl(\{\mu\} \cup X\bigr) + \operatorname{lcs}\bigl(\{\mu\} \cup Y\bigr) - |\mu|. \tag{4}$$

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

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 subsequence of length $\#(I \cap J)$. It is also a common subsequence of all strings in $\{\mu\} \cup X \cup Y$. From which we deduce

$$\operatorname{lcs}\bigl(\{\mu\} \cup X \cup Y\bigr) \geqslant \#(I \cap J). \tag{5}$$

On the other hand, since I and J are subsets of $[1, |\mu|]$, we have $\#(I \cup J) \leqslant |\mu|$ and therefore

$$\#(I \cap J) = p + q - \#(I \cup J) \geqslant p + q - |\mu|. \tag{6}$$

Combining (5) and (6) gives (4) and concludes the proof. □

Lemma 6 generalizes Lemma 4 to the case of a language.

Lemma 6. *For all $\mu \in \Sigma^\star$ and for all finite languages X over Σ, we have*:

$$\sum_{x \in X} d_L(\mu, x) + (\#X - 1)|\mu| \geqslant |X| - \operatorname{lcs}\big(\{\mu\} \cup X\big)$$

where $|X|$ denotes $\sum_{x \in X} |x|$.

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 4.

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

$$d_L(\mu, x_0) \geqslant |x_0| - \operatorname{lcs}\big(\{\mu, x_0\}\big), \tag{7}$$

$$\sum_{x' \in X'} d_L(\mu, x') + (\#X' - 1)|\mu| \geqslant |X'| - \operatorname{lcs}\big(\{\mu\} \cup X'\big), \tag{8}$$

$$\operatorname{lcs}\big(\{\mu\} \cup X\big) \geqslant \operatorname{lcs}\big(\{\mu\} \cup X'\big) + \operatorname{lcs}\big(\{\mu, x_0\}\big) - |\mu|. \tag{9}$$

Inequalities (7) and (8) result respectively from Lemma 4 and from the induction hypothesis. Lemma 5 applied with $(X, Y) := (X', \{x_0\})$ yields (9). Adding (7), (8) and the trivial inequality $|\mu| \geqslant |\mu|$ we obtain

$$\begin{aligned}\sum_{x \in X} d_L(\mu, x) + (\#X')|\mu| &\geqslant |X'| + |x_0| - \operatorname{lcs}\big(\{\mu\} \cup X'\big) - \operatorname{lcs}\big(\{\mu, x_0\}\big) + |\mu| \\ &\geqslant |X'| + |x_0| - \operatorname{lcs}\big(\{\mu\} \cup X\big),\end{aligned}$$

where the last inequality deduces from (9). Since $\#X' = \#X - 1$ and $|X'| + |x_0| = |X|$, this concludes the proof. □

We can now prove the main theorem of this section. Our proof is inspired from the one of [2]. However, our reduction differs: instead of adding new symbols to the alphabet, we construct a language by concatenating a block of `0`'s to every word and adding new words that are comparatively small powers of `0`'s.

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

Proof. Suppose Σ is the binary alphabet $\{\mathtt{0}, \mathtt{1}\}$. The schema of the proof is the following: we reduce LCS (parameterized in $\#L$) to MEDIAN STRING (parameterized in $\#W$) in order to apply Theorem 1 and conclude.

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

$$n := \#L, \quad N := \max\left\{|L| + \frac{n(n-1)}{2} - k, n-1\right\},$$
$$K := |L| + (n-1)N - k - \frac{n(n-1)}{2}, \quad W := L\mathtt{0}^N \cup \left\{\mathtt{0}^i \colon i \in [1, n-1]\right\}.$$

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

$$\mathrm{lcs}(L) \geqslant k \quad \Leftrightarrow \quad \mathcal{S}(W) \leqslant K.$$

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

The idea of the proof is to choose μ as a potential median and computes the sum of the Levenshtein distances to all strings in W. First, we observe that the strings $\mathtt{0}^i$ are subsequences of μ and that μ is a subsequence of each string $x\mathtt{0}^N$. In such a case, Lemma 3 gives us a formula to compute the Levenshtein distance between μ and any string in W.

For all $i \in [1, n-1]$, we have $i \leqslant n-1 \leqslant N \leqslant |\mu|_{\mathtt{0}}$, so $\mathtt{0}^i$ is a subsequence of μ. Hence, by Lemma 3, we have

$$d_L(\mu, \mathtt{0}^i) = |\mu| - |\mathtt{0}^i| = k + N - i.$$

Moreover, for all $x \in L$, μ is a subsequence of $x\mathtt{0}^N$; again Lemma 3 applies and we obtain

$$d_L(\mu, x\mathtt{0}^N) = |x\mathtt{0}^N| - |\mu| = |x| - k.$$

Using these equalities, we compute the sum of the Levenshtein distances between μ and strings of W

$$\sum_{w\in W} d_L(\mu, w) = \sum_{x\in L} d_L(\mu, x\mathtt{0}^N) + \sum_{i=1}^{n-1} d_L(\mu, \mathtt{0}^i) = \sum_{x\in L}\big(|x| - k\big) + \sum_{i=1}^{n-1}(k + N - i)$$
$$= |L| - nk + (n-1)k + (n-1)N - \frac{n(n-1)}{2} = K$$

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

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

$$N - |\mu|_{\mathtt{0}} \leqslant |x'|_{\mathtt{0}} - |\mu|_{\mathtt{0}} \leqslant d_L(\mu, x')$$

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

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

and

$$\begin{aligned} n|\mu|_{\mathtt{0}} &\geqslant nN - K \\ &= nN - \left(|L| + (n-1)N - k - \frac{n(n-1)}{2}\right) \\ &= N - |L| + k + \frac{n(n-1)}{2} \\ &\geqslant \left(|L| + \frac{n(n-1)}{2} - k\right) - |L| + k + \frac{n(n-1)}{2} \\ &= n(n-1) \end{aligned}$$

which yields $\mu_{\mathtt{0}} \geqslant n-1$. This implies that for all $i \in [1, n-1]$, $\mathtt{0}^i$ is a subsequence of μ, and so $d_L(\mu, \mathtt{0}^i) = |\mu| - i$. Thus,

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

We can now write

$$\begin{aligned} \sum_{w \in W} d_L(\mu, w) &= \sum_{x' \in L\mathtt{0}^N} d_L(\mu, x') + (n-1)|\mu| - \frac{n(n-1)}{2} \\ &\geqslant |L\mathtt{0}^N| - \operatorname{lcs}\big(\{\mu\} \cup L\mathtt{0}^N\big) - \frac{n(n-1)}{2}, \end{aligned} \tag{10}$$

where the application of Lemma 6 with $X := L\mathtt{0}^N$ yields the last inequality.

On the other hand, we have:

$$|L\mathtt{0}^N| = \sum_{x \in L} |x\mathtt{0}^N| = \sum_{x \in L} \big(|x| + N\big) = |L| + nN \tag{11}$$

and by Lemma 1

$$\operatorname{lcs}\big(\{\mu\} \cup L\mathtt{0}^N\big) \leqslant \operatorname{lcs}(L\mathtt{0}^N) = \operatorname{lcs}(L) + N. \tag{12}$$

By hypothesis, $K \geqslant \mathcal{S}(W) \geqslant \sum_{w \in W} d_L(\mu, w)$; combining this with (10), (11) and (12) yields

$$K \geqslant \sum_{w \in W} d_L(\mu, w) \geqslant \big(|L| + nN\big) - \big(\operatorname{lcs}(L) + N\big) - \frac{n(n-1)}{2}$$

and thus

$$\operatorname{lcs}(L) \geqslant |L| + (n-1)N - \frac{n(n-1)}{2} - K = k.$$

This concludes the proof. □

3. Hardness of CENTER STRING under a weighted edit distance

Let d_E be a fixed integer valued edit distance that is a metric and satisfies the following property:

Property 1.

$$\forall a, b \in \Sigma \quad a \neq b \quad \Rightarrow \quad d_E(a, \varepsilon) < d_E(a, b) + d_E(b, \varepsilon).$$

Property 1 means that deleting a letter in a string costs *strictly* less than changing this letter into another letter and deleting that letter. It tightens slightly the triangle inequality and is a natural property for an edit distance. In this section, we prove that CENTER STRING under d_E is intractable. Our proof relies on a reduction of a weighted counterpart of LCS0 to CENTER STRING. After introducing the concept of weight, Section 3.1 proves that the weighted counterparts of LCS0 and LCS are intractable. Section 3.2 generalizes Lemma 3 to the weighted case and proves the main result.

3.1. *The* WEIGHTED COMMON SUBSEQUENCE *problem*

Let us first define a *weight*.

Definition 5. A *weight* over $\Sigma^\star$ is a mapping $\lambda : \Sigma^\star \to \mathbb{N}$ such that:

- for all $a \in \Sigma$, $\lambda(a) > 0$ and
- for all $x, y \in \Sigma^\star$, $\lambda(xy) = \lambda(x) + \lambda(y)$.

A weight λ is a morphism over the free monoid (for details on the free monoid see [15]). Hence, $\lambda(\varepsilon) = 0$ and λ is defined by its restriction to Σ:

$$\forall w \in \Sigma^\star \quad \lambda(w) = \lambda\big(w[1]\big) + \lambda\big(w[2]\big) + \cdots + \lambda\big(w\big[|w|\big]\big).$$

The weight over $\Sigma^\star$ that maps each element of Σ to 1, maps each string of $\Sigma^\star$ to its length.

We can now introduce the weighted counterparts to LCS and LCS0.

Definition 6. Let λ be a weight over $\Sigma^\star$.

The λ-WEIGHTED COMMON SUBSEQUENCE problem (denoted by λ-WCS) is: given a non-empty, finite language L over Σ and $k \in \mathbb{N}$, does there exist $s \in \mathrm{CSub}(L)$ with $\lambda(s) = k$?

The λ-WCS0 problem is the restriction of λ-WCS to the instances (L, k) such that for all $x \in L$, $\lambda(x) = 2k$.

We now show the intractability of λ-WCS0 for any fixed weight λ over $\Sigma^\star$ (Theorem 5). To prove this theorem requires Lemma 8. Lemma 8 considers a class of morphisms over the free monoid that "amplify" the letters of a string, i.e., that replace each letter a by a power of a. Let L be a language and f such a morphism. The result stated means that for each t that is a common subsequence to the image of L, one can find a common subsequence s

of L such that t is a subsequence of $f(s)$. This holds since it is possible to amplify some letters in t to obtain a common subsequence of $f(L)$ that has a preimage by f.

Although Lemma 8 is used in Theorem 5 in the finite case, it is also valid in the case of infinite languages, which is proved thanks to the following property of compactness.

Lemma 7 (Compactness). *Let X be a non-empty language over Σ, i.e., $X \subseteq \Sigma^\star$ and $X \neq \emptyset$. It exists Y, a finite and non-empty subset of X, such that* $\mathrm{CSub}(X) = \mathrm{CSub}(Y)$.

Proof. Choose a string u in X. The set of the subsequences of u, $\mathrm{Sub}(u)$, is finite and so is its subset S defined by $S := \mathrm{Sub}(u) \setminus \mathrm{CSub}(X)$. By definition of S, for each $s \in S$ one can find a string $x_s \in X$ such that s is not a subsequence of x_s. Setting $Y := \{u\} \cup \{x_s\colon\ s \in S\}$, we have that Y is a non-empty and finite subset of X. It remains to prove that $\mathrm{CSub}(X) = \mathrm{CSub}(Y)$. First, $Y \subseteq X$ implies $\mathrm{CSub}(X) \subseteq \mathrm{CSub}(Y)$. Now, let $s \in \mathrm{CSub}(Y)$ and assume that $s \notin \mathrm{CSub}(X)$. As $u \in Y$, s is a subsequence of u. By hypothesis, s belongs to S and is thus, not a subsequence of x_s, which contradicts $s \in \mathrm{CSub}(Y)$. Thus, we have $s \in \mathrm{CSub}(X)$ and $\mathrm{CSub}(Y) \subseteq \mathrm{CSub}(X)$. □

Lemma 8. *Let $f : \Sigma^\star \to \Sigma^\star$ be a mapping satisfying $f(a) \in \{a\}^+$ for each $a \in \Sigma$ and $f(xy) = f(x)f(y)$ for any x, $y \in \Sigma^\star$. Then, for any non-empty language L over Σ and for any $t \in \mathrm{CSub}(f(L))$, it exists $s \in \mathrm{CSub}(L)$ such that t is a subsequence of $f(s)$.*

Proof. If $\varepsilon \in L$ then $\varepsilon \in f(L)$ and $\mathrm{CSub}(f(L)) = \{\varepsilon\}$ and $t = \varepsilon$. In this case, setting $s := \varepsilon$ we get that $s \in \mathrm{CSub}(L)$ and t is a subsequence of $f(s)$. Now, let us consider the case where all strings in L are non-empty. Moreover, by Lemma 7, we can assume that L is finite and proceed by induction over $|L|$. As $L := \{\varepsilon\}$ is the only non-empty language satisfying $|L| = 0$ and this case is now excluded, we can further assume that $|L| > 0$. Two alternatives arise:

1. *All strings in L end by the same letter, say $a \in \Sigma$.* It exists $L' \subseteq \Sigma^\star$ such that $L = L'a$. It follows that $|L'| < |L|$ and $f(L) = f(L')f(a)$. Let α be the longest suffix of t that is a subsequence of $f(a)$ and t' be a string in $\Sigma^\star$ such that $t = t'\alpha$. By construction $t' \in \mathrm{CSub}(f(L'))$; so the induction hypothesis applies: it exists $s' \in \mathrm{CSub}(L')$ satisfying $t' \in \mathrm{Sub}(f(s'))$. Setting $s := s'a$, we obtain that $s \in \mathrm{CSub}(L)$ and $t \in \mathrm{Sub}(f(s))$, what we wanted to show.
2. *At least two strings end with distinct letters.* I.e., it exists a, $b \in \Sigma$ and u, $v \in \Sigma^\star$ such that $a \neq b$ and $\{ua, vb\} \subseteq L$. So, either a or b is not a suffix of t. As a and b play symmetrical roles, assume a is not a suffix of t. Since t is a subsequence of $f(ua) = f(u)f(a)$, it is also a subsequence of $f(u)$, as $f(a)$ contains only a's. Setting $L' := (L \setminus \{ua\}) \cup \{u\}$, we get $t \in \mathrm{CSub}(f(L'))$ and also $|L'| < |L|$ (more precisely, $|L'| = |L| - 1$ if $u \notin L$ and $|L'| = |L| - |ua|$ otherwise). Thus, the induction hypothesis applies: it exists $s \in \mathrm{CSub}(L')$ such that $t \in \mathrm{Sub}(f(s))$. Since by construction of L', we have $\mathrm{CSub}(L') \subseteq \mathrm{CSub}(L)$, we get $s \in \mathrm{CSub}(L)$, what we needed to show. □

Theorem 5. *Let λ be a weight over $\Sigma^\star$. Then the* λ-WCS0 *problem is* NP*-hard even if Σ is binary. Moreover,* λ-WCS0*, parameterized in $\#L$, is* W[1]*-hard.*

Proof. By Theorem 2, it is sufficient to reduce LCS0 (parameterized in $\#L$) to λ-WCS0 (parameterized in $\#L$). Suppose that Σ is the binary alphabet $\{0,1\}$. Let (L,k) be an instance of LCS0, L being a non-empty finite language over Σ and k a positive integer. We construct $(\tilde{L},\tilde{k})$ such that it is an instance of λ-WCS0. Let

$$(\tilde{L},\tilde{k}) := \big(f(L), \lambda(0)\lambda(1)k\big),$$

where the mapping $f:\{0,1\}^\star \to \{0,1\}^\star$ is given by: $f(0) = 0^{\lambda(1)}$, $f(1) = 1^{\lambda(0)}$ and for all $x,\ y \in \{0,1\}^\star$, $f(xy) = f(x)f(y)$. The mapping f replaces each 0 by a number of 0's equal to the weight of a 1, and symmetrically each 1 by a number of 1's equal to the weight of a 0. The idea behind this rewriting is to obtain strings $f(x)$ whose weight do not depends on their composition in 0's and 1's (which is the case in general), but solely on their length. Indeed, we have $\lambda(f(0)) = \lambda(f(1)) = \lambda(0)\lambda(1)$ and thus,

$$\forall x \in \{0,1\}^\star \quad \lambda\big(f(x)\big) = \lambda(0)\lambda(1)|x|. \tag{13}$$

Therefore, if $x \in L$, then $\lambda(f(x)) = \lambda(0)\lambda(1)(2k) = 2\tilde{k}$. Hence, $(\tilde{L},\tilde{k})$ is an instance of λ-WCS0.

The restriction of the morphism f to $\{0,1\}$ is injective and $f(\{0,1\}) = \{0^{\lambda(1)}, 1^{\lambda(0)}\}$ is a code (see [15, Chapter 6] for a definition) over $\{0,1\}$. Hence, f is injective [15] and so, $\#\tilde{L} = \#L$. This proves that the reduction of an instance (L,k) of LCS0 into the instance $(\tilde{L},\tilde{k})$ of λ-WCS0 is parameter preserving. Since it is polynomial, it remains to prove that $\mathrm{lcs}(L) \geqslant k$ if and only if there exists $\tilde{s} \in \mathrm{CSub}(\tilde{L})$ such that $\lambda(\tilde{s}) = \tilde{k}$.

Suppose $\mathrm{lcs}(L) \geqslant k$. Then, there exists $s \in \mathrm{CSub}(L)$ such that $|s| = k$. Let $\tilde{s} := f(s)$: $\tilde{s}$ belongs to $\mathrm{CSub}(\tilde{L})$ and $\lambda(\tilde{s}) = \tilde{k}$ (by Eq. (13)).

Conversely, suppose there exists $\tilde{s} \in \mathrm{CSub}(\tilde{L})$ such that $\lambda(\tilde{s}) = \tilde{k}$. By Lemma 8, there exists $s \in \mathrm{CSub}(L)$ such that $\tilde{s}$ is a subsequence of $f(s)$. Thus, we have:

$$\tilde{k} = \lambda(\tilde{s}) \leqslant \lambda\big(f(s)\big) = \lambda(0)\lambda(1)|s|$$

(the last equality coming from Eq. (13)) and from which we deduce $|s| \geqslant \frac{\tilde{k}}{\lambda(0)\lambda(1)} = k$. We can now write $\mathrm{lcs}(L) \geqslant |s| \geqslant k$, which completes the proof. □

Let λ be any fixed weight over $\Sigma^\star$. Since λ-WCS0 is a restriction of λ-WCS, an immediate corollary of Theorem 5 is that λ-WCS (resp. λ-WCS parameterized in $\#L$) is NP-hard (resp. W[1]-hard) even if Σ is binary.

3.2. *Hardness of* CENTER STRING *under any weighted edit distance*

Property 1 generalizes Lemma 3 to the case of a weighted edit distance.

Lemma 9. *For every $x,\ y \in \Sigma^\star$, we have*:

(i) $d_E(x,y) \geqslant d_E(x,\varepsilon) - d_E(y,\varepsilon)$,
(ii) *if y is a subsequence of x then* $d_E(x,y) = d_E(x,\varepsilon) - d_E(y,\varepsilon)$, *and*
(iii) *if d_E satisfies Property 1 and if* $d_E(x,y) = d_E(x,\varepsilon) - d_E(y,\varepsilon)$, *then y is a subsequence of x.*

Proof. Statement (i) is an immediate corollary of the triangle inequality: $d_E(x,\varepsilon) \leqslant d_E(x,y) + d_E(y,\varepsilon)$.

Let us prove statement (ii). Assume y is a subsequence of x. One can transform x into y by deleting $|x| - |y|$ letters of x. More precisely, for each $a \in \Sigma$ one needs to delete $|x|_a - |y|_a$ occurrences of a, which costs $(|x|_a - |y|_a)d_E(a,\varepsilon)$. The total editing cost for all letters is

$$\begin{aligned}\sum_{a\in\Sigma}\big(|x|_a - |y|_a\big)d_E(a,\varepsilon) &= \sum_{a\in\Sigma}|x|_a d_E(a,\varepsilon) - \sum_{a\in\Sigma}|y|_a d_E(a,\varepsilon)\\ &= d_E(x,\varepsilon) - d_E(y,\varepsilon).\end{aligned}$$

It follows that $d_E(x,y) \leqslant d_E(x,\varepsilon) - d_E(y,\varepsilon)$. As $d_E(x,\varepsilon) - d_E(y,\varepsilon) \leqslant d_E(x,y)$ is also true, we obtain the equality of statement (ii).

Let us now prove statement (iii). Assume that Property 1 is satisfied and that $d_E(x,y) = d_E(x,\varepsilon) - d_E(y,\varepsilon)$. Let $((x_1,y_1),(x_2,y_2),\ldots,(x_n,y_n))$ be an alignment between x and y of optimal cost $\sum_{i=1}^{n} d_E(x_i,y_i) = d_E(x,y)$.

To prove that y is a subsequence of x, it suffices to show that for any $i \in [1,n]$ we have $y_i \in \{\varepsilon, x_i\}$. As $x_1x_2\ldots x_n = x$, we get

$$\begin{aligned}\sum_{i=1}^{n} d_E(x_i,\varepsilon) = d_E(x,\varepsilon) &= d_E(x,\varepsilon) - d_E(y,\varepsilon) + d_E(y,\varepsilon) = d_E(x,y) + d_E(y,\varepsilon)\\ &= \sum_{i=1}^{n} d_E(x_i,y_i) + \sum_{i=1}^{n} d_E(y_i,\varepsilon) = \sum_{i=1}^{n}\big(d_E(x_i,y_i) + d_E(y_i,\varepsilon)\big),\end{aligned}$$

where the fourth equality follows from the alignment's optimality and from the fact that $y_1y_2\ldots y_n = y$. As for any position $i \in [1,n]$, the triangle inequality

$$d_E(x_i,\varepsilon) \leqslant d_E(x_i,y_i) + d_E(y_i,\varepsilon)$$

is satisfied, we obtain:

$$d_E(x_i,\varepsilon) = d_E(x_i,y_i) + d_E(y_i,\varepsilon).$$

By Property 1, this is true only if $y_i \notin \Sigma$ or if $y_i = x_i$. Thus, we have $y_i \in \{\varepsilon, x_i\}$, which proves the last statement of this lemma. □

The following lemma introduces a special weight: the morphism that maps a string to its distance to ε.

Lemma 10. *The mapping*

$$\Sigma^\star \to \mathbb{N}$$
$$w \to d_E(w,\varepsilon)$$

is a weight over $\Sigma^\star$.

Proof. To transform w into ε, we have to delete each letter in w and thus:

$$d_E(w,\varepsilon) = d_E\big(w[1],\varepsilon\big) + d_E\big(w[2],\varepsilon\big) + \cdots + d_E\big(w\big[|w|\big],\varepsilon\big). \qquad \square$$

The proof of the intractability of CENTER STRING is based on the reduction of WCS0 weighted by the above mentioned weight to CENTER STRING. We choose this weight for WCS0 because it is related to the distance d_E used in CENTER STRING.

Theorem 6. *Suppose d_E is an integer valued edit distance that is a metric satisfying Property 1. Then, the* CENTER STRING *problem under d_E is* NP*-complete even if Σ is binary. Moreover,* CENTER STRING *under d_E parameterized in $\#W$ is* W[1]*-hard.*

Proof. Suppose that Σ is the binary alphabet $\{\mathtt{0},\mathtt{1}\}$. Let $\lambda := d_E(.,\varepsilon)$ the weight over $\Sigma^\star$ as defined in Lemma 10. We reduce λ-WCS0 to CENTER STRING: we transform an instance (L,k) of λ-WCS0 in the instance $(W,K) := (L \cup \{\varepsilon\}, k)$ of CENTER STRING. The transformation is clearly polynomial and parameter preserving ($\#W \in \{\#L, \#L+1\}$). Hence, it remains to check that $\mathcal{R}(W)$ is at most K if and only if there exists $s \in \mathrm{CSub}(L)$ such that $\lambda(s) = k$.

(*Only if part*) First, assume that there exists $s \in \mathrm{CSub}(L)$ such that $\lambda(s) = k$. We show that $\mathcal{R}(W) \leqslant K$. Statement (ii) of Lemma 9 implies that for all $x \in L$: $d_E(x,s) = d_E(x,\varepsilon) - d_E(s,\varepsilon) = \lambda(x) - \lambda(s) = 2k - k = k$. As $d_E(\varepsilon,s) = \lambda(s) = k = K$, for any $x \in W$ it follows that $d_E(x,s) = K$ and thus, $\mathcal{R}(W) \leqslant k = K$.

(*If part*) Now, assume that $\mathcal{R}(W) \leqslant K$. We show that it exists an s in $\mathrm{CSub}(L)$ such that $\lambda(s) = k$.

By hypothesis, it exists $s \in \{\mathtt{0},\mathtt{1}\}^\star$ such that for any $w \in W$ we have $d_E(w,s) \leqslant K = k$. As ε belongs to W, we know by hypothesis that $d_E(\varepsilon,s) \leqslant k$. By definition of the radius, for any $x \in L$:

$$\begin{aligned}
k &\geqslant d_E(x,s) \\
&\geqslant d_E(x,\varepsilon) - d_E(s,\varepsilon) \quad \text{by statement (i) of Lemma 9} \\
&= 2k - d_E(s,\varepsilon) \quad \text{because } x \in L \\
&\geqslant 2k - k \quad \text{since } d_E(\varepsilon,s) \leqslant k \\
&= k.
\end{aligned}$$

It follows that the previous inequalities are in fact equalities. Thus, by statement (iii) of Lemma 9, $d_E(x,s) = d_E(x,\varepsilon) - d_E(s,\varepsilon)$ implies that s is a subsequence of x. Moreover, $2k - d_E(s,\varepsilon) = k$ and therefore $\lambda(s) = d_E(s,\varepsilon) = k$, which completes the proof. $\square$

The previous theorem means that, unless P = NP (resp. FPT = W[1]) it does not exists a weighted edit distance satisfying natural properties under which CENTER STRING is solvable in polynomial time (resp. is FPT in the number of input strings).

Note that if one replaces d_E by d_L the proof of Theorem 6 is a valid proof for the unweighted case (Theorem 3).

4. Conclusion

In Section 2.1 (see also [21]), we have shown that CENTER STRING under Levenshtein distance is NP-complete and W[1]-hard with respect to the number of input strings, even for binary alphabet. In Section 3, we generalize these results to any weighted edit distance that satisfies a natural condition. This condition is fulfilled in many applications of computational biology for instance. It remains open to find any particular weighted edit distance (of course, one that does not satisfy our condition) for which CENTER STRING would be polynomial, but this seems improbable.

Concerning MEDIAN STRING, we have shown (Section 2.2) that under the Levenshtein distance it is also NP-complete and W[1]-hard with respect to the number of input strings, even for binary alphabets. The complexity under a particular weighted edit distance remains open and seems non-trivial, since if $\Sigma = \{\mathtt{0}, \mathtt{1}\}$ and if the scores of insertions/deletions of $\mathtt{0}$ and of $\mathtt{1}$ are not equal then our reduction does not seem to hold.

Although CENTER STRING and MEDIAN STRING are NP-complete, there exist approximation algorithms with bounded errors [8] and heuristic algorithms that are used in practice [10,19]. For example, given a finite language W over Σ, a *set center* (resp. *set median*) of W can be found in polynomial time and is an approximate center (resp. median) of W with performance ratio 2 (resp. $2 - \frac{2}{\#W}$). Note that we call set center (resp. set median) of W any string that minimizes the maximum (resp. the sum) 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?

Acknowledgements

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 à la dynamique symbolique”.

Appendix A. A polynomial time algorithm for CENTER STRING with a fixed number of strings

In this section, d_E denotes a weighted metric edit distance over $\Sigma^\star$ that takes natural values. We set

$$E := \max_{a \in \Sigma} d_E(\varepsilon, a) = \max_{a \in \Sigma} d_E(a, \varepsilon).$$

E is the insertion and deletion cost of the heaviest symbols.

A.1. Additional notations

For any string w and any $k \in [0, |w|]$, $w^{(k)}$ denotes the *prefix* of w of length k, that is $w[1]w[2]\dots w[k]$ (note that $w^{(0)} = \varepsilon$). We denote by $\mathbb{Z}$ the set of all integers.

For any $n \in \mathbb{N}$ and any $P \in \mathbb{Z}^n$, we call the *support* of P, the set denoted by $\mathrm{supp}(P)$, of $i \in [1, n]$ such that $P[i] \neq 0$. For all $I \subseteq [1, n]$, we call *characteristic function* of I in $[1, n]$, the element $\mathbf{1}_I$ of $\{0, 1\}^n$ given by:

$$\forall i \in [1, n] \quad \mathbf{1}_I[i] = \begin{cases} 1 & \text{if } i \in I, \\ 0 & \text{if } i \notin I. \end{cases}$$

We denote $\mathbf{1}_{\{j\}}$ by $\mathbf{1}_j$ for any $j \in [1, n]$.

Theorem A.1 states the recurrence for the computation of d_E between two strings [28].

Theorem A.1. *Let d_E be any weighted edit distance over $\Sigma^\star$. For any $x, y \in \Sigma^\star$ and any $a, b \in \Sigma$, we have*:

$$\begin{aligned} d_E(xa, \varepsilon) &= d_E(x, \varepsilon) + d_E(a, \varepsilon), \\ d_E(\varepsilon, yb) &= d_E(\varepsilon, y) + d_E(\varepsilon, b), \\ d_E(xa, yb) &= \min \begin{cases} d_E(xa, y) + d_E(\varepsilon, b), \\ d_E(x, yb) + d_E(a, \varepsilon), \\ d_E(x, y) + d_E(a, b). \end{cases} \end{aligned}$$

A.2. A bound for the radius

The following property allows us to bound the radius of a language.

Proposition A.1. *Let W be a non-empty finite language over Σ and M the length of the longest strings in W. Then the radius of W is at most equal to EM.*

Proof. First, we bound the insertion cost of a string w by the cost of inserting a string that is a power of the heaviest symbol as long as w. For any $w \in \Sigma^\star$, we have:

$$d_E(\varepsilon, w) = \sum_{i=1}^{|w|} d_E\big(\varepsilon, w[i]\big) \leqslant \sum_{i=1}^{|w|} E = E|w|$$

and thus

$$\mathcal{R}(W) \leqslant \max_{w \in W} d_E(\varepsilon, w) \leqslant \max_{w \in W} E|w| = EM. \qquad \square$$

The bound given above is tight as shown by the following example.

Example A.1. Suppose that for all $x, y \in \Sigma^\star$, $d_E(x, y) = |x| + |y| - 2\,\mathrm{lcs}(x, y)$, i.e., that substitutions cost at least 2 and insertions/deletions cost 1. In other words substitution are unnecessary, since one can always replace a substitution by a deletion followed by an insertion. In this case, we have $E = 1$. Let $\Sigma := \{\mathtt{0}, \mathtt{1}\}$, $M \in \mathbb{N}$ and $W := \{\mathtt{0}^M, \mathtt{1}^M\}$: ε and $\mathtt{0}^M\mathtt{1}^M$ are centers of W and W admits M as radius.

A.3. Main algorithm

Let $n \in \mathbb{N}^*$ and $W := \{w_1, w_2, \ldots, w_n\}$ be a language with n strings over Σ. We want to compute the radius of W. We proceed by *dynamic programming*. Let $M := \max_{w \in W} |w|$ be the length of the longest strings in W and let

$$\mathcal{P} := \big[0, |w_1|\big] \times \big[0, |w_2|\big] \times \cdots \times \big[0, |w_n|\big].$$

An element of $\mathcal{P}$ is a combination of lengths, one for the prefix of each w_i. We denote by $\Delta : \mathcal{P} \times \mathbb{Z}^n \to \{\top, \bot\}$ the boolean valued mapping that for any $(P, D) \in \mathcal{P} \times \mathbb{Z}^n$ is defined by:

$$\Delta[P, D] \quad \Leftrightarrow \quad \exists s \in \Sigma^\star,\ \forall i \in [1, n], d_E\big(s, w_i^{(P[i])}\big) \leqslant D[i].$$

D is a vector of maximum values for the edit distances. $\Delta[P, D]$ is true iff it exists a string s such that for each string $w_i \in W$, the edit distance between s and the prefix of w specified by $P[i]$ is at most $D[i]$. By Proposition A.1, it suffices to inspect the entries $\Delta[(|w_1|, |w_2|, \ldots, |w_n|), D]$ for $D \in [0, EM]^n$ to compute the radius of W. Our algorithm computes by dynamic programming the restriction of Δ to $\mathcal{P} \times [0, EM]^n$.

Before giving the recurrence relation in Theorem A.2 we start by two lemmas. Lemma A.1 follows from the fact that the metric d_E does not take negative values. Lemma A.2 rewrites $\Delta[P, D]$ when $P = (0, 0, \ldots, 0)$; it initializes the recurrence.

Lemma A.1. *For any $P \in \mathcal{P}$ and any $D \in \mathbb{Z}^n \setminus \mathbb{N}^n$, $\Delta[P, D]$ is false.*

Lemma A.2. *For any $D \in \mathbb{N}^n$, $\Delta[(0, 0, \ldots, 0), D]$ is true.*

Proof. Let $s := \varepsilon$. If $P = (0, 0, \ldots, 0)$ then for all $i \in [1, n]$, $w_i^{(P[i])} = \varepsilon$, and thus,

$$d_E\big(s, w_i^{(P[i])}\big) = 0. \qquad \square$$

The following Theorem A.2 states the main recurrence that enables the computation of $\Delta[P, D]$'s by dynamic programming.

Theorem A.2. *Let $P \in \mathcal{P}$ with $P \neq (0, 0, \ldots, 0)$ and $D \in \mathbb{Z}^n$. We have $\Delta[P, D]$ if and only if it exists $j \in \operatorname{supp}(P)$ such that*

$$\Delta\big[P - \mathbf{1}_j, D - d_E\big(w_j\big[P[j]\big], \varepsilon\big)\mathbf{1}_j\big]$$

OR it exists $a \in \Sigma$ and $J \subseteq \operatorname{supp}(P)$ such that

$$\Delta\Big[P - \mathbf{1}_J,\ D - d_E(a, \varepsilon)\mathbf{1}_{[1,n]\setminus J} - \sum_{j \in J} d_E\big(a, w_j\big[P[j]\big]\big)\mathbf{1}_j\Big]. \tag{A.1}$$

The first term of the logical OR is the case of an insertion at the end of some $w_j^{(P[j])}$. The second term (Eq. (A.1)) is the case of a deletion or a substitution (or a match) at the end of all the $w_j^{(P[j])}$'s.

Proof. For each $(s, P, D) \in \Sigma^\star \times \mathcal{P} \times \mathbb{Z}^n$ we denote by $\Gamma[s, P, D]$ the boolean value of the assertion:

$$\forall i \in [1, n] \quad d_E\big(s, w_i^{(P[i])}\big) \leqslant D[i].$$

This notation allows us to shorten the proof since:

$$\Delta[P, D] \quad \Leftrightarrow \quad \exists s \in \Sigma^\star,\ \Gamma[s, P, D].$$

Let $P \in \mathcal{P}$ with $P \neq (0, 0, \ldots, 0)$ and $D \in \mathbb{Z}^n$.

We first show the "if" part.

• Assume that the first term of the logical OR is true. I.e., suppose it exists $j \in \mathrm{supp}(P)$ and $s \in \Sigma^\star$ such that $\Gamma[s, P - \mathbf{1}_j, D - d_E(w_j[P[j]], \varepsilon)\mathbf{1}_j]$ is true. For any $i \in [1, n] \setminus \{j\}$, as $P[i] = (P - \mathbf{1}_j)[i]$, we get

$$\begin{aligned} d_E\big(s, w_i^{(P[i])}\big) &= d_E\big(s, w_i^{((P-\mathbf{1}_j)[i])}\big) \\ &\leqslant \big(D - d_E\big(w_j\big[P[j]\big], \varepsilon\big)\mathbf{1}_j\big)[i] \\ &= D[i]. \end{aligned}$$

For w_j, if we rewrite its prefix of length $P[j]$ in its prefix of length $P[j]-1$ concatenated with its $P[j]$th letter, we obtain

$$\begin{aligned} d_E\big(s, w_j^{(P[j])}\big) &= d_E\big(s, \big(w_j^{((P-\mathbf{1}_j)[j])}\big) w_j\big[P[j]\big]\big) \\ &\leqslant d_E\big(s, w_j^{((P-\mathbf{1}_j)[j])}\big) + d_E\big(\varepsilon, w_j\big[P[j]\big]\big) \quad \text{by Theorem A.1} \\ &\leqslant \big(D - d_E\big(w_j\big[P[j]\big], \varepsilon\big)\mathbf{1}_j\big)[j] + d_E\big(\varepsilon, w_j\big[P[j]\big]\big) \quad \text{by hyp.} \\ &= D[j]. \end{aligned}$$

It follows that $\Gamma[s, P, D]$ holds and that $\Delta[P, D]$ is true.

• Now, assume that the second term of the logical OR is true. I.e., it exists $a \in \Sigma$ and $J \subseteq \mathrm{supp}(P)$ and $s' \in \Sigma^\star$ such that

$$\Gamma\left[s', P - \mathbf{1}_J, D - d_E(a, \varepsilon)\mathbf{1}_{[1,n]\setminus J} - \sum_{j \in J} d_E\big(a, w_j\big[P[j]\big]\big)\mathbf{1}_j\right]$$

is true. On one hand, for any $i \in [1, n] \setminus J$, by decomposing the prefixes of the w_i's we get

$$\begin{aligned} d_E\big(s'a, w_i^{(P[i])}\big) &= d_E\big(s'a, w_i^{((P-\mathbf{1}_J)[i])}\big) \quad \text{since } i \notin J \\ &\leqslant d_E\big(s', w_i^{((P-\mathbf{1}_J)[i])}\big) + d_E(a, \varepsilon) \quad \text{by Theorem A.1} \\ &\leqslant \left(D - d_E(a, \varepsilon)\mathbf{1}_{[1,n]\setminus J} - \sum_{j \in J} d_E\big(a, w_j\big[P[j]\big]\big)\mathbf{1}_j\right)[i] \\ &\quad + d_E(a, \varepsilon) \quad \text{by hyp.} \\ &= D[i]. \end{aligned}$$

On the other hand, for any $i \in J$, we have

$$\begin{aligned} d_E\big(s'a, w_i^{(P[i])}\big) &= d_E\big(s'a, \big(w_i^{((P-\mathbf{1}_i)[i])}\big) w_i\big[P[i]\big]\big) \\ &\leqslant d_E\big(s', w_i^{((P-\mathbf{1}_i)[i])}\big) + d_E\big(a, w_i\big[P[i]\big]\big) \quad \text{by Theorem A.1} \\ &\leqslant \left(D - d_E(a,\varepsilon)\mathbf{1}_{[1,n]\setminus J} - \sum_{j\in J} d_E\big(a, w_j\big[P[j]\big]\big)\mathbf{1}_j\right)[i] \\ &\quad + d_E\big(a, w_i\big[P[i]\big]\big) \quad \text{by hyp.} \\ &= D[i]. \end{aligned}$$

It follows that $\Gamma[s'a, P, D]$ holds, which implies that $\Delta[P, D]$ is true. This completes the first part of the proof.

Let us now show the "only if" part.

Suppose it exists $s \in \Sigma^\star$ satisfying $\Gamma[s, P, D]$. Two cases arise.

- First, assume there exists $j \in \mathrm{supp}(P)$ such that:

$$d_E\big(s, w_j^{(P[j])}\big) = d_E\big(s, w_j^{(P[j]-1)}\big) + d_E\big(\varepsilon, w_j\big[P[j]\big]\big). \tag{A.2}$$

In this case, we obtain that $\Gamma[s, P - \mathbf{1}_j, D - d_E(\varepsilon, w_j[P[J]])\mathbf{1}_j]$ is true and we are done. Note that Eq. (A.2) means it exists an alignment between s and $w_j^{(P[j])}$ with minimum cost $d_E(s, w_j^{(P[j])})$ ending by $(\varepsilon, w_j[P[j]])$ (i.e., by the deletion of the last letter of $w_j^{(P[j])}$).

- Conversely, assume that for each $j \in \mathrm{supp}(P)$, Eq. (A.2) is false.

This assumption requires $s \neq \varepsilon$ and thus, it exists $a \in \Sigma$ and $s' \in \Sigma^\star$ such that $s = s'a$. Furthermore, for each $j \in \mathrm{supp}(P)$, an alignment between s and $w_j^{(P[j])}$ with minimum cost $d_E(s, w_j^{(P[j])})$ ends either by (a, ε) (deletion of the last letter of s) or by $(a, w_j[P[j]])$ (match or substitution between the last letters of both words). Let us denote by J the subset of $j \in \mathrm{supp}(P)$ that are in the last case, i.e., such that there exists an alignment between s and $w_j^{(P[j])}$ with cost $d_E(s, w_j^{(P[j])})$ that ends by $(a, w_j[P[j]])$, then:

$$\forall j \in J \quad d_E\big(s, w_j^{(P[j])}\big) = d_E\big(s', w_j^{(P[j]-1)}\big) + d_E\big(a, w_j\big[P[j]\big]\big).$$

On the other hand, for any $j \in [1, n] \setminus J$, an alignment between s and $w_j^{(P[j])}$ with cost $d_E(s, w_j^{(P[j])})$ ends by (a, ε) (note that if $j \notin \mathrm{supp}(P)$ then *any* alignment between s and $w_j^{(P[j])} = \varepsilon$ ends by (a, ε)). Hence we obtain

$$\forall j \in [1, n] \setminus J \quad d_E\big(s, w_j^{(P[j])}\big) = d_E\big(s', w_j^{(P[j])}\big) + d_E(a, \varepsilon).$$

And so

$$\Gamma\left[s', P - \mathbf{1}_J,\ D - d_E(a,\varepsilon)\mathbf{1}_{[1,n]\setminus J} - \sum_{j\in J} d_E\big(a, w_j\big[P[j]\big]\big)\mathbf{1}_j\right]$$

is true, what we wanted. This completes both the proof of the "only if" part and the proof of the theorem. □

Theorem A.3. *Assume $n \in \mathbb{N}^*$ is fixed and consider the* CENTER STRING *problem under d_E for finite languages $W \subseteq \Sigma^\star$ of cardinality n. It exists an algorithm that computes the radius and a center of W under d_E in polynomial time $O(\#\Sigma \times M^{2n})$, where M is length of a longest string in W.*

Proof. We store in memory the restriction of Δ to $\mathcal{P} \times [0, EM]^n$ in a $2n$-dimensional table of boolean values. The algorithm proceeds in two steps.

1. *Initialization*. According to Lemma A.2, we set, for each $D \in [0, EM]^n$, the bit coding for $\Delta[(0, 0, \ldots, 0), D]$ to $\top$. This step takes $O(\#\mathcal{P} \times (1 + EM)^n)$ time, which is roughly bounded by $O(\#\Sigma \times M^{2n})$.
2. *Recurrence*. We enumerate all $P \in \mathcal{P}$ in lexicographical order, and for each P, we compute all the entries $\Delta[P, D]$ for $D \in [0, EM]^n$; for this $\#\mathcal{P} \times (1 + EM)^n$ boolean values are computed. The recurrence relation given in Theorem A.2 allows us to compute each entry in $O(\#\Sigma)$ time (note that Eq. (A.1) has to be evaluated for all $a \in \Sigma$ and negative values in D are handled by Lemma A.1). So, the whole step takes $O(\#\Sigma \times \#\mathcal{P} \times (1 + EM)^n)$ time, which is bounded by $O(\#\Sigma \times M^{2n})$.

Altogether to compute the radius of W, the algorithm requires $O(\#\Sigma \times M^{2n})$ time as stated. A center of W can then be obtained by backtracking in the matrix that stores Δ. □

References

[1] A. Ben-Dor, G. Lancia, J. Perone, R. Ravi, Banishing bias from consensus sequences, in: Proceedings of the 8th Annual Symposium on Combinatorial Pattern Matching (CPM'97), 1997, pp. 247–261.
[2] C. de la Higuera, F. Casacuberta, Topology of strings: median string is NP-complete, Theoret. Comput. Sci. 230 (2000) 39–48.
[3] X. Deng, G. Li, Z. Li, B. Ma, L. Wang, A PTAS for distinguishing (sub)string selection, in: Proceedings of the 29th International Colloquium on Automata, Languages and Programming (ICALP'02), 2002, pp. 740–751.
[4] R.G. Downey, M.R. Fellows, Parameterized Complexity, Springer, 1999.
[5] M.R. Fellows, J. Gramm, R. Niedermeier, On the parameterized intractability of CLOSEST SUBSTRING and related problems, in: Proceedings of the 19th Symposium on Theoretical Aspects of Computer Science (STACS'02), 2002, pp. 262–273.
[6] J. Gramm, R. Niedermeier, P. Rossmanith, Fixed-parameter algorithms for CLOSEST STRING and related problems, Algorithmica 37 (1) (2003) 25–42.
[7] D. Gusfield, Efficient methods for multiple sequence alignment with guaranteed error bounds, Bull. Math. Biol. 55 (1) (1993) 141–154.
[8] D. Gusfield, Algorithms on Strings, Trees, and Sequences. Computer Science and Computational Biology, Cambridge University Press, 1997.
[9] T. Jiang, E.L. Lawler, L. Wang, Approximation algorithms for tree alignment with a given phylogeny, Algorithmica 16 (3) (1996) 302–315.
[10] T. Kohonen, Median strings, Pattern Recognition Letters 3 (1985) 309–313.
[11] J.K. Lanctot, M. Li, B. Ma, S. Wang, L. Zhang, Distinguishing string selection problems, Inform. and Comput. 185 (1) (2003) 41–55.
[12] M. Li, B. Ma, L. Wang, Finding similar regions in many strings, in: Proceedings of the 31st Annual ACM Symposium on Theory of Computing (STOC'99), 1999, pp. 473–482.
[13] M. Li, B. Ma, L. Wang, Finding similar regions in many sequences, J. Comput. System Sci. 65 (2002) 73–96.

[14] M. Li, B. Ma, L. Wang, On the closest string and substring problems, J. ACM 49 (2) (2002) 157–171.
[15] M. Lothaire, Algebraic Combinatorics on Words, Cambridge University Press, 2002.
[16] B. Ma, A polynomial time approximation scheme for the closest substring problem, in: Proceedings of the 11th Annual Symposium on Combinatorial Pattern Matching (CPM'00), 2000, pp. 99–107.
[17] D. Maier, The complexity of some problems on subsequences and supersequences, J. ACM 25 (2) (1978) 322–336.
[18] L. Marsan, 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) (2000) 345–362.
[19] C.D. Martinez-Hinarejos, A. Juan, F. Casacuberta, Use of median string for classification, in: Proceedings of the 15th International Conference on Pattern Recognition, vol. 2, 2000, pp. 907–910.
[20] C.D. Martínez, A. Juan, F. Casacuberta, Improving classification using median string and NN rules, in: Proceedings of the 9th Spanish Symposium on Pattern Recognition and Image Analysis, vol. 2, 2001, pp. 391–395.
[21] F. Nicolas, É. Rivals, Complexities of the centre and median string problems, in: R. Baeza-Yates, E. Chavez, M. Crochemore (Eds.), Proceedings of the 14th Annual Symposium on Combinatorial Pattern Matching (CPM'03), in: Lecture Notes in Comput. Sci., vol. 2676, Springer-Verlag, 2003, pp. 315–327.
[22] P. Pevzner, Computational Molecular Biology: An Algorithmic Approach, MIT Press, 2000.
[23] K. Pietrzak, On the parameterized complexity of the fixed alphabet shortest common supersequence and longest common subsequence problems, J. Comput. System Sci. 67 (4) (2003) 757–771.
[24] R. Ravi, J. Kececioglu, Approximation algorithms for multiple sequence alignment under a fixed evolutionary tree, Discrete Appl. Math. 88 (1–3) (1998) 355–366.
[25] D. Sankoff, Minimal mutation trees of sequences, SIAM J. Appl. Math. 28 (1) (1975) 35–42.
[26] J.S. Sim, K. Park, The consensus string problem for a metric is NP-complete, J. Discrete Algorithms 1 (1) (2003) 111–117.
[27] D.J States, P. Agarwal, Compact encoding strategies for DNA sequence similarity search, in: Proceedings of the 4th International Conference on Intelligent Systems for Molecular Biology (ISMB'96), AAAI Press, 1996, pp. 211–217.
[28] R.A. Wagner, M.J. Fischer, The string-to-string correction problem, J. ACM 21 (1) (1974) 168–173.
[29] L. Wang, D. Gusfield, Improved approximation algorithms for tree alignment, J. Algorithms 25 (2) (1997) 255–273.