Closest Substring Problems for Regular Languages

Yo-Sub Han[1], Sang-Ki Ko[2], Timothy Ng[3], and Kai Salomaa[4]

[1] Department of Computer Science, Yonsei University, South Korea
emmous@yonsei.ac.kr
[2] AI Research Center, Korea Electronics Technology Institute, South Korea
sangkiko@keti.re.kr
[3] David R. Cheriton School of Computer Science, University of Waterloo, Canada
tim.ng@uwaterloo.ca
[4] School of Computing, Queen's University, Canada
ksalomaa@cs.queensu.ca

Abstract. It is well known that given a finite set of strings of equal length, the Consensus String problem—the problem of deciding whether or not there exists a consensus string whose distance is at most r from every string in the given set—is proven to be NP-complete. A similar problem called the Closest Substring problem asks whether there exists a string w of length ℓ such that each string in a given set L has a substring whose distance is at most r (called radius) from w. As the Closest Substring problem is a generalized version of the Consensus String problem, it is obvious that the problem is NP-hard for a finite set of strings. We show that the Closest Substring problem for regular languages represented by nondeterministic finite automata (NFAs) is PSPACE-complete. The main difference from the previous work is that we consider an infinite set of strings, which is recognized by an NFA as input instead of a finite set of strings. We also prove that the Closest Substring problem for acyclic NFAs lies in the second level of the polynomial-time hierarchy Σ_2^{P} and is both NP-hard and coNP-hard.

Keywords: closest substring problem, computational complexity, regular languages, edit distance

1 Introduction

It is a very important task to find the *consensus string* for detecting data commonalities from a set of strings in many practical applications such as computational biology [4], coding theory [9], data compression [10], and so forth.

There are several definitions of a consensus string for a set of strings. Frances and Litman defined the consensus string based on the concept of the *radius* [9]. The radius of a string w with respect to a set S of strings is the smallest number r such that the distance between w and any string in S is bounded by r. They consider the *Hamming distance* as a distance metric between two strings of equal length that counts the number of positions where they differ. It is known that

Language class	Complexity (Lower/upper bound)
A set of strings	NP-complete [9]
Sub-regular (acyclic (non)deterministic FAs)	(coNP,NP)-hard / Σ_2^{P}
Regular ((non)deterministic FAs)	PSPACE-complete
Context-free	PSPACE-hard / EXPTIME
Context-sensitive	Undecidable

Table 1. The complexity results for the Closest Substring problem when l and r are given in unary.

the Consensus String problem based on radius is NP-complete even when the strings are given over the binary alphabet [9].

Another important related problem is the Closest Substring problem [9]:

> *Instance:* a set of k strings $s_1, s_2, \ldots, s_k$ and two positive integers $r, l \in \mathbb{N}$.
> *Question:* does there exist a string s of length l such that, for each $i \in \{1, \ldots, k\}$, there exists a substring s'_i of length l in s_i such that the Hamming distance between s and s'_i is at most r?

Note that the Closest Substring problem is also NP-hard since the Consensus String problem is a special case of the Closest Substring problem. Fellows et al. [8] investigated the problem from the parameterized complexity point of view and proved that the Closest Substring problem is W[1]-hard with respect to k. Later, Marx [13] showed that the problem is W[1]-hard even if both r and k are parameters. Recently, Schmid [16] investigated a variant where the length differences of the input strings are bounded.

We study the Closest Substring problem for regular languages which is to find a closest substring from a given regular language represented by an FA. In this setting, we consider a set of infinite number of strings instead of a finite set of strings in the classical consensus string problems. Moreover, we consider an arbitrary fixed edit-distance where the costs of the edit operations are not necessarily unit cost. Formally, the following problem is considered:

> *Instance:* a regular language L represented by an FA and integers $r, l \in \mathbb{N}$.
> *Question:* does there exist a string w (called a *consensus substring*) of length l such that every string $w' \in L$ has a substring whose distance is at most r from w?

Recently, the same authors investigated a similar problem for multiple regular languages where we try to find a consensus string which is within a given radius r from each of regular languages [11]. We also notice that all complexity upper bounds we prove in the paper can be applied to the cases when we have NFAs as input and all lower bounds hold for deterministic FAs (DFAs). The main complexity results are presented in Table 1.

2 Preliminaries

We provide important definitions and notation, and formulate several variants of our problem.

Basic definitions. The cardinality of a finite set S is denoted $|S|$. The symbol Σ always stands for a finite alphabet. The set of all strings over Σ is Σ^*, the set of nonempty strings is Σ^+ and the set of strings of length at most m is $\Sigma^{\leq m}$. The empty string is ε. The complement of a language L over alphabet Σ is L^c $(= \Sigma^* - L)$. For $w \in \Sigma^*$, infix(w) is the set of infixes, or substrings, of w.

A nondeterministic finite automaton (NFA) is a tuple $A = (\Sigma, Q, \delta, Q_0, F)$ where Σ is the input alphabet, Q is the finite set of states, $\delta\colon Q \times \Sigma \to 2^Q$ is the multivalued transition function, $Q_0 \subseteq Q$ is the set of initial states and $F \subseteq Q$ is the set of final states. In the usual way δ is extended to a function $2^Q \times \Sigma^* \to 2^Q$ and the language accepted by A is $L(A) = \{w \in \Sigma^* \mid \delta(Q_0, w) \cap F \neq \emptyset\}$. The automaton A is a deterministic finite automaton (DFA) if $|Q_0| = 1$ and δ is a single valued function $Q \times \Sigma \to Q$. It is well known that the DFAs and NFAs recognize the class of *regular languages* [19].

Edit distance and some technical properties. Atomic edit operations on a string x consist of substituting an element of Σ by another element of Σ, deleting an element of Σ from x, or inserting an element of Σ into x. We recall that a function $d : \Sigma^* \times \Sigma^* \to \mathbb{N}_0$ is a *distance* if it satisfies, for all $x, y, z \in \Sigma^*$,

(i) $d(x, y) = 0$ if and only if $x = y$, (identity)
(ii) $d(x, y) = d(y, x)$, and (symmetry)
(iii) $d(x, z) \leq d(x, y) + d(y, z)$. (triangle inequality)

The Levenshtein distance $d_L(x, y)$ between strings x and y is the smallest number of atomic edit operations that transform x into y [7, 12, 14]. We can define the cost function for the edit distance by assigning unit cost to every edit operation. We also use the Hamming distance as a metric for measuring the difference between two strings. Note that the Hamming distance between two strings is the number of positions at which the symbols are different. In other words, the Hamming distance only allows substitutions and must be restricted to equal-length strings. We denote the Hamming distance between two strings x and y by $d_H(x, y)$.

The *neighbourhood* of radius r of a language L [14] under the distance d is defined as $E(L, d, r) = \{\ x \in \Sigma^* \mid (\exists y \in L)\ d(x, y) \leq r\ \}$. The distance d is *additive* if for any decomposition $w = w_1 w_2 \in \Sigma^*$, and radius $r \geq 0$, $[E(w, d, r) = \bigcup_{r_1 + r_2 = r} E(w_1, d, r_1) \cdot E(w_2, d, r_2)$.

Note that the Levenshtein distance d_L and the Hamming distance d_H are both additive [5]. Unless explicitly mentioned otherwise, we consider the problem with respect to an arbitrary, but fixed, additive edit distance $d_e : \Sigma^* \times \Sigma^* \to \mathbb{N}_0$. That is, the costs of the individual edit operations in $(\Sigma \times \{\varepsilon\}) \times (\Sigma \times \{\varepsilon\})$ are constants, but not necessarily unit cost.

We recall a result on the nondeterministic state complexity of neighbourhoods of regular languages. The result is originally due to Povarov [15] who states it for the Hamming distance neighbourhoods. However, exactly the same construction works for any additive integral distance metric, where the costs of elementary edit operations are non-negative integers [14]. We add to the statement of the result a time bound needed for the construction.

Proposition 1 ([14, 15]). *Let A be an NFA with n states and $r \in \mathbb{N}$. The neighbourhood of $L(A)$ of radius r with respect to the additive distance d can be recognized by an NFA B with $n \cdot (r+1)$ states. The NFA B can be constructed in time that depends polynomially on n and r.*

Problem definition. For $w, w' \in \Sigma^*$ the *minimum infix distance*, (*mininfix distance,* for short), from w to w' is defined as $d_{\text{minfix}}(w, w') = \min\{d_e(w, y) \mid y \in \text{infix}(w')\}$.

That is, $d_{\text{minfix}}(w, w')$ is the smallest edit distance between w and a substring of w'. In particular, $d_{\text{minfix}}(w, w')$ is always at most $|w|$. Note that the mininfix distance measure is not, in general, symmetric. For example, $d_{\text{minfix}}(b, ab) = 0$ and $d_{\text{minfix}}(ab, b) = 1$ when $a, b \in \Sigma$.

For $w \in \Sigma^*$ and $L \subseteq \Sigma^*$ we define the *inner distance between w and L* as

$$d_{\text{inner}}(w, L) = \begin{cases} \sup\{d_{\text{minfix}}(w, w') \mid w' \in L\} & \text{if } L \neq \emptyset, \\ \infty & \text{otherwise.} \end{cases}$$

The inner distance between w and L is the largest minimum infix distance between w and a string of L. From the definition it follows that $d_{\text{inner}}(w, L) \leq |w|$.

A consensus substring of a language L has a bounded mininfix distance to any other string of the language L.

Definition 2. *A string w is a radius r consensus substring of language L if $d_{\text{inner}}(w, L) \leq r$.*

In other words, w is a radius r consensus substring of L if and only if every $w' \in L$ can be decomposed to $w' = xyz$ where $d_e(w, y) \leq r$.

We consider the algorithmic problem of determining the existence of a consensus substring of a given length for a regular language. Hence, the Closest Substring problem for NFAs (respectively, DFAs) asks, for a given NFA (respectively, DFA) A, whether or not $L(A)$ has a length ℓ consensus substring of radius r. The complexity of the problem can be different depending on whether the length and the radius are part of the input instance, or they are considered as constants. While the standard assumption is that integers given as input are encoded in binary, we also study several problems where some integers are given in unary.

3 The closest substring problem for NFAs is PSPACE-complete

Before tackling the main problem, we observe several relationships among the following input values: the language $L \subseteq \Sigma^*$, the length $\ell \in \mathbb{N}$ of a consensus

string, and the radius $r \in \mathbb{N}$. We denote by $\min(L)$ the length of the shortest string in L. First we observe the following relationship.

Lemma 3. *Consider an edit distance d_e and let c_{mindel} be the smallest deletion cost of a symbol of the alphabet Σ. Let $L \subseteq \Sigma^*$ be a non-empty language and $r \in \mathbb{N}$. If L has a length ℓ consensus substring $w \in \Sigma^*$ of radius r, then $\ell \leq \min(L) + \frac{r}{c_{\text{mindel}}}$ holds.*

Proof. Suppose that x is a shortest string in L and w has length greater than $\min(L) + \frac{r}{c_{\text{mindel}}}$. Then, since the cost of deletions (and insertions) is at least c_{mindel}, no substring of x can be transformed into w by a sequence of edit operations with cost r. □

Lemma 3 says that the length ℓ of a consensus substring should be bounded by a value that depends on the shortest string in L and the radius r. We can also observe the following fact.

Lemma 4. *Consider an edit distance d_e over alphabet Σ and let c_{maxdel} be the largest cost of an individual deletion operation. Let $L \subseteq \Sigma^*$ be a non-empty language and $r \in \mathbb{N}$. Then, every string $w \in \Sigma^*$ of length $\ell \leq \frac{r}{c_{\text{maxdel}}}$ is a radius r consensus substring of L.*

Proof. Let $\ell \in \mathbb{N}$. If $\ell \leq \frac{r}{c_{\text{maxdel}}}$, then any string w of length ℓ can be aligned with the empty string at cost at most r simply by deleting all symbols, which means that the inner distance from w to L is at most r. □

Based on Lemma 3 and Lemma 4, we consider different possible relationships among $\min(L)$, ℓ and r:

- If $\ell > \min(L) + \frac{r}{c_{\text{mindel}}}$, then we know that there cannot be a string of length ℓ with inner distance at most r to L.
- If $\ell \leq \frac{r}{c_{\text{maxdel}}}$, then any string of length ℓ has inner distance at most r to any language.

Therefore, we can restrict consideration to the case where the required length ℓ for consensus string satisfies the following inequalities:

$$\frac{r}{c_{\text{maxdel}}} < \ell \leq \min(L) + \frac{r}{c_{\text{mindel}}}. \tag{1}$$

3.1 Upper bound

Now we are ready to solve the Closest Substring problem for regular languages. The following lemma first presents a PSPACE algorithm for the Closest Substring problem where the length ℓ of a consensus substring is given in unary.

Lemma 5. *The* Closest Substring *problem for NFAs can be solved in* PSPACE *when the length ℓ of consensus substring is given in unary.*

Proof. Recall that the input for the problem is the length $\ell \in \mathbb{N}$ as a unary string 0^ℓ, the radius $r \in \mathbb{N}$ in binary, and an NFA A.

First, we nondeterministically guess a string $w \in \Sigma^\ell$ and construct an NFA B for $\Sigma^* E(w, d_e, r)\Sigma^*$, where d_e is the edit distance used. Note that B has $O(lr)$ states by Proposition 1 and hence the size of B is polynomial in ℓ and r. Since ℓ is given in unary, the only concern is that r may not be polynomial in the size of input. If c_{maxdel} is the maximum cost of individual deletion operations of d_e, we know by (1) that it is sufficient to consider only cases where $r < c_{\text{maxdel}} \cdot \ell$ which are polynomial in the size of the input.

Since $L(A) \subseteq L(B)$ can be decided in PSPACE, we can decide whether or not the guessed string w is a radius r consensus substring of $L(A)$ in PSPACE. □

Lemma 5 states that we can solve the Closest Substring problem in PSPACE if the length ℓ of a closest substring is given in unary. On the other hand, if the radius is given in unary we also get a polynomial space algorithm by (1) since ℓ should be bounded from the above by $\min(L) + \frac{r}{c_{\text{mindel}}}$, where $\min(L)$ is bounded from the above by the number of states in NFA that accepts L.

Corollary 6. *The* Closest Substring *problem for NFAs can be solved in* PSPACE *when the radius r is given in unary.*

For the case when both the length and the radius are given in binary, the following result gives a PSPACE upper bound for edit distances where the cost of all deletion operations is the same. Note that, due to symmetry, this implies that also the cost of all insertion operations must be the same.

Theorem 7. *Let d_e be an edit distance where the cost of deleting a single character σ does not depend on σ. Then the* Closest Substring *problem for NFAs under the edit distance d_e can be solved in* PSPACE.

Proof. The input for the problem consists of the length ℓ, radius r and an input NFA A. Let the input alphabet for A be $\Sigma = \{a_1, a_2, \ldots, a_k\}$ and $\min(L(A))$ be the length of the shortest string in $L(A)$. Denote by c_{del} the cost of deleting a single character in d_e, by our assumption the cost does not depend on the character.

First consider the case where $r \geq c_{\text{del}} \cdot |\Sigma| \cdot \min(L(A))$. Without loss of generality we can assume that $\ell > \frac{r}{c_{\text{del}}} \geq |\Sigma| \cdot \min(L(A))$ since otherwise, by Lemma 4, any string of length ℓ has inner distance r to $L(A)$. (Now the value c_{del} is the maximum cost of any single deletion operation.) Choose

$$w_{\text{cons}} = (a_1 a_2 \cdots a_k)^{\min(L(A))} \cdot a_1^{\ell - |\Sigma| \cdot \min(L(A))}.$$

We claim that w_{cons} has inner distance r to $L(A)$. To see this, consider an arbitrary string $z \in L(A)$ and take a substring z_1 of z having length $\min(L(A))$. (z_1 can be any substring of z of length $\min(L(A))$.)

We can align w_{cons} to z_1 by deleting all except $\min(L(A))$ symbols: the symbols not deleted are chosen from the prefix $(a_1 a_2 \cdots a_k)^{\min(L(A))}$ so that they

match z_1 (from each block $a_1 a_2 \cdots a_k$ choose the symbol that appears next in z_1). By Lemma 3, we can assume that $\ell \leq \min(L(A)) + \frac{r}{c_{\mathrm{del}}}$, because otherwise we can simply answer that the input instance does not have a solution. Thus, the number of deleted symbols is at most $\lfloor \frac{r}{c_{\mathrm{del}}} \rfloor$ and the cost is at most r.

The remaining possibility is that $r < c_{\mathrm{del}} \cdot |\Sigma| \cdot \min(L(A))$. This together with $\ell \leq \min(L(A)) + \frac{r}{c_{\mathrm{del}}}$ (from Lemma 3) implies that $\ell, r \in O(n)$, where n is the number of states of the NFA A. This means that, exactly as in the proof of Lemma 5, we can decide nondeterministically in polynomial space whether or not there exists a string of length ℓ with inner distance r to $L(A)$. □

We conjecture that the Closest Substring problem is in PSPACE for any edit distance d_e. However, the proof method used above does not work without the assumption that all individual deletion operations have the same cost.

3.2 Lower bound

Now we mention that the Closest Substring problem for regular languages is NP-hard even when two numerical inputs ℓ and r are given in unary. Recall that the Consensus String problem [9] is, given a finite set S of strings of equal length ℓ, to decide whether or not there exists a radius $r \leq \ell$ consensus string under the Hamming distance metric. It is well known that the Consensus String problem is NP-hard and the length ℓ of the consensus string and the radius r are linear in the size of input. In order to prove the Closest Substring problem is NP-hard, we reduce the Consensus String problem to the Closest Substring problem, where the length ℓ of consensus substring and radius r are given in unary, by relying on the following lemma.

Proposition 8 ([11]). *Let $h : \Sigma^* \to (\Sigma \cup \{\$\})^*, \$ \notin \Sigma$ be a morphism defined by $h(\sigma) = \$^l\sigma$ for all $\sigma \in \Sigma$. Given two strings w and w' of length l over Σ, $d_H(w, w') = d_e(h(w), h(w'))$.*

The following result on the NP-hardness of the Closest Substring problem for regular languages is quite immediate from the well-known fact that the Consensus String problem is NP-complete [9].

Lemma 9. *The Closest Substring problem for DFAs is NP-hard when the length ℓ of closest substring and radius r are given in unary.*

Proof. We reduce the Consensus String problem [9] to the Closest Substring problem where l and r are given in unary notation as part of input.

Thanks to the morphism h in Proposition 8, it is easily seen that the Consensus String problem with the same input under the edit distance metric is also NP-hard. Since we can construct an acyclic DFA accepting the finite set S of strings even using linear space, we can reduce the Consensus String problem to the Closest Substring problem with l and r in unary notation in polynomial time. □

Note that the NP-hardness applies to the case even when the input language is given as an acyclic DFA since the Consensus String problem considers finite sets of equal-length strings as input which can be recognized by acyclic DFAs of polynomial size.

For a non-trivial lower bound, we show that the Closest Substring problem for DFAs is PSPACE-hard by a reduction from the validation of regular expressions of a certain form. First recall the hardness result due to Björklund et al. [2].

Proposition 10 ([2]). *For a given DFA A over the alphabet $\Sigma = \{a, b, c\}$ and a non-negative integer $n \in \mathbb{N}$ in unary, it is* PSPACE*-complete to decide whether or not $L(A) \subseteq \Sigma^* a \Sigma^n b \Sigma^*$.*

The statement of the above result in [2] does not explicitly say that the integer n is given in unary notation, however, exactly the same proof works also with this assumption. We use the following two lemmas to prove the PSPACE-hardness.

Lemma 11. *If w has inner distance at most r to L_2 and $L_1 \subseteq L_2$, then w has inner distance at most r to L_1. If w has inner distance at most r to L_3 and to L_4, then w has inner distance r to $L_3 \cup L_4$.*

In the following always $\Sigma = \{a, b, c\}$, $\Sigma' = \Sigma \cup \{\#, \natural\}$ and for $n \in \mathbb{N}$ consider the edit distance d_0 on Σ' defined by conditions

$$\begin{aligned} &d_0(\#, a) = d_0(\#, b) = d_0(\#, c) = d_0(\#, \natural) = 1, \;\; d_0(\sigma_1, \sigma_2) = 2 \\ &\text{when } \sigma_1, \sigma_2 \in \{a, b, c, \natural\},\ \sigma_1 \neq \sigma_2, \text{ and } d_0(\sigma, \varepsilon) = 2 \text{ for } \sigma \in \Sigma'. \end{aligned} \tag{2}$$

Lemma 12. *Let $\Sigma' = \{a, b, c, \#, \natural\}$ and*

$$L_n = \{\ awb \mid w \in \{a, b\}^n \text{ or } w = c^n \text{ or } w = \natural^n\ \}, \;\; n \in \mathbb{N}. \tag{3}$$

The string $a\#^n b$ has inner distance n to L_n. There is no other string of length $n + 2$ with inner distance n to L_n.

Theorem 13. *There exists an edit distance d_e such that the* Closest Substring *problem for DFAs under the edit distance d_e is* PSPACE*-hard even when the length ℓ of consensus substring and radius r are given in unary.*

Proof. Let $\Sigma = \{a, b, c\}$, $\Sigma' = \Sigma \cup \{\#, \natural\}$ and consider the edit distance d_0 defined in (2). Let L_n be the language defined in (3), $n \in \mathbb{N}$. Let A be a given DFA over Σ and define $L'_n = L(A) \cup L_n$. The language L_n has a DFA of size $3n + 1$. Thus, if A has m states, a DFA for L'_n needs at most $m \cdot (3n + 1)$ states.

By Proposition 10 it is sufficient to show that there is a string of length $n + 2$ over Σ' with inner distance n to L'_n (with respect to the edit distance d_0) if and only if $L(A) \subseteq \Sigma^* a \Sigma^n b \Sigma^*$.

(i) "only if" Let $w \in \Sigma'^*$ be a string of length $n + 2$ with inner distance n to L'_n. By Lemma 11, w has inner distance n also to L_n and hence, by Lemma 12, $w = a\#^n b$. For the sake of contradiction assume that there exists a string

$$u \in L(A) - \Sigma^* a \Sigma^n b \Sigma^*. \tag{4}$$

Since w has inner distance n to $L'_n(\supseteq L(A))$, it follows that u has a substring u' such that $d_0(w, u') \leq n$. Since strings of $L(A)$ do not contain the symbol #, the cost of aligning the n occurrences of # in w with u' is at least n, and the cost is n only when all #'s are substituted to symbols of Σ. Thus, the alignment of w with u' cannot involve any further edit operations and, in particular, $|u'| = n + 2$, u' must begin with an a and end with a symbol b. This contradicts (4).

(ii) "if" By Lemma 12, the inner distance from $a\#^n b$ to L_n is n. Since every string of $L(A)$ has a substring in $a\Sigma^n b$, and the cost of substituting # with a symbol of Σ is one, the inner distance from $a\#^n b$ to $L(A)$ is n. The claim follows from Lemma 11.

This completes the proof. □

We note that even if we are given a particular string w and asked whether the string w is a radius r consensus substring of a given regular language, the problem is still PSPACE-hard since the proof of Theorem 13 requires that only the string $a\#^n b$ must be the consensus substring of $L(A)$. We also note that the distance d_0 used in the proof of Theorem 13 satisfies the property that all symbols have the same deletion cost. This together with Theorem 7 gives:

Corollary 14. *There exists an edit distance d_e such that the* Closest Substring *problem under the edit distance d_e is* PSPACE*-complete both for NFAs and for DFAs.*

Lastly, we establish the following result as a consequence of Lemma 5 and Theorem 13:

Corollary 15. *The* Closest Substring *problem is* PSPACE*-complete both for NFAs and for DFAs when the length ℓ of consensus substring and radius r are given in unary.*

4 The closest substring problem for other formal language classes

We assume that the readers are familiar with the notion of *alternating Turing machine* (ATM) [6, 17] and here only recall the notion informally. The states of an ATM can be *existential* or *universal.* The computation from a configuration with an existential state chooses nondeterministically one possible transition and the computation originating from a configuration with a universal state is continued along all possible paths. The computation accepts if all paths in the resulting tree lead to acceptance. We deal only with polynomial time computations and assume that all computations terminate. The general definition is a little more involved [6].

An ATM M is *k-alternation bounded* ($k \in \mathbb{N}$) if any computation of M—computation is a sequence of configurations—alternates between existential and universal configurations at most $k - 1$ times. For $k \geq 1$, a Σ_k-machine (respectively, a Π_k-machine) is a k-alternation bounded ATM where the initial

state is existential (respectively, universal). By a Σ_0- or Π_0-machine we mean a deterministic Turing machine.

Definition 16 ([6, 17]). *For $k \in \mathbb{N}$, the class of languages accepted by polynomial time Σ_k- (respectively, Π_k-) machines is denoted Σ_k^{P} (respectively, Π_k^{P}).*

The classes Σ_k^{P} and Π_k^{P} are the same as the classes of the polynomial time hierarchy [1, 6]. In particular, $\Sigma_0^{\mathsf{P}} = \Pi_0^{\mathsf{P}} = \mathsf{P}$, $\Sigma_1^{\mathsf{P}} = \mathsf{NP}$ and $\Pi_1^{\mathsf{P}} = \mathsf{coNP}$. The polynomial time hierarchy is contained in PSPACE.

In the following, we show that the Closest Substring problem, where the length ℓ of consensus substring is given in unary, lies in the second level of the polynomial time hierarchy if we consider acyclic NFAs as input.

Theorem 17. *The* Closest Substring *problem for acyclic NFAs is in Σ_2^{P} when the length ℓ of consensus substring is given in unary.*

Proof. An input instance consists of an acyclic NFA A with n states and the length $\ell \in \mathbb{N}$ of consensus substring. If $L(A) \neq \emptyset$, A must accept a string w of length at most $n-1$. Thus, if $\ell \geq n + \lfloor \frac{r}{c_{\text{mindel}}} \rfloor$, no substring of w can be within edit distance r of a string of length ℓ and the algorithm can simply answer "no".

Thus, without loss of generality we can assume that $\ell < n + \lfloor \frac{r}{c_{\text{mindel}}} \rfloor$. A Σ_2-machine M first existentially writes down a string $w \in \Sigma^\ell$. Then using universal branching M follows all computations of A of length at most $n-1$, where $n-1$ is the upper bound on the length of the longest string in $L(A)$, and checks whether or not the underlying string v has a substring in $E(w, d_e, r)$. The universal branch accepts if a substring in $E(w, d_e, r)$ was found or if the computation of A is not accepting, and otherwise the branch rejects. The check can be done by applying a standard edit distance algorithm [18] to all substrings of the underlying string v that have length between $\ell - \lfloor \frac{r}{c_{\text{mindel}}} \rfloor$ and $\ell + \lfloor \frac{r}{c_{\text{mindel}}} \rfloor$. At a given time, the algorithm needs to remember only the last $\ell + \lfloor \frac{r}{c_{\text{mindel}}} \rfloor$ symbols of v and the computation time in one universal branch is polynomial since ℓ and r are both linear in n.

The algorithm checks, for an existentially chosen $w \in \Sigma^\ell$, that all strings of $L(A)$ of length at most n have a substring with edit distance at most r from w using the universal branching. Therefore, we can see that the Closest Substring problem for acyclic NFAs is in Σ_2^P. □

We also show that the Closest Substring problem for acyclic DFAs is coNP-hard when both the length ℓ of consensus substring and radius r are given in unary by reduction from the complement of the Square Tiling problem which is known to be NP-complete [3]. We first prove a similar argument to Proposition 10 for acyclic DFAs instead of general DFAs and further prove that the Closest Substring problem for acyclic DFAs is coNP-hard when the length ℓ of consensus substring and radius r are given in unary.

Lemma 18. *For a given acyclic DFA A over the alphabet $\Sigma = \{a, b, c\}$ and a non-negative integer $n \in \mathbb{N}$ in unary, it is coNP-complete to decide whether $L(A) \subseteq \Sigma^* a \Sigma^n b \Sigma^*$.*

Now we are ready to state the following hardness result.

Theorem 19. *The* Closest Substring *problem for acyclic DFAs is* coNP*-hard even when the length ℓ and radius r are given in unary.*

Proof. By the proof of Theorem 13, we see that the problem of determining $L(A) \subseteq \Sigma^* a \Sigma^n b \Sigma^*$ for a given FA reduces to the Closest Substring problem. Hence, we can reduce the same problem for a given acyclic DFA, which is proven to be coNP-complete in Lemma 18 to the Closest Substring problem for acyclic DFAs when the numeric input values are given in unary. □

Interestingly, the Closest Substring problem is still decidable even when we are given a context-free language as input since we can compute the intersection between a regular language and a context-free language and decide the emptiness of the resulting context-free language. However, it is not clear whether we can solve the problem in PSPACE as well for context-free languages since we cannot restrict the numeric input ℓ and r to be linearly dependent on the size of the representation of context-free languages.

Theorem 20. *The* Closest Substring *problem for context-free languages can be solved in* EXPTIME *when the length ℓ of consensus substring is given in unary.*

Proof. Assume that we are given a pushdown automaton (PDA) P for a context-free language L, namely $L(P) = L$. For every string w of length ℓ, we check whether w is a radius r consensus substring of $L(P)$. We construct an NFA B for $L_1 = \Sigma^* E(w, d_e, r) \Sigma^*$ in polynomial time and obtain a DFA for the complement of L_1 in exponential time by determinizing the NFA B. Then, we can compute the intersection between $L(P)$ and L_1^c in polynomial time in the size of P and the DFA for L_1^c. Since $L(P) \cap L_1^c = \emptyset$ can be decided in exponential time, it is easy to see that the problem can be decided in EXPTIME. □

As a final remark, we mention that the Closest Substring problem is undecidable for context-sensitive languages as the emptiness problem for context-sensitive languages is undecidable.

Corollary 21. *The* Closest Substring *problem for context-sensitive languages is undecidable.*

Acknowledgements

This work was supported by Institute for Information & communications Technology Promotion (IITP) grant funded by the Korea government (MSIP) (2018-0-00255, Autonomous digital companion framework and application).

References

1. J. L. Balcázar, J. Diaz, and J. Gabarró. *Structural Complexity I.* Springer, 2nd edition, 1995.
2. H. Björklund, W. Martens, and T. Schwentick. Validity of tree pattern queries with respect to schema information. In *Proceedings of the 38th International Symposium on Mathematical Foundations of Computer Science*, pages 171–182, 2013.
3. P. V. E. Boas. The convenience of tilings. In *Complexity, Logic, and Recursion Theory*, pages 331–363. Marcel Dekker Inc, 1997.
4. L. Bulteau, F. Hüffner, C. Komusiewicz, and R. Niedermeier. Multivariate Algorithmics for NP-Hard String Problems. *Bulletin of the EATCS*, 114, 2014.
5. C. S. Calude, K. Salomaa, and S. Yu. Additive distances and quasi-distances between words. *Journal of Universal Computer Science*, 8(2):141–152, 2002.
6. A. K. Chandra, D. C. Kozen, and L. J. Stockmeyer. Alternation. *Journal of the ACM*, 28(1):114–133, 1981.
7. M. M. Deza and E. Deza. *Encyclopedia of Distances.* Springer Berlin Heidelberg, 2009.
8. M. R. Fellows, J. Gramm, and R. Niedermeier. On the parameterized intractability of motif search problems. *Combinatorica*, 26(2):141–167, 2006.
9. M. Frances and A. Litman. On covering problems of codes. *Theory of Computing Systems*, 30(2):113–119, 1997.
10. R. L. Graham and N. J. A. Sloane. On the covering radius of codes. *IEEE Transactions on Information Theory*, 31(3):385–401, 2006.
11. Y.-S. Han, S.-K. Ko, T. Ng, and K. Salomaa. Consensus string problem for multiple regular languages. In *Proceedings of the 11th International Conference on Language and Automata Theory and Applications*, pages 196–207, 2017.
12. L. Kari and S. Konstantinidis. Descriptional complexity of error/edit systems. *Journal of Automata, Languages and Combinatorics*, 9:293–309, 2004.
13. D. Marx. Closest substring problems with small distances. *SIAM Journal on Computing*, 38(4):1382–1410, 2008.
14. T. Ng, D. Rappaport, and K. Salomaa. State complexity of neighbourhoods and approximate pattern matching. In *Proceedings of the 19th International Conference on Developments in Language Theory*, pages 389–400, 2015.
15. G. Povarov. Descriptive complexity of the hamming neighborhood of a regular language. In *Proceedings of the 1st International Conference on Language and Automata Theory and Applications*, pages 509–520, 2007.
16. M. L. Schmid. Finding consensus strings with small length difference between input and solution strings. *ACM Transactions on Computation Theory*, 9(3):13:1–13:18, 2017.
17. M. Sipser. *Introduction to the Theory of Computation.* International Thomson Publishing, 1st edition, 1996.
18. E. Ukkonen. Algorithms for approximate string matching. *Information and Control*, 64(1):100–118, 1985.
19. S. Yu. *Regular Languages*, In *Handbook of Formal Languages: Vol. 1*, (G. Rozenberg, A. Salomaa, Eds.), pages 41–110, 1997.