The complexity of string partitioning

Anne Condon[a], Ján Maňuch[a,b,1], Chris Thachuk[c,2]

[a]*Dept. of Computer Science, University of British Columbia, Vancouver BC, Canada*
[b]*Dept. of Mathematics, Simon Fraser University, Burnaby BC, Canada*
[c]*Dept. Computer Science, University of Oxford, Oxford, UK*

Abstract

Given a string w over a finite alphabet Σ and an integer K, can w be partitioned into strings of length at most K, such that there are no *collisions*? We refer to this question as the *string partition* problem and show it is **NP**-complete for various definitions of collision and for a number of interesting restrictions including $|\Sigma| = 2$. This establishes the hardness of an important problem in contemporary synthetic biology, namely, oligo design for gene synthesis.

Keywords: string partitioning, equality-free, prefix-free, suffix-free, factor-free, NP-completeness, collision-aware oligo design, gene synthesis

1. Introduction

Many problems in genomics have been solved by the application of elegant polynomial-time string algorithms, while others amount to solving known **NP**-complete problems; for instance, sequence assembly amounts to solving *shortest common superstring* [1], and genome rearrangement to *sorting strings by reversals and transpositions* [2]. The hardness of these problems has motivated extensive research into heuristic algorithms as well as polynomial-time algorithms for useful restrictions [3, 4, 5, 6, 7, 8, 9]. In a similar vein, we establish the hardness of the following fundamental question: can a string be partitioned into factors (*i.e.* substrings), of bounded length, such that no two *collide*? We refer to this as the *string partition* problem and study it under various restrictions and definitions of what it means for two factors to *collide*.

The study of string partitioning is motivated by an increasingly important problem arising in contemporary synthetic biology, namely gene synthesis. This technology is emerging as an important tool for a number of purposes including the determination of RNAi targeting specificity of a particular gene [10], design of novel proteins [11] and the construction of complete bacterial genomes [12].

Email addresses: `condon@cs.ubc.ca` (Anne Condon), `jmanuch@cs.ubc.ca` (Ján Maňuch), `chris.thachuk@cs.ox.ac.uk` (Chris Thachuk)

[1]Research supported by NSERC Discovery Grant.

[2]Research supported by ERC Advanced Grant VERIWARE and Oxford Martin School.

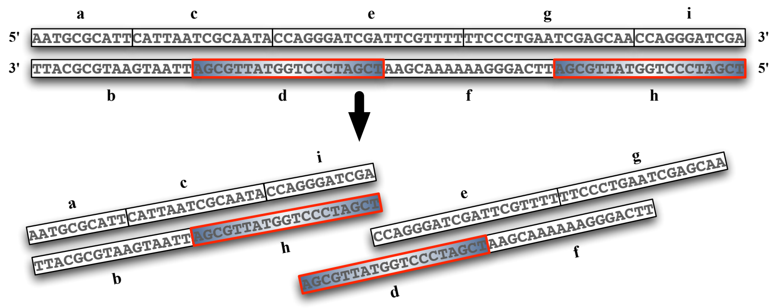
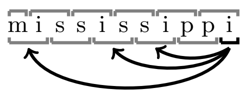


Figure 1: An intended self-assembly (top) of a set of oligos — short strands of DNA, labeled a through h — for a desired DNA duplex. A foiled self-assembly (bottom) of the same oligos due to d and h being identical.

There have been numerous studies utilizing synthetic genes to determine the potential of gene vaccines [13, 14, 15, 16]. Despite the tremendous need for synthetic genes for both interrogative studies and for therapeutics, construction of genes, or any long DNA or RNA sequence, is not a trivial matter. Current technology can only produce short oligonucleotides (oligos) accurately. As such, a common approach is to design a set of oligos — short sequences of nucleic acids — that could assemble into the desired sequence [17].

To understand the connection between string partitioning and gene synthesis, consider the following. A DNA *oligo*, or *strand* is a string over the four letter alphabet $\{\mathtt{A}, \mathtt{C}, \mathtt{G}, \mathtt{T}\}$. The *reverse complement* F' of an oligo F is determined from F by replacing each `A` with a `T` and vice versa, each `C` with a `G` and vice versa, and reversing the resulting string. Two DNA oligos F and F' are said to *hybridize* if a sufficiently long factor of F is the reverse complement of a factor of F' (see Figure 1). A DNA *duplex* consists of a positive strand and its reverse complement, the negative strand. The *collision-aware oligo design for gene synthesis* (CA-ODGS) problem is to determine cut points in the positive and negative strands, which demarcate the oligos to be synthesized, such that the resulting design will successfully self-assemble. For the oligos to self-assemble correctly, they should 1) alternate between the positive and negative strands, with some overlap between successive oligos, and 2) only hybridize to the oligos they overlap with by design. Since there is variability in the length of the selected oligos, there are exponentially many designs.

In previous work [18], the authors provided some evidence that the CA-ODGS problem may be hard by showing that partitioning a string into factors, of bounded length, such that no two are equal is **NP**-complete, even for strings over a quaternary alphabet. See Figure 1 for an example design that assembles incorrectly into two fragments, with the wrong ordering of oligos and therefore primary sequence, due to identical oligos. In this work, we study the underlying string partition problem in much greater detail. We show that partitioning strings such that no selected string is a copy/factor/prefix&suffix/prefix/suffix



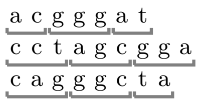


Figure 2: (Left) Two partitions are shown for the string *mississippi*. The selected strings in both partitions have maximum length 2. The partition shown above the string is factor-free: no selected string is a factor of another; however, the partition shown below the string is not factor-free. (Right) A valid factor-free multiple string partition of a set of three strings into selected strings of maximum length 3.

of another is **NP**-complete. We begin by showing that the more general problem of partitioning a set of strings is hard and then we show how those instances can be reduced to single string instances, for each respective definition of collision. See Figure 2 for an example of a single string instance (left) and set of strings instance (right). In all cases, we demonstrate that the problems remain hard even when restricted to binary strings. A preliminary version of this work with the proofs for the equality-free case has appeared in [19].

2. Preliminaries

A *string* w is a sequence of letters over an alphabet Σ. Let $|w|$ denote the length of w, w^{R} a mirror image (reversal) of w, and let $(w)^i$ denote the string w repeated i times. The empty string is denoted as ε. String x is a *factor* of w if $w = \alpha x \beta$, for some (possibly empty) strings α and β. Similarly, x is a *prefix* (*suffix*) of w if $w = x\beta$ ($w = \alpha x$) for some (possibly empty) strings α and β. The prefix (suffix) of length k of w will be denoted as $\mathrm{prefix}_k(w)$ ($\mathrm{suffix}_k(w)$). Similarly, let $\mathrm{factor}_{i,j}(w)$ denote the factor of w of length $j - i$ starting at position i. Note that $\mathrm{prefix}_i(w) = \mathrm{factor}_{1,i+1}(w)$.

A K-*partition* of w is a sequence $P = p_1, p_2, \ldots, p_l$, for some l, where each p_i is a string over Σ of length at most K and $w = p_1 p_2 \ldots p_l$. We say that strings $p_1, \ldots, p_l$ are *selected* in the K-partition and that strings $p_i \ldots p_j$, $1 \leq i \leq j \leq l$, are *super-selected*, with respect to the selected strings. We say P is *equality-free*, *prefix-free*, *suffix-free*, *prefix&suffix-free* or *factor-free* if for all i, j, $1 \leq i \neq j \leq l$, neither p_i nor p_j is a copy, prefix, suffix, prefix or suffix, or factor, respectively, of the other. We say such partitions are *valid* (for the problem in question); otherwise, we say the partition contains a *collision*. We generalize the notion of a K-partition to a set of strings $\mathcal{W}$ to mean a K-partition for each string in $\mathcal{W}$. The length of $\mathcal{W}$ is the combined length of the strings in the set and will be denoted by $\|\mathcal{W}\|$. A K-partition for a set of strings is valid if no two elements in any, possibly different, partition collide. Finally, we will refer to the boundaries of a partition of string w as *cut points*, where the first cut point 0 and the last cut point $|w|$ are called trivial. For instance, the first partition of mississippi in Figure 2 has the following non-trivial cut points $1, 3, 5, 7$ and 9. We say that a partition P is a *refinement* of partition P' if the set of cut points of P is a superset of the set of cut points of P', i.e., if every string selected in P' is super-selected in P.

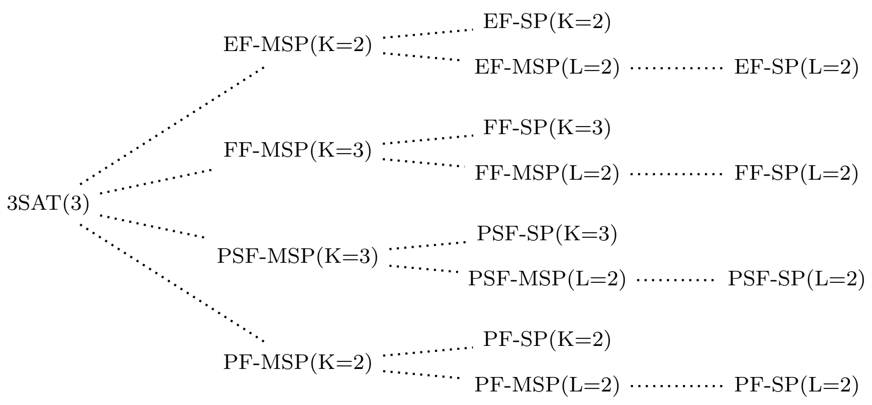


Figure 3: Chain of reductions for different string partition variations from original 3SAT(3) problem. K is maximum selected string size and L is maximum alphabet size. Parameters are unbounded if not shown. EF, FF, PSF and PF are equality-free, factor-free, prefix&suffix-free and prefix(suffix)-free, respectively.

In what follows we will prove **NP**-completeness of various string partitioning problems by showing a polynomial reduction from an arbitrary instance of 3SAT(3), a problem shown to be **NP**-complete by Papadimitriou [20].

Problem 1 (3SAT(3)).
Instance: A formula ϕ with a set C of clauses over a set X of variables in conjunctive normal form such that:

1. every clause contains two or three literals,
2. each variable occurs in exactly three clauses, once negated and twice positive.

Question: Is ϕ satisfiable?

3. The String Partition Problems

For each $\mathcal{X} : \mathcal{X}'$ in $\{\text{equality : E}, \text{prefix\&suffix : PS}, \text{prefix(suffix) : P}, \text{factor : F}\}$, we will consider two string partition problems.

Problem 2 ($\mathcal{X}$-Free Multiple String Partition ($\mathcal{X}'$F-MSP) Problem).

Instance: Finite alphabet Σ of size L, a positive integer K, and a set of strings $\mathcal{W}$ in Σ^*.
Question: Is there an $\mathcal{X}$-free, K-partition P of $\mathcal{W}$?

Problem 3 ($\mathcal{X}$-Free String Partition ($\mathcal{X}'$F-SP) Problem).
Instance: Finite alphabet Σ of size L, a positive integer K, and a string w in Σ^*.
Question: Is there an $\mathcal{X}$-free, K-partition P of w?

We will show **NP**-completeness of all these problems even when restricted to the constant size of the partition ($K = 2, 3$), or to the binary alphabet ($L = 2$). See Figure 3 showing the chain of reductions used to prove the complexity of the four variations and related restrictions of the problem.

4. Equality-Free String Partition Problems

In this section, we will show **NP**-completeness of Equality-Free String Partition problems even when restricted to the constant size of the partition ($K = 2$), or to the binary alphabet ($L = 2$).

4.1. Equality-Free Multiple String Partition with Unbounded Alphabet

We now describe a polynomial reduction from 3SAT(3) to EF-MSP with $K = 2$ and unbounded alphabet. Let ϕ be an instance of 3SAT(3), with set $C = \{c_1, \ldots, c_m\}$ of clauses, and set $X = \{x_1, \ldots, x_n\}$ of variables. We shall define an alphabet Σ and construct a set of strings $\mathcal{W}$ in Σ^*, such that $\mathcal{W}$ has a collision-free 2-partition if and only if ϕ is satisfiable. Let $|c_i|$ denote the number of literals contained in the clause c_i and let $c_i^1, \ldots, c_i^{|c_i|}$ be the literals of clause c_i.

We construct $\mathcal{W}$ to be a union of three types of strings: clause strings ($\mathcal{C}$), enforcer strings ($\mathcal{E}$) and forbidden strings ($\mathcal{F}$). First, for each clause of ϕ, we create a clause string C such that an equality-free 2-partition of $\mathcal{C}$ unambiguously selects exactly one literal from C. We refer to the selected strings corresponding to literals as *selected literals*. Intuitively, the selected literals of the clause strings are intended to be a satisfying truth assignment for the variables of ϕ. Second, for each variable we create an enforcer string to ensure that selected literals are *consistent*. Specifically, the enforcer strings ensure that a positive and a negative literal for the same variable cannot be simultaneously selected. Finally, we find it helpful to create so called forbidden strings that ensure certain strings cannot be selected in the clause and enforcer strings.

We construct an alphabet Σ, which includes a letter for each literal occurrence in the clauses, one letter for each variable, and the letters $\boxminus$ and $\boxplus$ used as delimiters, as follows:

$$\Sigma = \{\hat{x}_i;\ x_i \in X\} \cup \{\hat{c}_i^j;\ c_i \in C \wedge 1 \leq j \leq |c_i|\} \cup \{\boxminus, \boxplus\}.$$

Note that $|\Sigma|$ is linear in the size of the 3SAT(3) problem ϕ (at most $n + 3m + 2$).

Construction of forbidden strings. To ensure that certain strings cannot be selected in $\mathcal{C}$ or $\mathcal{E}$, we will use the following set of forbidden strings $\mathcal{F} = \{\boxminus, \boxplus\}$.

Observation 1. *No string from the forbidden set $\mathcal{F}$ can be selected in $\mathcal{C}$ or $\mathcal{E}$.*

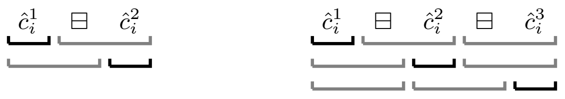


Figure 4: The 2-literal clause string (left) and 3-literal clause string (right) used in the reduction from 3SAT(3) to EF-MSP. Shown below each string are all valid 2-partitions. *Selected* literals of a partition are shown in black.

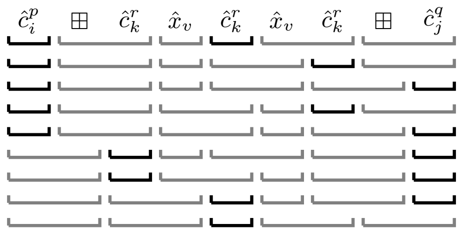


Figure 5: All possible 2-partitions are shown for the enforcer string of a variable x_v having two positive literals c_i^p and c_j^q, and one negative literal c_k^r. In each partition, either $\hat{c}_k^r$ is selected or both $\hat{c}_i^p$ and $\hat{c}_j^q$ are which guarantees that letters for positive and negated literals of x_v cannot be simultaneously selected in $\mathcal{C}$.

Construction of clause strings. For each clause $c_i \in C$, construct the *i-th clause string* to be $\hat{c}_i^1 \boxminus \hat{c}_i^2$ if $|c_i| = 2$, and $\hat{c}_i^1 \boxminus \hat{c}_i^2 \boxminus \hat{c}_i^3$ if $|c_i| = 3$.

Lemma 1. *Given that no string from the forbidden set $\mathcal{F}$ is selected in $\mathcal{C}$, exactly one literal letter must be selected for each clause string in any equality-free 2-partition of $\mathcal{C}$.*

Proof. Consider the clause string for clause c_i. Whether c_i has two or three literals, the forbidden substring $\boxminus$ cannot be selected alone. Therefore, each $\boxminus$ must be selected with an adjacent literal letter. This leaves exactly one other literal letter which must be selected (see Figure 4). $\square$

Construction of enforcer strings. We must now ensure that no literal of ϕ that is selected in $\mathcal{C}$ is the negation of another selected literal. By definition of 3SAT(3), each variable appears exactly three times: twice positive and once negated. Let c_i^p and c_j^q be the two positive and c_k^r the negated occurrences of a variable x_v. Then construct the *enforcer string* for this variable as follows $\hat{c}_i^p \boxplus \hat{c}_k^r \hat{x}_v \hat{c}_k^r \hat{x}_v \hat{c}_k^r \boxplus \hat{c}_j^q$.

Lemma 2. *Given that no string from the forbidden set $\mathcal{F}$ is selected in $\mathcal{C} \cup \mathcal{E}$, any equality-free 2-partition of $\mathcal{C} \cup \mathcal{E}$ must be consistent. In addition, for any consistent choice of selecting letters for literals in $\mathcal{C}$, there is an equality-free 2-partition of $\mathcal{C} \cup \mathcal{E} \cup \mathcal{F}$.*

Proof. Consider the enforcer string for variable x_v with positive literals $c_i^p = c_j^q = x_v$, and the negated literal $c_k^r = \neg x_v$. Figure 5 shows all 9 possible 2-partitions of the enforcer string (since $\boxplus$ is a forbidden string, each $\boxplus$ must be selected with an adjacent letter). It follows that in each of them either $\hat{c}_k^r$ is selected or both $\hat{c}_i^p$ and $\hat{c}_j^q$ are. In the first case, $\hat{c}_k^r$ cannot be selected in $\mathcal{C}$ and thus satisfied literals are chosen consistently for x_v. In the second case, letters for neither of the positive occurrences of x_v can be selected in $\mathcal{C}$.

To show the second part of the claim, observe that there is a 2-partition of the enforcer string compatible with any of four valid combinations of selecting letters for the corresponding literals in $\mathcal{C}$ (for example, by choosing the fifth or the last 2-partitions in Figure 5). Since enforcer strings share only one letter in common, namely, $\boxplus$, which is never selected in the enforcer strings, there are no collisions between 2-partitions of all enforcer strings. Furthermore, there are no collisions between strings selected in $\mathcal{C}$ and in $\mathcal{E}$: strings of length two selected in $\mathcal{C}$ contain the letter $\boxminus$, which does not appear in the enforcer strings; strings of length one are literals and the partitioning of enforcer strings was chosen in a way that literals (in $\mathcal{C}$) cannot be selected again in $\mathcal{E}$. $\square$

This completes the reduction. Notice that the reduction is polynomial as the combined length of the constructed set of strings $\mathcal{W} = \mathcal{C} \cup \mathcal{E} \cup \mathcal{F}$ is at most $5m + 9n + 2$.

Theorem 1. *Equality-Free Multiple String Partition (EF-MSP) is **NP**-complete for partition size $K = 2$.*

Proof. It is easy to see that EF-MSP Problem is in **NP**: a nondeterministic algorithm needs only guess a partition P where $|p_i| \leq K$ for all p_i in P and check in polynomial time that no two strings in P are equal. Furthermore, it is clear that an arbitrary instance ϕ of 3SAT(3) can be reduced to an instance of EF-MSP, specified by a set of strings $\mathcal{W} = \mathcal{C} \cup \mathcal{E} \cup \mathcal{F}$, in polynomial time and space by the reduction detailed above.

Now suppose there is a satisfying truth assignment for ϕ. Simply select one corresponding true literal per clause in $\mathcal{C}$. The construction of clause strings guarantees that a 2-partition of the rest of each clause string is possible. Also, since a satisfying truth assignment for ϕ cannot assign truth values to opposite literals, then Lemma 2 guarantees that a valid partition of the enforcer strings is possible which does not conflict with the clause strings. Therefore, there exists an equality-free multiple string partition of $\mathcal{W}$.

Likewise, consider an equality-free multiple string partition of $\mathcal{W}$. Lemma 1 ensures that at least one literal per clause is selected. Furthermore, Lemma 2 guarantees that if there is no collision, then no two selected variables in the clauses are negations of each other. Therefore, this must correspond to a satisfying truth assignment for ϕ (if none of the three literals of a variable is selected in the partition of $\mathcal{C}$ then this variable can have arbitrary value in the truth assignment without affecting satisfiability of ϕ). $\square$

4.2. Equality-Free String Partition with Unbounded Alphabet

The result of the previous subsection can be easily extended for a single string by concatenating the multiple strings in a way that guarantees that each original string is super-selected.

Theorem 2. *Equality-Free String Partition (EF-SP) is* ***NP****-complete for partition size $K = 2$.*

Proof. To show that EF-SP Problem for $K = 2$ is **NP**-complete, we will reduce EF-MSP Problem for $K = 2$ to it. Consider an arbitrary instance I of EF-MSP having a set of strings $\mathcal{W} = \{w_1, w_2, \dots, w_\ell\}$ over alphabet Σ, and maximum partition size $K = 2$. We construct an instance $\bar{I}$ of EF-SP as follows. Let $\widehat{\Sigma} = \{\boxdot\} \cup \{d_i, \text{ for } 1 \le i < \ell\}$, where $\widehat{\Sigma} \cap \Sigma = \emptyset$. Set the alphabet of $\bar{I}$ to $\bar{\Sigma} = \Sigma \cup \widehat{\Sigma}$ and the maximum partition size to $\bar{K} = 2$. Note that $|\bar{\Sigma}| = |\Sigma| + \ell$. Finally, construct the string

$$\bar{w} = \boxdot\ \boxdot\ \boxdot\ \boxdot\ \boxminus w_1 d_1\ \boxdot\ \boxdot d_1 w_2 d_2\ \boxdot\ \boxdot d_2 \dots d_{\ell-1}\ \boxdot\ \boxdot d_{\ell-1} w_\ell\,.$$

The prefix of $\bar{w}$ of length five can be partitioned in two different ways each selecting $\boxdot$. Consequently, in any 2-partition of $\bar{w}$, remaining occurrences of $\boxdot$ must be selected together with an adjacent letter different from $\boxdot$, i.e., all strings $d_i\boxdot$ and $\boxdot d_i$ must be selected. Therefore, any 2-partition of $\bar{w}$ contains a 2-partition of $\mathcal{W}$ and the strings $\mathcal{D} = \{\boxdot, \boxdot\boxdot, \boxdot\boxminus, d_1\boxdot, \boxdot d_1, \dots, d_{\ell-1}\boxdot, \boxdot d_{\ell-1}\}$. On the other hand, since all strings in $\mathcal{D}$ contain $\boxdot \notin \Sigma$, any 2-partition of $\bar{w}$ together with $\mathcal{D}$ forms a 2-partition of $\mathcal{W}$. It follows that there is a 2-partition of $\mathcal{W}$ if and only if there is a 2-partition of $\bar{w}$. The reduction is in polynomial time and space as $|\bar{w}| = \|\mathcal{W}\| + 4\ell + 1$. □

4.3. Equality-Free Multiple String Partition with Binary Alphabet

We will now encode all strings by mapping letters of unbounded alphabet to binary strings of length $\delta \ge \log_2 |\Sigma|$. In addition, we will add a large set of forbidden strings that will guarantee that there are no cut points inside these coding binary strings.

Theorem 3. *The EF-MSP with maximum partition size $K = 2$ can be polynomially reduced to the EF-MSP Problem with the alphabet size $L = 2$. Consequently, the EF-MSP is* ***NP****-complete for binary alphabet. In addition, this reduction satisfies the following property: for any set C containing n distinct binary strings of length δ, where n is the size of the alphabet of the EF-MSP with maximum partition size $K = 2$ and $\delta \ge \log_2 n$, every selected string in a valid partition (if it exists) of the EF-MSP with the binary alphabet is a prefix of a string in C^2, and its maximum partition size is $\bar{K} = 2\delta$.*

Proof. We will show a reduction from the EF-MSP with maximum partition size $K = 2$. Consider an arbitrary instance I of EF-MSP having a set of strings $\mathcal{W} = \{w_1, w_2, \dots, w_\ell\}$ over alphabet $\Sigma = \{a_1, \dots, a_n\}$, and maximum partition size $K = 2$. We will construct an instance $\bar{I}$ of EF-MSP over binary alphabet $\bar{\Sigma} = \{0, 1\}$. Let δ be any number greater or equal to $\log_2 n$. Let $C = \{c_1, \dots, c_n\}$ be a set of any distinct binary codewords of length δ. We set $\bar{K}$ to 2δ. Let h be a homomorphism from Σ to C such that $h(a_i) = c_i$, for every $i = 1, \dots, n$. The set of strings of $\bar{I}$ will contain $h(\mathcal{W})$, i.e., the original strings in $\mathcal{W}$ mapped by h to the binary alphabet $\bar{\Sigma}$. However, we need to guarantee that the partition of strings in $h(\mathcal{W})$ does not contain fragments of codewords. For this reason, we also add to $\bar{\mathcal{W}}$ the following strings:

$$\widehat{\mathcal{W}} = \{\text{prefix}_i(c);\ c \in C, i = 1, \dots, \delta - 1\} \cup \\ \{\text{prefix}_i(cd);\ c, d \in C, i = \delta + 1, \dots, 2\delta - 1\}$$

We set $\bar{\mathcal{W}} = h(\mathcal{W}) \cup \widehat{\mathcal{W}}$.

First, consider a valid 2-partition P of $\mathcal{W}$. We construct a $\bar{K}$-partition $\bar{P}$ of $\bar{\mathcal{W}}$ as follows. For each string s selected in P, we select the corresponding $h(s)$ in $\bar{P}$. For each string $t \in \widehat{\mathcal{W}}$, we select t entirely. Note that strings selected from $h(\mathcal{W})$ have length either δ or 2δ, while strings selected from $\widehat{\mathcal{W}}$ have lengths different from δ and 2δ. Therefore, there cannot be any collisions between these two groups of selected strings. Furthermore, there are no collisions in the first group, since there were no collisions in P. Obviously, there are no collisions in the second group of selected strings. It follows that $\bar{P}$ is a valid $\bar{K}$-partition of $\bar{\mathcal{W}}$.

Conversely, consider a valid $\bar{K}$-partition $\bar{P}$ of $\bar{\mathcal{W}}$. First, we will show that all strings in $\widehat{\mathcal{W}}$ are selected without non-trivial cut points. We will prove that by induction on the length i of strings. The base case, $i = 1$, is trivially true, as one-letter strings cannot be partitioned into shorter strings. Now, assume the claim is true for all strings in $\widehat{\mathcal{W}}$ of lengths smaller than $i < 2\delta$ and different from δ. Consider a string $u \in \widehat{\mathcal{W}}$ of length i. Assume that u is partitioned into strings $u_1, \dots, u_t$, where $t \geq 2$. Note that the length of u_1 is smaller than i. If the length of u_1 is different from δ, we have a collision, as $u_1 \in \widehat{\mathcal{W}}$ and by the induction hypothesis, it was selected without non-trivial cut points. Assume that the length of u_1 is δ. Then u_2 is a prefix of a codeword of length smaller than $\min\{\delta, i\}$, and we have a collision again as in the previous case. It follows that $t = 1$, i.e., u is selected without non-trivial cut points in $\bar{P}$. Second, we show that all strings selected in the partition of strings in $h(\mathcal{W})$ have lengths either δ or 2δ. Assume that this is not the case for some string $s \in h(\mathcal{W})$. Note that $s = c_{i_1} c_{i_2} \dots c_{i_p}$, for some indices $i_1, \dots, i_p$. Let $s = s_1 \dots s_q$ be the partition of s and let j be the smallest j such that the length of s_j is not δ or 2δ. Then $s_1 \dots s_{j-1} = c_{i_1} \dots c_{i_r}$, for some $r < p$. Consequently, s_j is a prefix of $c_{i_{r+1}} c_{i_{r+2}}$, i.e., $s_j \in \widehat{\mathcal{W}}$, and we have a collision, since s_j was already selected in partition of $\widehat{\mathcal{W}}$. Hence, each string in $h(\mathcal{W})$ is partitioned into strings of lengths either δ or 2δ, which can be easily mapped to a valid 2-partition of $\mathcal{W}$.

It follows that there is a 2-partition of $\mathcal{W}$ if and only if there is $\bar{K}$-partition of $\bar{\mathcal{W}}$ and that the reduction satisfies the property described in the claim.

Finally, let us check that the reduction is polynomial. The size of $h(\mathcal{W})$ is $|\mathcal{W}|$ and the length of $h(\mathcal{W})$ is $\delta\|\mathcal{W}\|$. The size of $\widehat{\mathcal{W}}$—the set of all unique prefixes for codewords of length less than δ, and all unique prefixes of pairs of adjacent codewords with length greater than δ and less than 2δ—is at most $(n^2 + n)(\delta - 1)$ as there are n codewords in total. Therefore, the length of $\widehat{\mathcal{W}}$ is at most $n \cdot (1 + \cdots + \delta - 1) + n^2 \cdot (\delta + 1 + \delta + 2 + \cdots + 2\delta - 1) = (3n^2 + n)(\delta - 1)\delta/2$. Since δ can be chosen to be $\Theta(\log n)$, the size of $\bar{\mathcal{W}}$ is polynomial in the size of $\mathcal{W}$ and the size of the original alphabet Σ. $\square$

4.4. Equality-Free String Partition with Binary Alphabet

The following final reduction to a single string requires a quite technical set of delimiter strings.

Theorem 4. *Equality-Free String Partition (EF-SP) is* ***NP****-complete for binary alphabet ($L = 2$).*

Proof. We will show a reduction from the EF-MSP Problem with the binary alphabet ($L = 2$) satisfying properties listed in Theorem 3. Consider an instance I of EF-MSP having a set of strings $\mathcal{W} = \{w_1, w_2, \dots, w_\ell\}$ over alphabet $\Sigma = \{0, 1\}$, and maximum partition size $K = 2\delta$ such that all selected strings in any valid K-partition are prefixes of the elements of a set C^2, where C contains n distinct strings of length δ each starting with 0, $\ell \leq (n^2 + n)(\delta - 1)$, and $\delta \geq \max(9, 3\log_2(n + 1))$. By Theorem 3, this instance can be polynomially reduced to an instance of the EF-MSP with maximum partition size $K = 2$. We will construct an instance $\bar{I}$ of EF-SP over binary alphabet $\bar{\Sigma} = \{0, 1\}$ with the same partition size $K = 2\delta$. We will show that the size of $\bar{I}$ is polynomial in the size of I, and hence, it will follow by Theorems 1 and 3, that the EF-SP Problem is **NP**-complete.

To construct the string $\bar{w}$ we will interleave strings $w_1, \dots, w_\ell$ with delimiters $d_1, \dots, d_{\ell-1}$ defined in a moment as follows:

$$\bar{w} = w_1 d_1 w_2 d_2 w_3 \dots d_{\ell-1} w_\ell \,.$$

To define the delimiter strings, we will need the following functions. Let bin : $\mathbb{N} \to \{0, 1\}^*$ be a function mapping a positive integer to its standard binary representation without the leading one. For example $\mathrm{bin}(1) = \varepsilon$, $\mathrm{bin}(2) = 0$ and $\mathrm{bin}(10) = 010$. Next, the functions $\mathrm{pad}_i : \{0, 1\}^* \to \{0, 1\}^*$ will pad a given string with $i - 1$ ones and one zero on the left, i.e., $\mathrm{pad}_i(s) = (1)^{i-1}0s$. We will refer to strings returned by these functions as *padded strings*. The function chain : $\{0, 1\}^* \to \{0, 1\}^*$ maps a string s with i trailing zeros, i.e., $s = s'(0)^i$, where s' is either the empty string or a string ending with 1, to the following concatenation of padded strings and mirror images (reversals) of padded strings:

$$\begin{aligned}\mathrm{chain}(s) = \mathrm{pad}^{\mathrm{R}}_{K-|s|}(s)\, \mathrm{pad}_{K-|s|}(s')\, \mathrm{pad}^{\mathrm{R}}_{K-|s|}(s')\, \mathrm{pad}_{K-|s|}(s'0)\, \mathrm{pad}^{\mathrm{R}}_{K-|s|}(s'0) \\ \dots \mathrm{pad}_{K-|s|}(s'(0)^{i-1})\, \mathrm{pad}^{\mathrm{R}}_{K-|s|}(s'(0)^{i-1})\, \mathrm{pad}_{K-|s|}(s) \,.\end{aligned}$$

Finally, we set the delimiter d_j to chain(bin(j)), for every $j > 1$. For $j = 1$, we set d_1 to $0(1)^{K-1}(1)^{K(K-1)/2}(1)^{K-1}0$. To illustrate this definition, let us list the first five delimiter strings:

$$\begin{aligned}
d_1 &= 0(1)^{K-1}(1)^{K(K-1)/2}(1)^{K-1}0 \\
d_2 &= \mathrm{chain}(0) = 00(1)^{K-2}(1)^{K-2}00(1)^{K-2}(1)^{K-2}00 \\
d_3 &= \mathrm{chain}(1) = 10(1)^{K-2}(1)^{K-2}01 \\
d_4 &= \mathrm{chain}(00) = 000(1)^{K-3}(1)^{K-3}00(1)^{K-3}(1)^{K-3}0000(1)^{K-3}(1)^{K-3}000 \\
d_5 &= \mathrm{chain}(01) = 100(1)^{K-3}(1)^{K-3}001
\end{aligned}$$

Now, consider a valid K-partition P of $\mathcal{W}$. We construct a K-partition $\bar{P}$ of $\bar{w}$ as follows. Each substring w_j is partitioned in the same way as in P. Each delimiter d_j, where $j > 1$, is partitioned to its padded strings and mirror images of padded strings. In addition, the delimiter d_1 is partitioned into one mirror image of a padded string, strings $(1), (1)^2, \ldots, (1)^K$ in any order, and one padded string. Note that all strings selected in w_j's are prefixes of C^2, and since each $c \in C$ has length $\delta = K/2$ and starts with 0, all these selected strings start with 0 and the longest run of 1 they contain has length at most $\delta - 1$. Hence, they cannot collide with strings $(1), (1)^2, \ldots, (1)^K$ and with padded strings which all start with 1. To show they do not collide with mirror images of padded strings, we will show that each padded string (or its mirror image) contains a run of at least δ ones. By the definition of functions pad_i, each padded string or its mirror image selected in a delimiter d_j contains a substring $(1)^{K-|\,\mathrm{bin}(d_j)|-1}$, i.e., a run of $K - (\lceil \log_2 j \rceil - 1) - 1 = K - \lceil \log_2 j \rceil$ ones. Since $j < \ell \le (n^2 + n)(\delta - 1)$, it is enough to show that $\log_2[(n^2 + n)(\delta - 1)] \le \delta$. This follows from the fact that $\delta \ge 2\log_2(n + 1) + \delta/3$ and $\delta/3 \ge \log_2(\delta - 1)$ for $\delta \ge 9$. Finally, we need to show that all selected padded strings and their mirror images are distinct. Note that each selected padded string starts with at least δ ones and contains at least one zero, hence, it cannot be equal to a selected mirror image of padded string. Hence, it is enough to show that two delimiter d_j and $d_{j'}$, where $j, j' < \ell$ do not contain the same padded string or its mirror image. Without loss of generality, let us only consider the padded strings. If $\mathrm{bin}(j)$ and $\mathrm{bin}(j')$ have different lengths then the padded strings of d_j and $d_{j'}$ start with $(1)^{K-|\,\mathrm{bin}(j)|-1}0$ and $(1)^{K-|\,\mathrm{bin}(j')|-1}0$, hence they cannot be equal. Therefore, assume they have the same length. Let s (respectively, s') be the prefix of $\mathrm{bin}(j)$ (respectively, $\mathrm{bin}(j')$) without the trailing zeros. Clearly, $s \ne s'$. Now, the padded strings from d_j and $d_{j'}$ are the same only if $s0^i = s'0^{i'}$ for some i and i'. However, since both s and s' end with one or one of them is the empty string, we must have $i = i'$, and hence also $s = s'$, a contradiction. Since the K-partition P of $\mathcal{W}$ was valid, it follows that the K-partition of $\bar{w}$ is also valid.

Conversely, consider a valid K-partition $\bar{P}$ of $\bar{w}$. It is enough to show that $\bar{P}$ super-selects each delimiter in $\bar{w}$. We will show by induction on j that delimiters $d_1, \ldots, d_j$ are super-selected and furthermore, that each of these delimiters is partitioned into its padded strings and mirror images of padded strings. For the base case $j = 1$, it is easy to see that $\bar{P}$ must select string $0(1)^{K-1}$, then strings $(1)^1, \ldots, (1)^K$ in any order and string $(1)^{K-1}0$, and thus d_1 is super-selected in $\bar{P}$ and its padded string and its mirror image of a padded string are selected. Next, assume that the induction hypothesis is satisfied for delimiters $d_1, \ldots, d_{j-1}$. Consider delimiter string d_j. First, we will show that d_j contains cut points in $\bar{P}$ shown by symbols "$\cdot$" below:

$$\mathrm{pad}^{\mathrm{R}}_{K-|s|}(s) \cdot \mathrm{pad}_{K-|s|}(s')\, \mathrm{pad}^{\mathrm{R}}_{K-|s|}(s') \cdot \mathrm{pad}_{K-|s|}(s'0)\, \mathrm{pad}^{\mathrm{R}}_{K-|s|}(s'0) \cdot \\ \ldots \cdot \mathrm{pad}_{K-|s|}(s'(0)^{i-1})\, \mathrm{pad}^{\mathrm{R}}_{K-|s|}(s'(0)^{i-1}) \cdot \mathrm{pad}_{K-|s|}(s)\,,$$

where $s = \mathrm{bin}(j)$ and s' is the prefix of s without the trailing zeros and i

is the number of trailing zeros. Note that each symbol "$\cdot$" is preceded and followed by $K - |\operatorname{bin}(j)| - 1$ ones. Since $|\operatorname{bin}(j)| \leq \delta - 1$, we have a run of at least $K = 2\delta$ ones, thus this run must contain a cut point. By contradiction assume that there is a cut point before the position marked by symbol "$\cdot$" in this run of ones. Then the selected string starting at this cut point is in the form $(1)^{K-i-1}0u$, where $i < |\operatorname{bin}(j)|$ and $|u| \leq i$. Note that u might be the empty string and the selected string must contain the zero preceding u since all strings consisting only of ones are already selected in d_1. Let $v = u(0)^{i-|u|}$. Since $|v| = i < |\operatorname{bin}(j)|$, we have $v = \operatorname{bin}(j')$, where $j' < j$. The delimiter string $d_{j'}$ contains $\operatorname{pad}_{K-i}(u) = (1)^{K-i-1}0u$, which by the induction hypothesis has been already selected. Analogously, we arrive into a contradiction, if there is a cut point after "$\cdot$" in the run of ones surrounding the letter "$\cdot$". It follows that there is a cut point at each symbol "$\cdot$" above in $\bar{P}$.

Next, we show that each of super-selected strings of d_j:

$$\operatorname{pad}_{K-|s|}(s')\operatorname{pad}^{\mathrm{R}}_{K-|s|}(s'), \ldots, \operatorname{pad}_{K-|s|}(s'(0)^{i-1})\operatorname{pad}^{\mathrm{R}}_{K-|s|}(s'(0)^{i-1}),$$

has a cut point exactly in the middle. The length of each padded string or of its mirror image is at least $K - |s|$ and since $|\operatorname{bin}(j)| \leq \delta - 1$, this length is at least $\delta + 1$. Hence, there has to be at least one cut point in each of the above super-selected strings in $\bar{P}$. We will first prove the claim for the first super-selected string $\operatorname{pad}_{K-|s|}(s')\operatorname{pad}^{\mathrm{R}}_{K-|s|}(s')$. By contradiction, and without loss of generality, assume that there is a cut point inside $\operatorname{pad}_{K-|s|}(s') = (1)^{K-|s|-1}0s'$. Thus a string in the form $(1)^{K-|s|-1}0u$, where u is a proper prefix of s', is selected in $\bar{P}$. Consider string $v = u(0)^{|s|-|u|}$. Obviously, $|v| = |s|$ and v is lexicographically smaller than s, and thus $\operatorname{bin}(j') = v$ for some $j' < j$. By the induction hypothesis, string $\operatorname{pad}_{K-|v|}(u) = (1)^{K-|s|-1}0u$ has been already selected in $d_{j'}$, a contradiction. It follows by straightforward induction on i that the remaining super-selected strings are partitioned exactly in the middle. Finally, observe that if there is a cut point inside $\operatorname{pad}_{K-|s|}(s)$ then either one of the padded strings of $d_{j'}$ or one of the padded strings of d_j described above is selected again. Similarly, there cannot be any cut point inside $\operatorname{pad}^{\mathrm{R}}_{K-|s|}(s)$. Since the length of these two strings is exactly K, there has to be a cut point just after $\operatorname{pad}_{K-|s|}(s)$ and just before $\operatorname{pad}^{\mathrm{R}}_{K-|s|}(s)$, i.e., d_j is super-selected. This completes the induction proof, and we have that all delimiter strings in $\bar{w}$ are super-selected by $\bar{P}$, and thus $\bar{P}$ gives us also a partition of the set $\mathcal{W}$.

It follows that there is a K-partition of $\mathcal{W}$ if and only if there is K-partition of $\bar{w}$. Finally, let us check that the reduction is polynomial. The length of each padded string or its mirror image is at most K. The length of d_1 is $K(K+3)/2 < K^2$. String $\operatorname{bin}(j)$ for $1 < j < \ell$ has length at most $\delta - 1$, and hence each d_j contains at most $2\delta = K$ padded strings and mirror images of padded strings. Hence, $|d_j| \leq K^2$. Thus, the total length of $\bar{w}$ is at most $\|\mathcal{W}\| + \ell K^2$. □

$\hat{x}_v^3$ 0 $\hat{x}_v^3$
$1 \leq v \leq n$

Figure 6: The set of forbidden strings, $\mathcal{F}$, used in the reduction from 3SAT(3) to FF-MSP.

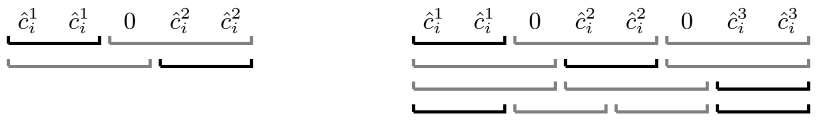

$\hat{c}_i^1$ $\hat{c}_i^1$ 0 $\hat{c}_i^2$ $\hat{c}_i^2$
$\hat{c}_i^1$ $\hat{c}_i^1$ 0 $\hat{c}_i^2$ $\hat{c}_i^2$ 0 $\hat{c}_i^3$ $\hat{c}_i^3$

Figure 7: The 2-literal clause string (left) and 3-literal clause string (right) used in the reduction from 3SAT(3) to FF-MSP. Shown below each string are all valid partitions. *Selected* literals for a partition are shown in black.

5. Factor-Free String Partition Problems

In this section, we will show **NP**-completeness of Factor-Free String Partition problems even when restricted to the constant size of the partition ($K = 3$), or to the binary alphabet ($L = 2$).

5.1. Factor-Free Multiple String Partition with Unbounded Alphabet

Let ϕ be an instance of 3SAT(3), with set $C = \{c_1, \ldots, c_m\}$ of clauses, and set $X = \{x_1, \ldots, x_n\}$ of variables. We shall define an alphabet Σ and construct a set of strings $\mathcal{W}$ in Σ^*, such that $\mathcal{W}$ has a factor-free 3-partition if and only if ϕ is satisfiable.

Let $|c_i|$ denote the number of literals contained in the clause c_i and let $c_i^1, \ldots, c_i^{|c_i|}$ be the literals of clause c_i. We define an alphabet Σ, which includes a letter for each literal occurrence in the clauses, three letters for each variable, and the letters 0 and 1, as follows:

$$\Sigma = \{\hat{x}_i^j;\ x_i \in X \wedge 1 \leq j \leq 3\} \cup \{\hat{c}_i^j;\ c_i \in C \wedge 1 \leq j \leq |c_i|\} \cup \{0, 1\}\,.$$

Note that $|\Sigma|$ is linear in the size of the 3SAT(3) problem ϕ (at most $3m+3n+2$).

We construct $\mathcal{W}$ to be the union of three sets of strings: clause strings ($\mathcal{C}$), enforcer strings ($\mathcal{E}$) and forbidden strings ($\mathcal{F}$) with the same function as in the equality-free case.

Construction of forbidden strings $\mathcal{F}$. To ensure that certain strings cannot be selected in $\mathcal{C}$ or $\mathcal{E}$, we construct a set of forbidden strings $\mathcal{F}$ as shown in Figure 6. Specifically, we forbid, for every variable v, any factor of the string $\hat{x}_v^3 0 \hat{x}_v^3$. The number of strings in $\mathcal{F}$ is n.

Construction of clause strings $\mathcal{C}$. For each clause $c_i \in C$, construct the *i-th clause string* to be $\hat{c}_i^1\hat{c}_i^1 0 \hat{c}_i^2\hat{c}_i^2$, if $|c_i| = 2$ and $\hat{c}_i^1\hat{c}_i^1 0 \hat{c}_i^2\hat{c}_i^2 0 \hat{c}_i^3\hat{c}_i^3$, if $|c_i| = 3$.

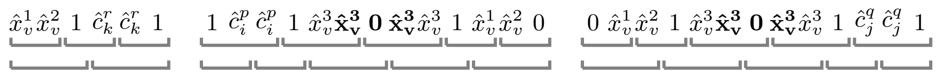

$\hat{x}_v^1\hat{x}_v^2\ 1\ \hat{c}_k^r\,\hat{c}_k^r\ 1$
$1\ \hat{c}_i^p\,\hat{c}_i^p\ 1\ \hat{x}_v^3\mathbf{\hat{x}_v^3}\ \mathbf{0}\ \mathbf{\hat{x}_v^3}\hat{x}_v^3\ 1\ \hat{x}_v^1\hat{x}_v^2\ 0$
$0\ \hat{x}_v^1\hat{x}_v^2\ 1\ \hat{x}_v^3\mathbf{\hat{x}_v^3}\ \mathbf{0}\ \mathbf{\hat{x}_v^3}\hat{x}_v^3\ 1\ \hat{c}_j^q\,\hat{c}_j^q\ 1$

Figure 8: The enforcer strings for the three literals of variable x_v used in the reduction from 3SAT(3) to FF-MSP. The two positive literals are denoted as $\hat{c}_i^p$ and $\hat{c}_j^q$ and the negative literal as $\hat{c}_k^r$. If the negative literal is selected in $\mathcal{C}$, then the enforcer string ensures neither positive literal can also be selected in $\mathcal{C}$ without creating a collision (top row). Likewise, if either or both of the positive literals are selected in $\mathcal{C}$, then the negative literal cannot be selected without creating a collision (bottom row). Forbidden factors are shown in bold.

Construction of enforcer strings $\mathcal{E}$. We must now ensure that no literal of ϕ that is selected in $\mathcal{C}$ is the negation of another selected literal. By definition of 3SAT(3), each variable appears exactly three times: twice positive and once negated. Let c_i^p and c_j^q be the two positive and c_k^r the negated occurrences of variable x_v. Then construct three *enforcer strings* for this variable as shown in Figure 8.

Observation 2. *Since every letter appears at least twice in $\mathcal{W} = \mathcal{C} \cup \mathcal{E} \cup \mathcal{F}$, no selected string can be a single letter.*

Lemma 3. *No string or factor of a string from the forbidden set $\mathcal{F}$ can be selected in $\mathcal{C}$ or $\mathcal{E}$.*

Proof. Consider any string $f \in \mathcal{F}$. If f is split into two or three selected strings, a single letter is selected, which is not possible. Regardless of the construction of $\mathcal{C}$ and $\mathcal{E}$, it follows that in any valid partition, since f is selected in $\mathcal{F}$, a factor of f cannot be selected in $\mathcal{C}$ nor in $\mathcal{E}$. $\square$

Lemma 4. *Given that no factor of a string from the forbidden set $\mathcal{F}$ is selected in $\mathcal{C} \cup \mathcal{E}$, at least one literal must be selected for each clause string in any factor-free 3-partition of $\mathcal{W}$.*

Proof. A literal letter cannot be selected alone without creating a collision. Therefore, we say a literal c_i^j is selected in clause c_i if and only if the string $\hat{c}_i^j\hat{c}_i^j$ is selected in the clause string for c_i. Whether c_i has two or three literals, a single letter 0 cannot be selected. A simple case analysis shows that in any valid partition at least one literal is selected (see Figure 7). $\square$

Lemma 5. *Given that no factor of a string from the forbidden set $\mathcal{F}$ is selected in $\mathcal{C} \cup \mathcal{E}$, any factor-free 3-partition of $\mathcal{W}$ must be consistent.*

Proof. Consider the three enforcer strings for some variable x_v with positive literals $c_i^p = c_j^q = x_v$, and the negated literal $c_k^r = \neg x_v$ shown in Figure 8. Note that the red strings in the middle of the last two enforcer strings are forbidden, and hence, no partition can have a cut point at the beginning or the end of the red string. Note also that the factor $x_v^3 x_v^3$ cannot be selected, as then $x_v^3 x_v^3 0$ or $0 x_v^3 x_v^3$ has to be selected, which is obviously not possible. Suppose the negative literal is selected in $\mathcal{C}$. Then the only partition which can be selected without creating a collision selects strings containing both $\hat{c}_i^p\hat{c}_i^p$ and $\hat{c}_j^q\hat{c}_j^q$ as factors, thus forbidding them from being selected in $\mathcal{C}$ (see Figure 8 (top partition)). Likewise, suppose one or both of the positive literals is selected in $\mathcal{C}$. Then only one partition of the first enforcer string is possible and it selects one string containing $\hat{c}_k^r\hat{c}_k^r$ as a factor, thus forbidding $\hat{c}_k^r\hat{c}_k^r$ from being selected in $\mathcal{C}$ (see Figure 8 (bottom partition)). Note that while these enforcer strings ensure literals selected in the clauses are consistent, it also ensures unwanted collisions do not occur since selected strings containing literals are prefixed or suffixed by a 1, a letter not used in the clause strings. Also note that the selected strings containing the variable letters do not collide. □

This completes the reduction. Notice that the reduction is polynomial as the combined length of the constructed set of strings $\mathcal{W} = \mathcal{C} \cup \mathcal{E} \cup \mathcal{F}$ is at most $8m + 32n + 3n = 8m + 35n$.

Theorem 5. *Factor-Free Multiple String Partition (FF-MSP) is* ***NP****-complete for partition size $K = 3$.*

Proof. It is easy to see that FF-MSP is in **NP**: a nondeterministic algorithm needs only guess a partition P where $|p_i| \leq K$ for all p_i in P and check in polynomial time that no string in P is a factor of another. Furthermore, it is clear that an arbitrary instance ϕ of 3SAT(3) can be reduced to an instance of FF-MSP, specified by a set of strings $\mathcal{W} = \mathcal{C} \cup \mathcal{E} \cup \mathcal{F}$, in polynomial time and space by the reduction detailed above.

Now suppose there is a satisfying truth assignment for ϕ. Simply select one corresponding true literal per clause in $\mathcal{C}$. The construction of clause strings guarantees that a 3-partition of the rest of each clause string is possible. Also, since a satisfying truth assignment for ϕ cannot assign truth values to opposite literals, then Lemma 5 guarantees that a valid partition of the enforcer strings are possible which does not conflict with the clause strings. Therefore, there exists a factor-free multiple string partition of $\mathcal{W}$.

Likewise, consider a factor-free multiple string partition of $\mathcal{W}$. Lemma 4 ensures that at least one literal per clause is selected. Furthermore, Lemma 5 guarantees that if there is no collision, then no two selected variables in the clauses are negations of each other. Therefore, this must correspond to a satisfying truth assignment for ϕ (if none of the three literals of a variable is selected in the partition of $\mathcal{C}$ then this variable can have arbitrary value in the truth assignment without affecting satisfiability of ϕ). □

5.2. Factor-Free String Partition with Unbounded Alphabet

We will use the following lemma to design delimiters for the reduction from the multiple string problem to the single string problem.

Lemma 6. *A valid K-partition P of a string w having $\alpha(x)^{3K-2}\beta$ as a factor, where α and β are single letters other than x, must select the strings $\alpha(x)^{K-1}$, $(x)^K$, and $(x)^{K-1}\beta$.*

Proof. Three strings are required to cover the factor $\delta = \alpha(x)^{3K-2}\beta$. However, more than three strings cannot be selected to cover δ, as otherwise at least two factors are selected consisting only of the letter x and must therefore collide. There is only one partition of δ covered by three factors which must select $\alpha(x)^{K-1}$, $(x)^K$, and $(x)^{K-1}\beta$. □

Theorem 6. *Factor-Free String Partition (FF-SP) is* ***NP****-complete for partition size* $K = 3$.

Proof. Consider an arbitrary instance I of FF-MSP having a set of strings $\mathcal{W} = \{w_1, w_2, \ldots, w_n\}$ over alphabet Σ, and maximum partition size K. We construct an instance I' of FF-SP as follows. Let $\widehat{\Sigma} = \{\alpha\} \cup \{\gamma_i, \text{ for } 1 \le i < n\}$, where $\widehat{\Sigma} \cap \Sigma = \emptyset$. Set the alphabet of I' to $\Sigma' = \Sigma \cup \widehat{\Sigma}$ and the maximum partition size to $K' = K$. Note that $|\Sigma'| = |\Sigma| + n$. Finally, construct the string $w' = w_1\alpha(\gamma_1)^{3K-2}\alpha w_2\alpha(\gamma_2)^{3K-2}\alpha \ldots \alpha(\gamma_{n-1})^{3K-2}\alpha w_n$. The reduction is in polynomial time and space as $|w'| = \sum_{w_i \in \mathcal{W}} |w_i| + 3K(n-1)$. By Lemma 6, the factors $\alpha(\gamma_i)^{3K-2}\alpha$ of string w' must be partitioned as $\alpha(\gamma_i)^{K-1}$, $(\gamma_i)^K$, $(\gamma_i)^{K-1}\alpha$, for $1 \le i < n$. Since any string containing a letter γ_i, $1 \le i < n$, cannot be a factor of any string in $\mathcal{W}$ it follows immediately that w' has a K'-partition if and only if $\mathcal{W}$ has a K-partition. □

5.3. Factor-Free Multiple String Partition with Binary Alphabet

In this section we are going to reduce the size of alphabet to 2. In order to do that we will map all letters of the original unbounded alphabet Σ except 0 and 1 to distinct binary strings of length t (t has to be large enough so that we have enough of strings), called codewords. Letters 0 and 1 will remain mapped to 0 and 1, respectively. Consequently, K will be set to $2t + 1$. We will use the same clause strings and a simplified version of the enforcer strings found in the unbounded case (just mapped to the binary alphabet). We will use only two forbidden strings, 000 and 010, to force valid K-partitions to cut the clause and enforcer strings just before or after the 0 or 1 letter. At the end, we will show that this does not introduce any new collisions in the K-partition corresponding to a truth assignment of the 3SAT(3) instance.

Construction of codewords. We will use the codewords of the following type $0(1)^i0(1)^{t-3-i}0$, where $i \in \{2, \ldots, t-5\}$. To make sure we have enough codewords for all literal and variable letters (at most $3m + 2n$), we have to choose $t \ge 3m + 2n + 6$.

Construction of forbidden strings. We will use only two forbidden strings $\mathcal{F} = \{000, 010\}$. Obviously, the only factor-free partition of $\mathcal{F}$ is without any nontrivial cut points. These two forbidden strings force any string containing uav as a factor, where u and v are codewords and a is a letter, to contain exactly one cut point around the letter a between u and v as formalized in the following lemma.

Lemma 7. *Any valid K-partitioning of $\alpha u a v \beta$ and $\mathcal{F}$, where u, v are codewords, $a \in \{0,1\}$ and α, β are arbitrary binary strings, contains either cut point $|\alpha| + |u|$ or $|\alpha| + |u| + 1$, but not both.*

Proof. Assume we have a valid partitioning of $\alpha u a v \beta$ without cut points at positions $|\alpha| + |u|$ and $|\alpha| + |u| + 1$. Since u ends with 0 and v starts with 0, there is a selected string containing $0a0$ as a factor, which is a forbidden string, a contradiction. $\square$

Construction of clause strings. We will use the same clause strings as for the unbounded alphabet:

$$\hat{c}_i^1 \hat{c}_i^1 0 \hat{c}_i^2 \hat{c}_i^2 \quad \text{or} \quad \hat{c}_i^1 \hat{c}_i^1 0 \hat{c}_i^2 \hat{c}_i^2 0 \hat{c}_i^3 \hat{c}_i^3 \,,$$

where $\hat{c}_i^j$ are distinct codewords described above. We say that a literal c_i^j is *selected* if $\hat{c}_i^j \hat{c}_i^j$ is super-selected in the K-partition.

Lemma 8. *In any factor-free K-partition of $\mathcal{C} \cup \mathcal{F}$, at least one of the literals in each clause is selected.*

Proof. By Lemma 7, there is exactly one cut point around each of the 0's between codewords in the clause string. This means that the K-partition of the clause string follows one of the patterns depicted in Figure 7 with the exception that any of the selected strings depicted in the figure can be further partitioned, *i.e.*, they are super-selected. The claim follows. $\square$

Construction of enforcer strings. We will use a slightly simplified version of the enforcer strings as those used for the unbounded alphabet:

$$\hat{x}_v^1 \hat{x}_v^2 1 \hat{c}_k^r \hat{c}_k^r 1, \quad 1 \hat{c}_i^p \hat{c}_i^p 1 \hat{x}_v^1 \hat{x}_v^2 0 \quad \text{and} \quad 0 \hat{x}_v^1 \hat{x}_v^2 1 \hat{c}_j^q \hat{c}_j^q 1 \,,$$

where $\hat{c}_i^p$ and $\hat{c}_j^q$ are codewords for the positive literals of a variable x_v, $\hat{c}_k^r$ is the codeword for the negated literal of x_v and $\hat{x}_v^1, \hat{x}_v^2$ are codewords for the variable letters of x_v. The difference from the unbounded case is that the factor $\hat{x}_v^3 \hat{x}_v^3 0 \hat{x}_v^3 \hat{x}_v^3 1$ can be safely removed from the second and third strings without changing the logic of the gadgets due to the property described in Lemma 7.

Observation 3. *In any factor-free K-partition no super-selected string can be a prefix (suffix) of any other super-selected string.*

Lemma 9. *In any factor-free K-partition of $\mathcal{C} \cup \mathcal{E} \cup \mathcal{F}$, the selected literals are consistent.*

Proof. By Lemma 7, either $\hat{x}_v^1 \hat{x}_v^2$ or $\hat{c}_k^r \hat{c}_k^r 1$ is super-selected in the first enforcer string. Note that if $\hat{x}_v^1 \hat{x}_v^2$ is super-selected then by Observation 3, $\hat{x}_v^1 \hat{x}_v^2 0$ and $0 \hat{x}_v^1 \hat{x}_v^2$ cannot be super-selected. Hence, if either $\hat{x}_v^1 \hat{x}_v^2 0$ or $0 \hat{x}_v^1 \hat{x}_v^2$ is super-selected then in the first enforcer string $\hat{c}_k^r \hat{c}_k^r 1$ is super-selected, and hence literal c_k^r cannot be selected.

By Lemma 7, either $1\hat{c}_i^p\hat{c}_i^p$ or $\hat{x}_v^1\hat{x}_v^2 0$ is super-selected in the second enforcer string. In the first case, by Observation 3, literal c_i^p cannot be selected; in the second case, by the above argument, literal c_k^r cannot be selected. Similarly, the last enforcer string ensures that literals c_j^q and c_k^r cannot be selected at the same time. $\square$

Theorem 7. *Factor-Free Multiple String Partition (FF-MSP) is* ***NP****-complete for the binary alphabet ($L = 2$).*

Proof. It follows by Lemmas 8 and 9 that if there is a factor-free K-partition of $\mathcal{W} = \mathcal{C} \cup \mathcal{E} \cup \mathcal{F}$, then the selected literals produce a satisfying assignment for the 3SAT(3) instance ϕ.

Now, assume that there is a satisfying assignment for ϕ. Select a literal in each clause which satisfies it and partition all clause and enforcer strings accordingly, ensuring literals selected in the clause strings are not selected in the enforcer strings. We will show that this K-partition P is factor-free. Obviously, the forbidden strings 000 and 010 are not factors of any selected string. In the clause strings, the K-partition P selects the following types of strings: $0aa$, $aa0$ and aa, where a is a codeword. In the enforcer string it selects the following types of strings: $1aa$, $aa1$, $0ab$, $ab0$, $ab1$, ab, $0a$, $a0$, $1a$, $a1$, where a, b are distinct codewords. It follows by the proof in the unbounded case that two strings where one is obviously a factor of the other, like aa and $0aa$, are not selected at the same time. Hence, it is enough to show that no new collisions are introduced by mapping the original letters to the binary alphabet.

Obviously, the strings of the same length are different as all codewords are different and the strings of different types (e.g., $0ab$ and $ab0$) would differ in the first two or last two letters. String ab (where we can have $a = b$) is not a factor of some σcd ($cd\sigma$), where $a \neq c$, $b \neq d$ are codewords and $\sigma \in \{0, 1\}$, as the 00 in the middle of ab would have to exactly match 00 in the middle of σcd ($cd\sigma$), which would imply $ab = cd$.

Finally, we will show that σa (and similarly, for $a\sigma$) is not a factor of cd and $\sigma' cd$ and $cd\sigma'$, where a, c, d are codewords and σ, σ' are letters, unless $\sigma = \sigma'$ and $a = c$. First, assume that $\sigma = 1$. Then σa contains two 101 factors with only 1's between them. String cd contains exactly two 101 factors but there is a 00 factor between their occurrences in cd, and the same is true for $\sigma' cd$ and $cd\sigma'$ if $\sigma' = 0$. Hence, assume that $\sigma' = 1 = \sigma$. Now, $cd\sigma'$ contains a factor $10(1)^+01$, but only at the very end, while in σa this factor is followed by at least two letters. Hence, the only possibility is that σa is a factor of σcd. However, pattern $10(1)^+01$ appears only at the beginning of σcd, and hence, σa would have to be a prefix of σcd and then $a = c$.

Second, assume that $\sigma = 0$. Then σa starts with 00. Similar case analysis would show that σa can only be a factor of σad, or of ca, $\sigma' ca$ or $ca\sigma'$. However, string $0a$ can be only selected from the second enforcer string $0xy1bb1$ and x never appears in the second position of any selected string of the type cd, $\sigma' cd$, $cd\sigma'$. $\square$

5.4. Factor-Free String Partition with Binary Alphabet

We will first design a sequence of strings which have to be selected no matter where they appear in the string we are partitioning.

Lemma 10. *Let $K \geq 1$ and for any $i \leq K$, let $d_i = (1)^i 0 (1)^{K-1-i}$. Then any factor-free K-partition of*

$$w = u_1 d_0 (1)^K d_0^{\mathrm{R}} u_2 d_1 d_1^{\mathrm{R}} \dots d_{N-2} d_{N-2}^{\mathrm{R}} u_N,$$

where $N \leq K/2$ and $u_1, \dots, u_N$ are arbitrary strings, selects the following strings $(1)^K, d_i, d_i^{\mathrm{R}}$, for every $i = 0, \dots, N-2$.

Proof. Let P be a factor-free multiple K-partition of w. We will show by induction on i that $(1)^K$, $d_0, \dots, d_i$ and $d_0^{\mathrm{R}}, \dots, d_i^{\mathrm{R}}$ are selected. The base case $i = 0$ follows by Lemma 6. For the inductive step, assume that $(1)^K$, $d_0, \dots, d_{i-1}$ and $d_0^{\mathrm{R}}, \dots, d_{i-1}^{\mathrm{R}}$ are selected, where $i \leq N-2$. We will show that d_i and d_i^{R} are also selected. The factor $d_i d_i^{\mathrm{R}}$ of w has length $2K$, hence, there is at least one cut point inside it. Let j be the first such cut point. Obviously, $j \leq K$. Assume that $j < K$. Let w_p be the selected string starting at cut point j. We will consider two cases:

Case 1. $j \geq i+1$. Then w_p is a prefix of $(1)^K$, a contradiction since $(1)^K$ is already selected.

Case 2. $j \leq i$. Then w_p is a prefix of d_{i-j}, a contradiction since d_{i-j} is already selected.

Hence, the first cut point inside the factor $d_i d_i^{\mathrm{R}}$ of w is at position K. By symmetrical argument, this is also the last such cut point. It follows that both d_i and d_i^{R} are selected. □

Corollary 1. *Let $K \geq 1$ and for any $i \leq K$, let $d_i = (1)^i 0 (1)^{K-1-i}$. Consider the string*

$$w = u_1 d_0 (1)^K d_0^{\mathrm{R}} u_2 d_1 d_1^{\mathrm{R}} \dots d_{N-2} d_{N-2}^{\mathrm{R}} u_N,$$

where $N \leq K/2$ and $u_1, \dots, u_N$ are arbitrary strings. If the string w has a factor-free K-partition then the sequence of strings $u_1, \dots, u_N$ has a factor-free multiple K-partition. On the other hand, if the sequence $u_1, \dots, u_N$ has a factor-free multiple K-partition such that each selected string contains at least two 0's then w has a factor-free K-partition.

Proof. The first implication follows immediately by Lemma 10. The second implication follows by the fact that each delimiter contains only one 0; hence, none of the selected strings in $u_1, \dots, u_N$ can be a factor of a delimiter. □

As the immediate consequence of Theorem 7 and Corollary 1, we have the following.

Theorem 8. *Factor-free String Partition (FF-SP) is **NP**-complete for the binary alphabet ($L = 2$).*

6. Prefix&Suffix-Free String Partition Problems

In this section, we will show **NP**-completeness of Prefix&Suffix-Free String Partition problems even when restricted to the constant size of the partition ($K = 3$), or to the binary alphabet ($L = 2$).

6.1. Prefix&Suffix-Free (Multiple) String Partition with Unbounded Alphabet

The constructions for factor-free string partition problems with unbounded alphabet can be applied to prefix&suffix-free partition problems with unbounded alphabet. Since every factor-free multiple string K-partition is also a prefix&suffix-free multiple string K-partition, it is enough to show that every prefix&suffix-free multiple string 3-partition of the constructed strings is "close" to a factor-free multiple string 3-partition. More precisely, we will show the following lemma.

Lemma 11. *Every prefix&suffix-free multiple string* 3*-partition of* $\mathcal{W} = \mathcal{C} \cup \mathcal{E} \cup \mathcal{F}$ *from Section 5.1 is a refinement of some factor-free multiple string* 3*-partition of* $\mathcal{W}$*.*

Proof. Consider a prefix&suffix-free multiple string 3-partition P of $\mathcal{W}$. First, we will show that for every letter $a \in \Sigma \setminus \{\hat{x}_i^2;\ x_i \in X\}$, a is not selected in P. Observe that for any letter $a \in \Sigma$ such that aa is a factor of one of the constructed strings, then in P, either one selected string starts with aa, or one selected string ends with aa, or one selected string ends with a and one starts with a. In either case, if a is selected in this or some other constructed string, it would be a prefix or a suffix of one of these strings. Hence, no such a can be selected. Next, we will show that letters in $\{0, 1\} \cup \{\hat{x}_i^1;\ x_i \in X\}$ cannot be selected either. Since 0 and 1 are the first and last letters in multiple constructed enforcer strings, neither of them can be selected. Consider a letter $\hat{x}_i^1$. It appears three times in the enforcer strings for variable x_i, and it appears as the first letter in the first enforcer string for x_i. If $\hat{x}_i^1$ is selected in the second or the third enforcer string then it would be a prefix of the first selected string of the first enforcer string, a contradiction. If $\hat{x}_i^1$ is selected in the first enforcer string, then in the second enforcer string it has to be selected together with 1 before and $\hat{x}_i^2$ after, which would imply that 0 is selected, a contradiction again. Finally, consider a letter $\hat{x}_i^2$. If it is selected in the first or the second enforcer string for x_i, then in the third enforcer string $\hat{x}_i^1\hat{x}_i^2 1$ is selected, and hence, also 0 would have to be selected, which is not possible. Hence, the only possibility is that $\hat{x}_i^2$ could be selected in the third enforcer string, which would imply that $0\hat{x}_i^1$ is selected in the same enforcer string in P, i.e., $0\hat{x}_i^1\hat{x}_i^2$ is super-selected in P.

Consider a 3-partition Q which contains the same selected strings as P with the exception that if strings $0\hat{x}_i^1$ and $\hat{x}_i^2$ are selected in the third enforcer string for x_i in P, then $0\hat{x}_i^1\hat{x}_i^2$ is selected in this string in Q. Hence, P is a refinement of Q. Note that all selected strings in Q have length 2 or 3. We will show that Q is factor-free. Assume that there are selected strings u and v in Q such that u

is a factor of v. Since u has length at least 2 and v length at most 3, u must be a prefix or a suffix of v. By the definition of Q, u and v must be super-selected in P. It follows that the first (last) selected string of u in P is a prefix (suffix) of the first (last) selected string of v in P, a contradiction. The claim follows.□

As a consequence of this lemma and the proofs in Section 5.1, we have the following result.

Theorem 9. *Prefix&Suffix-Free Multiple String Partition (PSF-MSP) is **NP**-complete for partition size $K = 3$.*

It is easy to see that Lemma 6 holds also for prefix&suffix-free partitions. Since the proof of Theorem 6 relies only on this lemma, we have the following result as well.

Theorem 10. *Prefix&Suffix-Free String Partition (PSF-SP) is **NP**-complete for partition size $K = 3$.*

6.2. Prefix&Suffix-Free Multiple String Partition with Binary Alphabet

As in Section 5.3, we will encode all letters of the original unbounded alphabet Σ, except 0 and 1, with distinct binary strings of length t, called codewords. Again, K will be set to $2t+1$. We will use the same clause strings and enforcer strings (although after mapping they will be different strings). However, we will need a substantially larger set of forbidden strings in order to force valid K-partitions to cut the clause and enforcer strings at the boundaries of codewords. We will also use a different set of codewords. At the end, we will show that this does not introduce any new collisions in the K-partition corresponding to a truth assignment of the 3SAT(3) instance.

Construction of codewords. We will use the codewords of the following type $001u100$, where u of length $t-5$ does not contain factor 00. The number of such codewords of length t is $\mathrm{Fib}(t-4)$, where $\mathrm{Fib}(i)$ denotes the i-th Fibonacci number: each codeword has to start with 001 and end with 100 and the middle part is any binary string without factor 00. The number of 00-free strings of length i, $f(i)$, can be recursively, computed as follows. A 00-free string of length i either start with 0 or 1. If it starts with 1, what follows is any 00-free string of length $i-1$. There are $f(i-1)$ of them. If it starts with 0, 1 must follow and then any other 00-free string of length $i-2$. Hence, $f(i) = f(i-1)+f(i-2)$. And since $f(0) = 1$ and $f(1) = 2$, we have $f(i) = \mathrm{Fib}(i+2)$. It is known that $\mathrm{Fib}(i) = \left\lfloor \frac{\varphi^i}{\sqrt{5}} + \frac{1}{2} \right\rfloor$, where $\varphi = (1+\sqrt{5})/2$. Hence, picking any $t \geq 4 + \log_\varphi \sqrt{5}(3m+2n-1/2)$ will guarantee that there are enough distinct strings with the above property to encode all variable and literal symbols. In the next subsection, we will also require that $t > \log_2(N) + 6$ for the construction of delimiter strings.

Construction of clause strings. We will use the same clause strings as for the unbounded alphabet:

$$\hat{c}_i^1\hat{c}_i^1 0\hat{c}_i^2\hat{c}_i^2 \quad \text{or} \quad \hat{c}_i^1\hat{c}_i^1 0\hat{c}_i^2\hat{c}_i^2 0\hat{c}_i^3\hat{c}_i^3\,,$$

where $\hat{c}_i^j$ are distinct codewords described above. We say that a literal c_i^j is *selected* if $\hat{c}_i^j\hat{c}_i^j$ is super-selected in the K-partition.

Construction of enforcer strings. We will use the same enforcer strings as for the unbounded alphabet:

$$\hat{x}_v^1\hat{x}_v^2 1\hat{c}_k^r\hat{c}_k^r 1, \quad 1\hat{c}_i^p\hat{c}_i^p 1\hat{x}_v^3\hat{x}_v^3 0\hat{x}_v^3\hat{x}_v^3 1\hat{x}_v^1\hat{x}_v^2 0 \quad \text{and} \quad 0\hat{x}_v^1\hat{x}_v^2 1\hat{x}_v^3\hat{x}_v^3 0\hat{x}_v^3\hat{x}_v^3 1\hat{c}_j^q\hat{c}_j^q 1\,,$$

where $\hat{c}_i^p$ and $\hat{c}_j^q$ are codewords for the positive literals of a variable x_v, $\hat{c}_k^r$ is the codeword for the negated literal of x_v and $\hat{x}_v^1, \hat{x}_v^2, \hat{x}_v^3$ are codewords for the variable letters of x_v.

Construction of forbidden strings. In order for the clause and enforcer strings to function correctly, we need to force cut points at the boundaries of codewords. To achieve that we will include a set of strings of length K for each clause and enforcer string. Given a string v of length $4t+1$, the sets of *proper long factors* of v, $\mathrm{PLF}(v)$ and $\mathrm{PLF}^*(v)$, are defined as $\{\mathrm{factor}_{i,K+i}(u);\ i = 1,\ldots,2t-1\}$ and $\{\mathrm{factor}_{i,K+i}(u);\ i = 1,\ldots,t-1,t+1,\ldots,2t-1\}$, respectively.

As in the factor-free case, we have the following simple observation.

Observation 4. *In any prefix&suffix-free K-partition, no super-selected string can be a prefix (suffix) of any other super-selected string.*

Lemma 12. *Assume that u, u', v, v' are codewords, $a \in \{0,1\}$ and α, β are arbitrary binary strings. Any valid K-partitioning of $\alpha uu'avv'\beta$ and $\mathrm{PLF}(uu'avv')$ selects either $uu'a$ or avv'. Any valid K-partitioning of $\alpha uu'avv'\beta$ and $\mathrm{PLF}^*(uu'avv')$ selects either $uu'a$, $u'a$, $u'av$, av or avv'.*

Proof. Assume we have a valid partitioning of $\alpha uu'avv'\beta$. By Observation 4, there are no cut points $|\alpha| + i$ for every $i = 1,\ldots,t-1,t+1,\ldots,2t-1,2t+2,\ldots,3t,3t+2,\ldots,4t$ in the second case (the set $\mathrm{PLF}^*(uu'avv')$) and also for $i = t$ and $i = 3t+1$ in the first case (the set $\mathrm{PLF}(uu'avv')$). The claim follows. □

We define the set of forbidden strings $\mathcal{F}$ as the union of the following sets:

- for each 2-clause c_i, $\mathrm{PLF}(\hat{c}_i^1\hat{c}_i^1 0\hat{c}_i^2\hat{c}_i^2)$,
- for each 3-clause c_i, $\mathrm{PLF}(\hat{c}_i^1\hat{c}_i^1 0\hat{c}_i^2\hat{c}_i^2) \cup \mathrm{PLF}(\hat{c}_i^2\hat{c}_i^2 0\hat{c}_i^3\hat{c}_i^3)$,
- for each variable x_v, $\mathrm{PLF}^*(\hat{x}_v^1\hat{x}_v^2 1\hat{c}_k^r\hat{c}_k^r) \cup \mathrm{PLF}(\hat{x}_v^3\hat{x}_v^3 0\hat{x}_v^3\hat{x}_v^3)$.

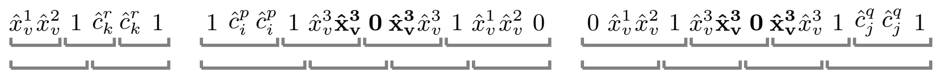

$\hat{x}_v^1\hat{x}_v^2\ 1\ \hat{c}_k^r\ \hat{c}_k^r\ 1$
$1\ \hat{c}_i^p\ \hat{c}_i^p\ 1\ \hat{x}_v^3\mathbf{\hat{x}_v^3}\ \mathbf{0}\ \mathbf{\hat{x}_v^3}\hat{x}_v^3\ 1\ \hat{x}_v^1\hat{x}_v^2\ 0$
$0\ \hat{x}_v^1\hat{x}_v^2\ 1\ \hat{x}_v^3\mathbf{\hat{x}_v^3}\ \mathbf{0}\ \mathbf{\hat{x}_v^3}\hat{x}_v^3\ 1\ \hat{c}_j^q\ \hat{c}_j^q\ 1$

Figure 9: The two cases of super-selected strings of the enforcer strings for the three literals of variable x_v used in the reduction from 3SAT(3) to PSF-MSP with binary alphabet. Forbidden factors are shown in bold.

It follows by Lemma 12 that any valid K-partition of $\mathcal{C} \cup \mathcal{E} \cup \mathcal{F}$ super-selects one of the partitions of clause strings depicted in Figure 7 and one of the partitions of enforcer strings depicted in Figure 9. Note that, for example, in the second partition of the second enforcer string, $1\hat{c}_i^p\hat{c}_i^p1$ is super-selected and the construction does not force the valid partition to further split this string to $1\hat{c}_i^p$ and $\hat{c}_i^p1$, as in the case with unbounded alphabet. However, we still have that if $\hat{c}_i^p\hat{c}_i^p$ is super-selected in the clause strings, then we have to choose the second partition type of the enforcer strings, and that if $\hat{c}_i^p\hat{c}_i^p$ is not super-selected in the clause strings, then there is a way to partition the enforcer strings by using the first partition type without introducing collisions. As a consequence, we have the following lemma and theorem.

Lemma 13. *In any prefix&suffix-free K-partition of $\mathcal{C} \cup \mathcal{E} \cup \mathcal{F}$, the selected literals are consistent.*

Theorem 11. *Prefix&Suffix-Free Multiple String Partition (PSF-MSP) is* ***NP****-complete for the binary alphabet ($L = 2$).*

Proof. Note that the construction is polynomial as the combined length of the strings in $\mathcal{C} \cup \mathcal{E} \cup \mathcal{F}$ is $O(K^2(n+m))$, where $K = O(\log(m+n))$. It follows by Lemma 13 that if there is a prefix&suffix-free K-partition of $\mathcal{W} = \mathcal{C}\cup\mathcal{E}\cup\mathcal{F}$, then the selected literals produce a satisfying assignment for the 3SAT(3) instance ϕ.

Now, assume that there is a satisfying assignment for ϕ. Select a literal in each clause which satisfies it and partition all clause and enforcer strings accordingly, ensuring literals selected in the clause strings are not selected in the enforcer strings. We will show that this K-partition P is prefix&suffix-free.

The K-partition selects the strings of the following types:

(1) length $K = 2t + 1$: $\text{factor}_{i,K+i}(ab\sigma cd)$, $i = 0, \dots, K-1$,
(2) length $K - 1 = 2t$: ab,
(3) length $t + 1$: $a\sigma$ and σa,

where $\sigma \in \{0, 1\}$ and a, b, c, d are codewords, i.e., binary strings of length t starting and ending with 00 and containing no other factors 00. It is enough to show that these strings are prefix&suffix-free. Note that although ab is a prefix of $ab\sigma$ and a suffix of σab, and $b\sigma$ (σa) is a suffix (prefix) of $ab\sigma$ (σab), they are never selected together with any of those when partition is based on any truth assignment to the 3SAT(3) instance.

First, we show that the strings of type (1) are not equal. For any $i = 2, \dots, K-3$, string $\text{factor}_{i,K+i}(ab\sigma cd)$ contains a unique factor $00\sigma00$ starting at position $K-3-i$. On the other hand, if $i = 0, 1, K-2, K-1$, $\text{factor}_{i,K+i}(ab\sigma cd)$ does not contain $00\sigma00$ as a factor, but contains a unique factor 0000 starting at positions $t-1, t-2, t+2, t+1$, respectively. Hence, if $\text{factor}_{i,K+i}(ab\sigma cd) = \text{factor}_{j,K+j}(a'b'\sigma'cd)$, then $i = j$. It follows that $\sigma = \sigma'$ and either $b = b'$ or $c = c'$. If $i = 0$ or $i = K-1$ (strings are selected in $\mathcal{C} \cup \mathcal{E}$), then we also have $a = a'$ or $d = d'$, respectively, i.e., the strings are the same before mapping to binary alphabet. Assume that $i \neq 0, K-1$, i.e., the string is from $\mathcal{F}$. Since σ and the first codeword before (after) σ uniquely determines the string on which PLF was applied, we must have $ab\sigma cd = a'b'\sigma'c'd'$.

Second, we show that the strings of type (2), ab, are not prefixes or suffixes of strings of type (1), except for $0ab$, $ab0$, $1ab$ or $ab1$. Assume that ab is a prefix of $u = \text{factor}_{i,K+i}(cd\sigma ef)$. Since ab starts with 001, we must have $i = t$ or $i = K-2$ and $\sigma = 1$. If $i = t$, then u contains a substring $00\sigma00$, which is not a substring of ab, a contradiction. If $i = K-2$ and $\sigma = 1$, then u starts with 00100, hence ab cannot be the prefix of u. A similar argument applies if ab is a suffix of u.

Finally, we show that the strings of type (3), σa and $a\sigma$, are not prefixes or suffixes of strings of type (1), (2) or (3), except for σab and $ba\sigma$. First, assume that σa is a prefix of some other selected string $u \neq \sigma ab$. Since σa starts with $\sigma 001$, we must have $u = \text{factor}_{i,K+i}(cd\sigma' ef)$ and either $i = t-1$ and $\sigma = 0$ or $i = K-4$ and $\sigma = \sigma' = 1$. In the first case, we have $a = \hat{x}_v^1$ and $d = a = \hat{x}_v^1$, which is not possible. In the second case, u starts with 100100 which is not a prefix of σa, a contradiction. Second, assume that $a\sigma$ is a prefix of some other selected string $u \neq a\sigma$. Then either (a) $u = ab$ or $u = ab\sigma'$, or $u = \text{factor}_{i,K+i}(cd\sigma' ef)$ with (b) $i = t$, $d = a$ and $\sigma = \sigma'$, or (c) $i = K-3$ and $\sigma' = 1$. In case (a), we have $\sigma = 0$, hence $a = \hat{x}_v^2$ for some v. Since no selected word with length $2t$ or $2t+1$ starts with $\hat{x}_v^2$, this case cannot happen. Consider case (b). If $\sigma = 0$, then $d = a = \hat{x}_v^2$, and hence, $\sigma' = 1$, a contradiction. If $\sigma = 1$, then either $d = a$ is a literal codeword or $d = a = \hat{x}_v^2$. In the first case, we have $\sigma' = 0$, a contradiction. In the second case, $u \in \text{PLF}^*(\hat{x}_v^1 \hat{x}_v^2 1 \hat{c}_k^r \hat{c}_k^r)$, hence $i \neq t$, a contradiction. Finally, in case (c), u starts with 00100, hence $a\sigma$ cannot be a prefix of u. The cases when $a\sigma$ or σa is a suffix of another selected string can be shown to be contradictory by using similar arguments. □

6.3. Prefix&Suffix-Free String Partition with Binary Alphabet

We will first design a sequence of strings which have to be selected no matter where they appear in the string we are partitioning. For integers $n, d \geq 0$, $b \geq 2$, where $d \geq \lfloor \log_b n \rfloor$, let $[n]_b^d$ denote the b-ary representation of n using d digits. For example, $[3]_2^4 = 0011$ and $[0]_2^0 = \varepsilon$.

Lemma 14. *Consider $K \geq 1$. For any $0 \leq i < 2^K$, let $d_i = [i]_2^K$, i.e., $d_0 = (0)^K$, $d_1 = (0)^{K-1}1$, $d_2 = (0)^{K-2}10$, $d_4 = (0)^{K-2}11$, etc. Then any prefix&suffix-free K-partition of*

$$w = d_0^{\mathrm{R}} d_1 u_1 d_1^{\mathrm{R}} d_2 u_2 \dots d_{N-1}^{\mathrm{R}} d_N u_N,$$

where $N \leq 2^{K/2} - 1$ and $u_1, \dots, u_N$ are arbitrary binary strings, selects the following strings $d_0^{\mathrm{R}}, \dots, d_{N-1}^{\mathrm{R}}$ and $d_1, \dots, d_N$.

Proof. First, let us observe some properties of strings d_i, where $i \leq N$. Since $N \leq 2^{K/2} - 1$, d_i starts with at least $K/2$ 0s. In addition, for any d_i and $a \leq K$, there exists a $j < i$ such that $(0)^a \operatorname{prefix}_{K-a}(d_i) = d_j$. We will use these properties in the proof.

Let $w = w_1 \dots w_m$ be a prefix&suffix-free K-partition of w. We will show by induction on i that $d_0^{\mathrm{R}}, \dots, d_i^{\mathrm{R}}$ and $d_1, \dots, d_{i+1}$ are selected. For the base case $i = 0$, consider the first two selected strings w_1 and w_2. Note that $\operatorname{prefix}_{2K-1}(w) = (0)^{2K-1}$. If $|w_1| < K$ or $|w_2| < K$, then both $w_1, w_2 \in 0^+$, which is a contradiction, since one is a prefix of the other. Hence, $w_1 = d_0^{\mathrm{R}}$ and $w_2 = d_1$, i.e., d_0^{R} and d_1 are both selected as required.

For the inductive step, assume that $d_0^{\mathrm{R}}, \dots, d_i^{\mathrm{R}}$ and $d_1, \dots, d_{i+1}$ are selected. We will show that d_{i+1}^{R} and d_{i+2} are also selected. The factor $d_{i+1}^{\mathrm{R}} d_{i+2}$ of w has length $2K$, hence, there is at least one cut point inside it. Let j be the first such cut point. Obviously, $j \leq K$. Assume that $j < K$. Let w_p be the selected string starting at cut point j. We will consider two cases:

Case 1. w_p starts with $(0)^{K/2}$. Then w_p is a prefix of a string v of length K starting at cut point j. By the above properties, $v = d_k$, for some $k < i + 2$. By the induction hypothesis, d_k is selected, a contradiction.

Case 2. w_p contains at least one 1 in the first $\min(K/2, |w_p|)$ letters. By the above properties, $j < K/2$, and hence, w_p is a prefix of string $\operatorname{suffix}_{K-j}(d_{i+1}^{\mathrm{R}})(0)^j = d_k^{\mathrm{R}}$ for some $k < i+1$. This is a contradiction, since by the induction hypothesis, d_k^{R} is selected.

Hence, we can assume that the first cut point inside the factor $d_{i+1}^{\mathrm{R}} d_{i+2}$ of w is at position K. By symmetrical argument, this is also the last such cut point. It follows that both d_{i+1}^{R} and d_{i+2} are selected.

Corollary 2. *Consider integer $p \geq 1$ and $K \geq 2(p+1)$. For any $0 \leq i < 2^K$, let $d_i = [i]_2^K$. Consider the string*

$$w = d_0^{\mathrm{R}} d_1 u_1 d_1^{\mathrm{R}} d_2 u_2 \dots d_{N-1}^{\mathrm{R}} d_N u_N,$$

where $N \leq 2^{\lfloor K/2 \rfloor - p} - 1$ and $u_1, \dots, u_N$ are arbitrary strings. If the string w has a prefix&suffix-free K-partition then the sequence of strings $u_1, \dots, u_N$ has a prefix&suffix-free K-partition. On the other hand, if the sequence $u_1, \dots, u_N$ has a prefix&suffix-free K-partition such that each selected string has length at least $\lfloor K/2 \rfloor$ and does not contain a substring $(0)^p$ then w has a prefix&suffix-free K-partition.

Proof. The first implication follows immediately by Lemma 14. For the second implication it is enough to show that no selected string s of some u_i is a prefix or suffix of some d_j (d_j^{R}), $j \leq N$. Note that since $N \leq 2^{\lfloor K/2 \rfloor - p} - 1$, every d_j (d_j^{R}) contains a substring $(0)^{\lceil K/2 \rceil + p}$. Since the length of u_i is at least $\lfloor K/2 \rfloor$, it must contain at least $\lceil K/2 \rceil + p + \lfloor K/2 \rfloor - K = p$ consecutive zeros, a contradiction.

Since no selected string in the construction from Theorem 11 contains a substring $(0)^6$, if we choose $t > \log_2(N) + 6$, we have the following consequence of Theorem 11 and Corollary 2.

Theorem 12. *Prefix&Suffix-free String Partition (PSF-SP) is* ***NP****-complete for the binary alphabet ($L = 2$).*

Note that this construction is polynomial as the length of the constructed string is $O(K|\mathcal{W}| + \|\mathcal{W}\|)$, where $\mathcal{W}$ is the instance for multiple string version of the problem.

7. Prefix/Suffix-Free String Partition Problems

In this section, we will show **NP**-completeness of Prefix&Suffix-Free String Partition problems even when restricted to the constant size of the partition ($K = 2$), or to the binary alphabet ($L = 2$). All proofs presented in this section are for the Prefix-Free String Partition problems. The results for the Suffix-Free String Partition problems follow by symmetry.

7.1. Prefix/Suffix-Free Multiple String Partition with Unbounded Alphabet

Similar to the equality-free and factor-free cases, we will show a polynomial reduction from an arbitrary instance of 3SAT(3).

Let ϕ be an instance of 3SAT(3), with set $C = \{c_1, \ldots, c_m\}$ of clauses, and set $X = \{x_1, \ldots, x_n\}$ of variables. We shall define an alphabet Σ and construct a set of strings $\mathcal{W}$ in Σ^*, such that $\mathcal{W}$ has a prefix-free 2-partition if and only if ϕ is satisfiable.

Let $|c_i|$ denote the number of literals contained in the clause c_i and let $c_i^1, \ldots, c_i^{|c_i|}$ be the literals of clause c_i. We construct an alphabet Σ, which includes four letters for each variable, a letter for each literal occurrence in the clauses and the letter \$ as folows:

$$\Sigma = \{\hat{x}_i^j;\ x_i \in X \wedge 1 \leq j \leq 4\} \cup \{\hat{c}_i^j;\ c_i \in C \wedge 1 \leq j \leq |c_i|\} \cup \{\$\}\,.$$

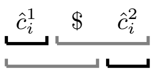
Note that $|\Sigma|$ is linear in the size of the 3SAT(3) problem ϕ (at most $4n+3m+1$).

We construct $\mathcal{W}$ to be the union of three sets of strings: clause strings ($\mathcal{C}$), enforcer strings ($\mathcal{E}$) and forbidden strings ($\mathcal{F}$) with the same function as in the equality-free and factor-free cases.

Construction of forbidden strings. The forbidden set, $\mathcal{F}$, consists of the single string \$\$. Without loss of generality, we refer to this as the *forbidden string.*

Lemma 15. *No factor of the forbidden string can be selected in $\mathcal{C}$ nor in $\mathcal{E}$.*

Proof. No proper factor of the forbidden string can be selected without creating a collision. Therefore, the entire string must be selected. Regardless of the construction of $\mathcal{C}$ and $\mathcal{E}$, it follows that in any valid partition, since \$\$ is selected in $\mathcal{F}$, a factor of it cannot be selected in $\mathcal{C}$ nor in $\mathcal{E}$. □



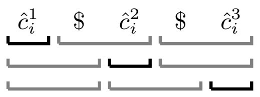
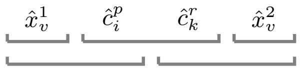


Figure 10: The 2-literal clause gadget (left) and 3-literal clause gadget (right) used in the reduction from 3SAT(3) to PF-MSP. Shown below each gadget are all valid partitions. *Selected* literals of a partition are shown in black.



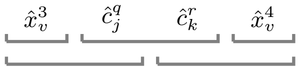


Figure 11: The pair of enforcer strings for a variable x_v used in the reduction from 3SAT(3) to PF-MSP. The two positive literals for variable x_v are denoted as $\hat{c}_i^p$ and $\hat{c}_j^q$ and the negative literal as $\hat{c}_k^r$.

Construction of clause strings. For each clause $c_i \in C$, construct the *i-th clause string* to be $\hat{c}_i^1\$\hat{c}_i^2$, if $|c_i| = 2$ and $\hat{c}_i^1\$\hat{c}_i^2\$\hat{c}_i^3$, if $|c_i| = 3$.

Lemma 16. *Given that no factor of the forbidden string is selected in $\mathcal{C} \cup \mathcal{E}$, exactly one literal must be selected for each clause string in any prefix-free 2-partition of $\mathcal{W}$.*

Proof. We say a literal c_i^j is selected in clause c_i if and only if the string $\hat{c}_i^j$ is selected in the clause string for c_i. Whether c_i has two or three literals, the forbidden string \$ cannot be selected alone. A simple case analysis shows that in any valid partition exactly one literal is selected (see Figure 10). $\square$

Construction of enforcer strings. We must now ensure that no literal of ϕ that is selected in $\mathcal{C}$ is the negation of another selected literal. By definition of 3SAT(3), each variable appears exactly three times: twice positive and once negated. Let c_i^p and c_j^q be the two positive and c_k^r the negated occurrences of a variable x_v. Then construct two *enforcer strings* for this variable as shown in Figure 11.

Lemma 17. *Given that no factor of the forbidden string is selected in $\mathcal{C} \cup \mathcal{E}$, any prefix-free 2-partition of $\mathcal{W}$ must be consistent.*

Proof. Consider the two enforcer strings for variable x_v with positive literals $c_i^p = c_j^q = x_v$, and the negated literal $c_k^r = \neg x_v$ shown in Figure 11.

Suppose the negative literal is selected in $\mathcal{C}$. Then the only partition which can be selected without creating a collision selects strings containing both $\hat{c}_i^p$ and $\hat{c}_j^q$ as a prefix, thus forbidding them from being selected in $\mathcal{C}$ (see Figure 11 (top row)).

Likewise, if one or both of the positive literals is selected in $\mathcal{C}$ then in any collision-free 2-partition a string is selected containing $\hat{c}_k^r$ as a prefix, thus forbidding the negative literal from being selected in $\mathcal{C}$ (see Figure 11 (bottom row)). $\square$

This completes the reduction. Notice that the reduction is linear as the combined length of the constructed set of strings $\mathcal{W} = \mathcal{C} \cup \mathcal{E} \cup \mathcal{F}$ is at most $5m + 8n + 2$.

Theorem 13. *Prefix(Suffix)-Free Multiple String Partition (PF-MSP) is* ***NP****-complete for partition size* $K = 2$*.*

Proof. It is easy to see that PF-MSP is in **NP**: a nondeterministic algorithm needs only guess a partition P where $|p_i| \leq K$ for all p_i in P and check in polynomial time that no string in P is a prefix of another. Furthermore, it is clear that an arbitrary instance ϕ of 3SAT(3) can be reduced to an instance of PF-MSP, specified by a set of strings $\mathcal{W} = \mathcal{C} \cup \mathcal{E} \cup \mathcal{F}$, in polynomial time and space by the reduction detailed above.

Now suppose there is a satisfying truth assignment for ϕ. Simply select one corresponding true literal per clause in $\mathcal{C}$. The construction of clause strings guarantees that a 2-partition of the rest of each clause string is possible. Also, since a satisfying truth assignment for ϕ cannot assign truth values to opposite literals, then Lemma 17 guarantees that a valid partition of the enforcer strings are possible. Therefore, there exists a prefix-free multiple string partition of $\mathcal{W}$.

Likewise, consider a prefix-free multiple string partition of $\mathcal{W}$. Lemma 16 ensures that exactly one literal per clause is selected. Furthermore, Lemma 17 guarantees that if there is no collision, then no two selected variables in the clauses are negations of each other. Therefore, this must correspond to a satisfying truth assignment for ϕ (if none of the three literals of a variable is selected in the partition of $\mathcal{C}$ then this variable can have an arbitrary value in the truth assignment without affecting satisfiability of ϕ). $\square$

7.2. Prefix/Suffix-Free String Partition with Unbounded Alphabet

To show the single string restriction of this problem is **NP**-complete, we design the same delimiter strings as specified in the factor-free construction. The result follows immediately by Theorem 13 and Lemma 6.

Theorem 14. *Prefix(Suffix)-Free String Partition (PF-SP) is* ***NP****-complete for partition size* $K = 2$*.*

7.3. Prefix/Suffix-Free Multiple String Partition with Binary Alphabet

In this section we start with the same construction as the multiple string unbounded alphabet case to form a set of strings $\mathcal{W}$, but show how the letters can be encoded into binary. We map the \$ letter to 1 and map all other letters of the original unbounded alphabet Σ to distinct binary strings of length t, called codewords. Consequently, K will be set to $2t$. We will establish that no codeword can properly contain a cut point. Furthermore, by design, no codeword is a prefix of another. Since the mapping to binary does not introduce new collisions, and since codewords cannot be cut in the middle, the correctness of the construction will follow from the results on the unbounded case.

Construction of codewords. We use codewords of the form $00(1)^i0(1)^{t-4-i}0$, where $i \in \{2, \ldots, t-6\}$. To ensure we have enough codewords for all literal and variable letters (at most $3m+4n$), we have to choose $t \geq 3m+4n+7$.

Construction of forbidden string. We will use the following set of forbidden strings: {11, 01, 101, 0001, 10001}. Considering only the forbidden set, a simple case analysis shows that each forbidden string must be entirely selected, otherwise a collision occurs. This set of forbidden strings ensures that no codeword can be cut in the middle.

Lemma 18. *Given that no strings selected in $\mathcal{C} \cup \mathcal{E}$ have a forbidden prefix, any prefix-free K-partition of $\mathcal{W}$ must not contain a cut point within a codeword.*

Proof. Recall that codewords are of length t and that all length two binary strings are prefixes of a forbidden string and therefore cannot be selected. Let us consider any cut point beginning within an arbitrary codeword w. For any proper suffix of w longer than two, it contains a prefix in the set $\{011, 111, 101, 110\}$. Each of these contains a forbidden string as a prefix and therefore a cut point in w cannot begin prior to positions $2, 3, \ldots, t-2$. We must now show that a cut point cannot begin prior to position $t-1$ or prior to position t. Recall that by construction, w is followed by either another codeword, the letter 1, or the empty string. If the empty string, it is not possible to a have a cut point in the position prior to $t-1$ or t since a string will be selected that has length less than 3 and will therefore be a prefix of a forbidden string. If w is followed by the letter 1 a cut point prior to $t-1$ or t will have a prefix in the set $\{101, 01\}$, both of which are forbidden strings. Finally consider the case that w is followed by another codeword and recall that all codewords begin with 001. If a cut point occurs prior to position $t-1$, and the selected string beginning at that position has length at least five, then it will contain 10001 as a prefix which is a forbidden string; any shorter selection will be a prefix of the forbidden string 10001. If a cut point occurs prior to position t and the selected string has length at least four, it will contain 0001 as a prefix which is a forbidden string; any shorter selection will be a prefix of the forbidden string 0001. $\square$

The above lemma ensures no codeword is divisible. The result is that the binary encoded instance I' of an unbounded alphabet instance I can be partitioned exactly in the same relative positions as the original instance. Since each codeword cannot be a prefix of another by design, then correctness of the binary case immediately follows.

Theorem 15. *Prefix(Suffix)-Free Multiple String Partition (PF-MSP) is* ***NP****-complete for binary alphabet ($L = 2$).*

7.4. Prefix/Suffix-Free String Partition with Binary Alphabet

Similar to the factor-free case, we will design delimiters to join the set $\mathcal{W}$ of the multiple string case, into one string, without changing the possibilities for partitioning the original set of strings and without introducing new types of collisions. Specifically, we will create a new string instance $I = WF$ where W is a string that concatenates all strings in $\mathcal{W}$, expect from the forbidden set $\mathcal{F}$, using delimiters we describe below. The string F has a special construction to ensure the strings from the forbidden set must be selected in F for any collision-free partition of I. Thus, the new instance I will have a collision-free partition if and only if $\mathcal{W}$ does.

Construction of delimiters. We design delimiters similar to the codewords of Section 7.3 that instead have length K. Specifically, they are of the form $00(1)^i0(1)^{K-4-i}0$, where $i \in \{2, \ldots, K-6\}$.

Lemma 19. *Given that no strings selected in W have a forbidden prefix, any prefix-free K-partition of W must select all delimiters. Furthermore, the delimiters do not collide with any valid selection of the original strings from the set $\mathcal{W}$.*

Proof. By Lemma 18 the delimiters, which are simply longer codewords, cannot contain a cut point in a middle position. Since they are of the maximum string length, K, they must be entirely selected in W. As a consequence of Lemma 18, in any valid partition of the set $\mathcal{W}$, the selected strings are either: (i) single codewords, (ii) a codeword prefixed by a 1, (iii) a codeword suffixed by a 1, or (iv) two adjacent codewords. Since each case is no longer than a delimiter, it is sufficient to show that none could be a prefix of a delimiter. In all four cases, none could be a prefix as the selected string would contain at least four 0s in the first $K/2+1$ positions, whereas a delimiter contains at most three. $\square$

Construction of forbidden string. We now construct the string $F = F_4F_3F_2F_1$. The string is constructed in a meticulous manner to ensure that each sub-part must select forbidden strings from a different partition of the forbidden set $\mathcal{F}$. In particular, $F_1 = 1(0)^{3K-2}111$ and it forces 11 to be selected; $F_2 = 001(0)^{K-3}(0)^{K-2}1110100(1)^{K-2}01$ and it forces 101 and 01 to be selected; $F_3 = 00(1)^{K-3}010001$ and it forces 10001 to be selected; and $F_4 = 00(1)^{k-4}010001$ and it forces 0001 to be selected.

Lemma 20. *In any prefix-free partition of F the set of forbidden strings must be selected. Furthermore, there exists a prefix-free partition of F that does not collide with a valid partition of W assuming that W does not contain a forbidden prefix.*

Proof. At a high level, each sub-part of F is constructed to ensure that one or more forbidden strings are selected, and the remainder of the sub-part consists of K length sub-strings that must be entirely selected. Furthermore, the construction of F ensures there is always a cut point between sub-parts and sub-part F_j is constructed with the knowledge of strings forbidden in F_i, for all $i < j$.

Consider sub-part F_1. It contains the $3K$-length substring $\alpha = 1(0)^{3K-2}1$. At least three strings must be selected to cover α. However, it cannot be more than three as otherwise at least two contain only 0 letters and therefore one must be a prefix of another. The only cover for α consisting of three strings must select $1(0)^{K-1}$, $(0)^K$, and $(0)^{K-1}1$, regardless of the sub-strings preceding or succeeding α. Since 1 cannot be selected alone (without being a prefix of $1(0)^{K-1}$) the string 11 must be selected. As a consequence, for any substring $(1)^i$ of F ("run of ones"), there cannot be a cut point before any of the first $i-1$ letters of this substring.

Second, consider sub-part F_2. Position 2 of F_2 is not a cut point, otherwise the selected string starting at this cut point would be a prefix of $1(0)^{K-1}$. Similarly, $3, \dots, K-1$ are not cut points either. It follows that either K or $K+1$ is a cut point. We will show that the second is not possible. Assume there is a cut point at position $K+1$. Then the selected string ending at this position is $01(0)^{K-2}$. At the end of F_2, by the "run of ones" argument, positions $2K+5, \dots, 3K+1$ are not cut points. Clearly, $3K+4$ is not a cut point either. And since $01(0)^{K-2}$ is selected, $3K+3$ is not a cut point as well. It follows that there is a cut point $3K+2$, and hence, 101 is selected. Consider the selected string starting at cut point $K+1$. Since it cannot consist of zeros only, it cannot end at positions $K+2, \dots, 2K-2$. By the "runs of ones" argument, it cannot end at position $2K-1$, and since 101 is selected at the end of F_2, neither at position $2K$. Therefore, it has to end at position $2K+1$, i.e., it is $(0)^{K-3}111$. Consider the selected string starting at cut point $2K+1$. It cannot end at positions $2K+2, 2K+3, 2K+4$, otherwise it would be a prefix of selected string $01(0)^{K-2}$. However, since positions $2K+5, \dots, 3K+1$ are not cut points (as argued above), it would have to be longer than K, a contradiction. It follows that K is a cut point. Using the same arguments as above, the selected string at position K has to end at position $2K$, i.e., it is $(0)^{K-2}11$. The selected string at position $2K$ starts with 101, hence, position $3K+2$ cannot be a cut point. It follows that $3k+3$ is a cut point and 01 is selected. That implies that the selected string ending at cut point $3K+3$ must start at $2K+3$, i.e., this selected string is $00(1)^{K-2}$, and similarly, the selected string ending at K must start at position 0 of F_2, i.e., $001(0)^{K-3}$ is selected. It follows that 101, between positions $2K$ and $2K+3$, is selected as well. Hence, F_2 in F is partitioned as $001(0)^{K-3}$, $(0)^{K-2}11$, 101, $00(1)^{K-2}$, 01, thus forbidding 101 and 01.

For F_3, since 101 and 11 are already forbidden, there cannot be a cut point prior to any 1 within the left-most run of 1s. Since 01 is forbidden, there cannot be a cut point in the second position, nor immediately after the left-most run of 1s. It follows that $00(1)^{K-3}$ must be entirely selected, regardless of the string preceding F_3. The remaining string 10001 must be entirely selected, otherwise it would conflict with 101 or 01. Similarly, the partitioning of F_4 ensures that $00(1)^{k-4}01$ and 0001 must be selected.

Note that all strings selected in F that are not in the forbidden set have length K. Thus, to show no new collisions have been introduced, it is sufficient to show that no string selected in a valid partition of W, that does not contain a forbidden prefix, cannot be a prefix of one of these strings. As noted earlier, every selected string in a valid partition of W contains at least one codeword as a sub-string. Each codeword contains three runs of 0s; however, each K-length selected string in F contains no more than two runs of 0s. □

By our straightforward polynomial time and space reduction of a binary PF-MSP instance into a binary PF-SP instance and by Lemmas 19 and 20 and Theorem 15 we have the following result.

Theorem 16. *Prefix(Suffix)-Free String Partition (PF-SP) is* ***NP****-complete for binary alphabet ($L = 2$).*

8. On Dichotomy

In all four cases, equality-free, factor-free, prefix&suffix-free and prefix/suffix-free, we have shown that the string partition problem is **NP**-complete for the binary alphabet. Obviously, it remains **NP**-complete for any alphabet of size greater or equal to two. On the other hand, the string partition problems become easy for the unary alphabet; a partition exists if and only if the string being partitioned is shorter or equal to a certain threshold: $1+2+\cdots+K = K(K+1)/2$ (in the equality-free case) and K (in all other cases). The first threshold follows from the fact that n can be written as the sum of distinct integers between 1 and K if and only if $n \leq K(K+1)/2$. The second threshold follows because only one string can be selected without introducing a conflict. For the multiple string partition problem with unary alphabet, in all cases other than the equality-free one, the problem is still trivial: there is a partition only if $\mathcal{W} = \{w\}$ and $|w| \leq K$. In the equality-free case we must have $\|\mathcal{W}\| \leq K(K+1)/2$ if a partition exists. However, this is not a sufficient condition. For example, $\mathcal{W} = \{aa, aa, aa\}$ with $K = 3$ satisfies this condition, and yet there is no equality-free multiple string partition. It remains open whether the problem is solvable in polytime in this case.

It is also easy to check that in all cases, the problem becomes trivial if the maximal size of partition $K = 1$: the partition exists if every letter appears exactly once in the partitioned string(s). We have shown that the partition problems are **NP**-complete in the equality-free and prefix/suffix-free cases if $K = 2$ and in the factor-free and prefix&suffix-free cases if $K = 3$. We conjecture that problems remain **NP**-complete for higher values of K, although we do not see a simple extension of our proofs which could show that. On the other hand, in the factor-free and prefix&suffix-free cases, the problem becomes polytime solvable if $K = 2$.

Theorem 17. *Factor-Free (Prefix&Suffix-Free) Multiple String Partition is in* ***P*** *for partition size $K = 2$.*

Proof. First, note that two strings s and s', with length at most two, are factor-free if and only if they are prefix&suffix-free, i.e., it is enough to consider one of these two problems. Without loss of generality, we will consider

PS-MSP. Second, observe that if a letter b appears at least twice in $\mathcal{W}$, then a valid partition cannot select b by itself as it would be a prefix or a suffix of any other selected string that contains another occurrence of b, i.e., each occurrence of b has to be selected together with the preceding or succeeding letter. Partition the alphabet Σ into two subsets Σ_1 and Σ_2, where Σ_1 contains only the letters that appear once in $\mathcal{W}$ and Σ_2 the letters that appear at least twice. Consider string $w \in \mathcal{W}$. Divide w into parts separated by letters from Σ_1: $w = w_1 a_1 w_2 a_2 \ldots a_t w_{t+1}$, where $a_1, \ldots, a_t \in \Sigma_1$ and $w_1, \ldots, w_{t+1} \in \Sigma_2^*$. Consider the part $w_i = b_1 \ldots b_{|w_i|}$, where $b_1, \ldots, b_{|w_i|} \in \Sigma_2$. In any valid partition of $\mathcal{W}$, there are exactly two ways that w_i could be partitioned. If $|w_i|$ is odd, the selected strings containing letters of w_i are either $a_i b_1, b_2 b_3, \ldots, b_{|w_i|-1} b_{|w_i|}$ or $b_1 b_2, \ldots, b_{|w_i|-2} b_{|w_i|-1}, b_{|w_i|} a_{i+1}$. If $|w_i|$ is even, the selected strings are either $a_i b_1, b_2 b_3, \ldots, b_{|w_i|-2} b_{|w_i|-1}, b_{|w_i|} a_{i+1}$ or $b_1 b_2, \ldots, b_{|w_i|-1} b_{|w_i|}$. To find a valid partition of $\mathcal{W}$ it is enough to determine which option to choose for every w_i of each $w \in \mathcal{W}$ so that (a) no $bb' \in \Sigma_2^2$ is selected twice and (b) no a_i is included simultaneously in two selected strings. To do this we will construct an instance of 2SAT which has a solution if and only if there is a collection of choices for all w_i's that satisfies (a) and (b). Create a variable x_{w_i} for every w_i of each $w \in \mathcal{W}$. The "false" value of x_{w_i} will represent the partition of w_i in which $a_i b_1$ is selected. Note that if $w_i = \varepsilon$ or if $i = 1$, the value of variable x_{w_i} is always "true". Similarly, if $i = t+1$ and $|w_i| > 0$, the value of x_{w_i} is always "false" if $|w_i|$ is odd, and is always "true" otherwise. We will encode condition (a) using 2-clauses on these variables. If $|w_i|$ is odd, let $F(w_i) = \{b_2 b_3, \ldots, b_{|w_i|-1} b_{|w_i|}\}$ and $T(w_i) = \{b_1 b_2, \ldots, b_{|w_i|-2} b_{|w_i|-1}\}$. If $|w_i|$ is even, let $F(w_i) = \{b_2 b_3, \ldots, b_{|w_i|-2} b_{|w_i|-1}\}$ and $T(w_i) = \{b_1 b_2, \ldots, b_{|w_i|-1} b_{|w_i|}\}$. Assume $w, w' \in \mathcal{W}$ and let $w = w_1 a_1 w_2 a_2 \ldots a_t w_{t+1}$ and $w' = w'_1 a'_1 w'_2 a'_2 \ldots a'_{t'} w'_{t'+1}$, where $a_i, a'_i \in \Sigma_1$ and $w_i, w'_i \in \Sigma_2^*$. If $F(w_i) \cap F(w'_j) \neq \emptyset$, then a valid partition of $\mathcal{W}$ cannot partition both w_i and w'_j using their first ("false") choices. We can represent this with a 2-clause $x_{w_i} \vee x_{w'_j}$. Similarly, non-empty intersections $F(w_i) \cap T(w'_j)$, $T(w_i) \cap F(w'_j)$ and $T(w_i) \cap T(w'_j)$ can be represented with 2-clauses $x_{w_i} \vee \neg x_{w'_j}$, $\neg x_{w_i} \vee x_{w'_j}$ and $\neg x_{w_i} \vee \neg x_{w'_j}$, respectively. To express condition (b), it is enough to guarantee that a_i is not simultaneously selected in the partition of w_{i-1} and w_i. Note that a_i is not selected in the partition of w_{i-1} if and only if either (i) $|w_{i-1}|$ is odd and $x_{w_{i-1}}$ is "false", or (ii) $|w_{i-1}|$ is even and $x_{w_{i-1}}$ is "true". Also, a_i is not selected in the partition of w_i if and only if x_{w_i} is "true". Hence, condition (b) can be expressed with 2-clause $\neg x_{w_{i-1}} \vee x_{w_i}$ if $|w_{i-1}|$ is odd and 2-clause $x_{w_{i-1}} \vee x_{w_i}$ if $|w_{i-1}|$ is even, for every $i = 2, \ldots, t+1$ and $w \in \mathcal{W}$. Since 2SAT is solvable in linear time [21], we can decide whether there is a partition of $\mathcal{W}$ in polynomial time.

9. Conclusion

We have established the complexity of the following fundamental question: given a string w over an alphabet Σ and an integer K, can w be partitioned into factors no longer than K such that no two *collide*? We have shown this problem is **NP**-complete for versions requiring that no string in the partition is a copy/factor/prefix or suffix/prefix/suffix of another. We also considered generalized versions of these problems that must partition a set of input strings. Furthermore, we have shown the problems remain hard even for binary strings. This resolves a number of open questions from previous work [18] and establishes the theoretical hardness of a practical problem in contemporary synthetic biology, specifically, the oligo design for the gene synthesis problem.

[1] R. M. Karp, Mapping the genome: some combinatorial problems arising in molecular biology, in: STOC '93: Proceedings of the twenty-fifth annual ACM symposium on Theory of computing, ACM, New York, NY, USA, 1993, pp. 278–285.

[2] D. A. Christie, R. W. Irving, Sorting strings by reversals and by transpositions, SIAM Journal on Discrete Mathematics 14 (2) (2001) 193–206.

[3] N. Eriksen, (1+ ε)-Approximation of sorting by reversals and transpositions, Theoretical Computer Science 289 (1) (2002) 517–529.

[4] T. Hartman, R. Shamir, A simpler and faster 1.5-approximation algorithm for sorting by transpositions, Information and Computation 204 (2) (2006) 275 – 290.

[5] S. Yancopoulos, O. Attie, R. Friedberg, Efficient sorting of genomic permutations by translocation, inversion and block interchange, Bioinformatics 21 (16) (2005) 3340–3346.

[6] S. Hannenhalli, Polynomial algorithm for computing translocation distance between genomes, Discrete Applied Mathematics 71 (1) (1996) 137–151.

[7] A. Goldstein, P. Kolman, J. Zheng, Minimum common string partition problem: Hardness and approximations, The Electronic Journal of Combinatorics 12 (R50) (2005) 1.

[8] E. Myers, G. Sutton, A. Delcher, I. Dew, D. Fasulo, M. Flanigan, S. Kravitz, C. Mobarry, K. Reinert, K. Remington, et al., A whole-genome assembly of Drosophila, Science 287 (5461) (2000) 2196.

[9] P. Pevzner, H. Tang, M. Waterman, An Eulerian path approach to DNA fragment assembly, Proceedings of the National Academy of Sciences of the United States of America 98 (17) (2001) 9748.

[10] D. Kumar, C. Gustafsson, D. F. Klessig, Validation of RNAi silencing specificity using synthetic genes: salicylic acid-binding protein 2 is required for innate immunity in plants, Plant Journal 45 (5) (2006) 863–868.

[11] J. C. Cox, J. Lape, M. A. Sayed, H. W. Hellinga, Protein fabrication automation, Protein Science 16 (3) (2007) 379–390.

[12] D. Gibson, G. Benders, C. Andrews-Pfannkoch, E. Denisova, H. Baden-Tillson, J. Zaveri, T. Stockwell, A. Brownley, D. Thomas, M. Algire, et al., Complete chemical synthesis, assembly, and cloning of a mycoplasma genitalium genome, Science 319 (5867) (2008) 1215–1220.

[13] C. T. Lin, Y. C. Tsai, L. He, R. Calizo, H. H. Chou, T. C. Chang, Y. K. Soong, C. F. Hung, C. H. Lai, A DNA vaccine encoding a codon-optimized human papillomavirus type 16 E6 gene enhances CTL response and anti-tumor activity, Journal of Biomedical Science 13 (4) (2006) 481–488.

[14] A. Cid-Arregui, V. Juárez, H. zur Hausen, A synthetic E7 gene of human papillomavirus type 16 that yields enhanced expression of the protein in mammalian cells and is useful for DNA immunization studies, Journal of Virology 77 (8) (2003) 4928–4937.

[15] R. Roden, T. Wu, Preventative and therapeutic vaccines for cervical cancer, Expert Review of Vaccines 2 (4) (2003) 495–516.

[16] K. Chlichlia, V. Schirrmacher, R. Sandaltzopoulos, Cancer immunotherapy: Battling tumors with gene vaccines, Current Medicinal Chemistry - Anti-Inflammatory & Anti-Allergy Agents 4 (2005) 353–365.

[17] W. P. Stemmer, A. Crameri, K. D. Ha, T. M. Brennan, H. L. Heyneker, Single-step assembly of a gene and entire plasmid from large numbers of oligodeoxyribonucleotides, Gene 164 (1) (1995) 49–53.

[18] A. Condon, J. Maňuch, C. Thachuk, Complexity of a collision-aware string partition problem and its relation to oligo design for gene synthesis, in: Computing and Combinatorics Conference (COCOON), Vol. 5902 of Lecture Notes in Computer Science, 2008, pp. 265–275.

[19] A. Condon, J. Maňuch, C. Thachuk, The complexity of string partitioning, in: Combinatorial Pattern Matching, Vol. 7354 of Lecture Notes in Computer Science, Springer, 2012, pp. 159–172.

[20] C. H. Papadimitriou, Computational Complexity, Addison-Wesley, 1994.

[21] B. Aspvall, M. F. Plass, R. E. Tarjan, A linear-time algorithm for testing the truth of certain quantified boolean formulas, Information Processing Letters 8 (3) (1979) 121–123.