SEQUENCING FROM COMPOMERS: THE PUZZLE

SEBASTIAN BÖCKER

Abstract. The board game Fragmind™ poses the following question: The player has to reconstruct an (unknown) string s over the alphabet Σ. To this end, the game reports the following information to the player, for every character $x \in \Sigma$: First, the string s is cleaved wherever the character x is found in s. Second, every resulting fragment y is scrambled by a random permutation so that the only information left is how many times y contains each character $\sigma \in \Sigma$. These scrambled fragments are then reported to the player.

Clearly, distinct strings can show identical cleavage patterns for all cleavage characters. In fact, even short strings of length 30+ usually have non-unique cleavage patterns. To this end, we introduce a generalization of the game setup called *Sequencing From Compomers*: We also generate those fragments of s that contain up to k uncleaved characters x, for some small and fixed threshold k. This modification dramatically increases the length of strings that can be uniquely reconstructed. We show that it is NP-hard to decide whether there exists some string compatible with the input data, but we also present a branch-and-bound runtime heuristic to find all such strings: The input data are transformed into subgraphs of the de Bruijn graph, and we search for walks in these subgraphs simultaneously.

The above problem stems from the analysis of mass spectrometry data from base-specific cleavage of DNA sequences, and gives rise to a completely new approach to DNA de-novo sequencing.

This is a reprint of: S. Böcker. Sequencing from compomers: The puzzle. Theory Comput Syst, 39(3):455–471, 2006. https://doi.org/10.1007/s00224-005-1238-y

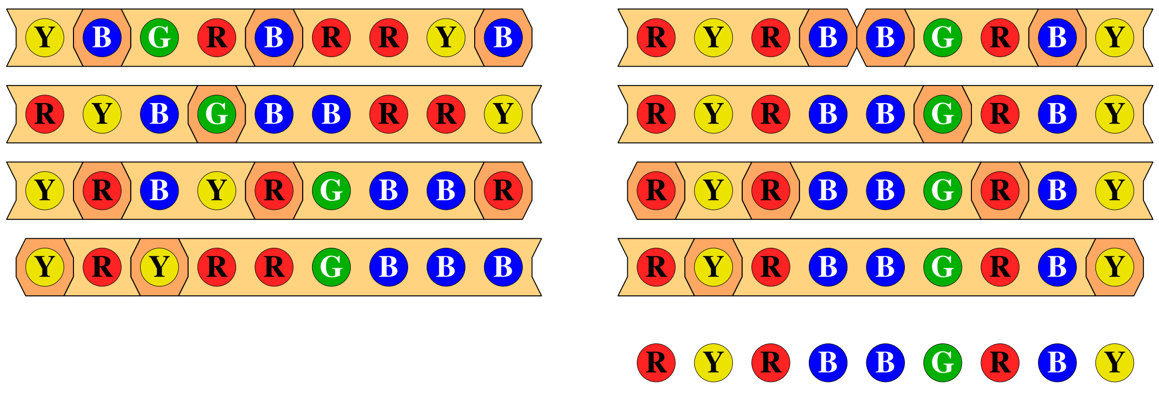

Figure 1. Modified Fragmind puzzle with solution on the right.

1. Introduction

The task of the solitaire board game Fragmind™ (see `www.fragmind.com/game`) is to reconstruct a color sequence from scrambled fragment information. With slight modifications, a typical instance of this puzzle looks like Fig. 1 on the left.[1] Here, every row corresponds to a unique cleavage character $x \in \Sigma$ over the alphabet $\Sigma := \{B,G,R,Y\}$. For every such character x we are given a collection of (X-shaped) fragments, plus some (O-shaped) cleavage sites. All fragments contain characters from $\Sigma - \{x\}$ while the cleavage sites consist of exactly one character x. The task of the game is to reorder the fragments/cleavage sites, as well as the characters inside the fragments in such a way that every column consists of exactly one character, see Fig. 1 on the right. To do so, there are two types of permissible moves:

- First, for every row we are allowed to arbitrarily reorder the fragments and cleavage sites, with the exception that two fragments must not touch each other — they must be separated by at least one cleavage site. Note that we are not allowed to swap fragments between rows.
- Second, for every fragment we can arbitrarily reorder all characters in this fragment.

[1] We slightly modify the definition of the puzzle, to be closer to the biochemical problem Fragmind was derived from. Fragmind puzzles ignore the rightmost fragment in every row, for the sake of simplicity.

For the puzzle presented in Fig. 1 on the left, we show how to reorder the first line, corresponding to cleavage character B: First, we reorder fragments and cleavage characters to RRY, B, B, GR, B, Y where no two fragments touch each other. Next, we reorder the characters in the first fragment from RRY to RYR. The resulting string RYRBBGRBY is a solution of this puzzle, because all columns agree on the right side of Fig. 1.

Where do problems of this type occur? The puzzle directly stems from the analysis of DNA sequences using mass spectrometry and base-specific cleavage. Our alphabet is the DNA alphabet $\Sigma := \{\text{A, C, G, T}\}$, and we are given a *sample string* over Σ (the target DNA molecule). We amplify the sample string using polymerase chain reaction. Next, we cleave the sample string with a base-specific biochemical cleavage reaction: That is, we cleave wherever a certain character (specific to the used cleavage reaction) appears in the string. Finally, we measure the masses of the resulting fragments using mass spectrometry. From a fragment's mass we can determine its (unordered) base compositions. See [1] for more details on the experimental setup of base-specific cleavage and mass spectrometry.

Combining base-specific cleavage and mass spectrometry has been successfully applied to a variety of problems such as SNP and mutation discovery [5], pathogen identification [8], or methylation analysis. These tasks are comparatively simple to answer computationally, because we just have to find a "best fit" of a measured mass spectrum with a rather small number of reference mass spectra. But for DNA de-novo sequencing, almost nothing is known about the underlying DNA sequence except its approximate length. In fact, only a small fraction of sequences show unique mass fingerprints even for short sequence lengths of 30 characters, see Section 8. It was therefore believed that base-specific cleavage and mass spectrometry cannot be used for de-novo sequencing of DNA.

Unexpectedly, a slight modification to the experimental setup heavily increases its resolving power. If we do not cleave every copy of the sample string at a certain position where the cleavage character x is present, but only a certain percentage of copies (say, 50 %) then we generate a much larger set of fragments and, hence, simplify the problem of reconstructing the sequence. This setup is comparable to the Partial Digestion Problem (PDP) [6, 7]. Accordingly, we call such cleavage reactions *partial* while the original cleavage reactions are called *complete*.

But there is a fundamental difference between PDP and our "experimental setup:" Using mass spectrometry, the intensity of a peak corresponds to the number of copies of the biomolecule generating the peak. Since this number drops exponentially in the number of uncleaved characters x in a fragment, it is likely that we can detect only those fragments that contain up to k uncleaved characters x, for some small threshold k. Unlike PDP where long fragments are likely to be detected, such long fragments cannot be detected in our setting, so we cannot adapt algorithms developed for PDP.

Several questions arise naturally in the context of (generalized) Fragmind puzzles: Given an instance of the problem, is there at least one solution? Given one solution s, is this solution unique? Ultimately, one wants to find all solutions of the problem in applications.

2. Formal representation of the problem

Mostly we will follow the notation of [1] and refer the reader there for a more detailed discussion.

Let s and y be strings over the alphabet Σ. We denote the maximal number of non-overlapping occurrences of y in s by $\operatorname{ord}_y(s)$.

For a string s and a character x, we define the *string spectrum* $\mathcal{S}(s, x)$ of s, x by:

$$\mathcal{S}(s,x) := \{y \in \Sigma^* : xyx \text{ is a substring of } xsx\} \tag{1}$$

The string spectrum $\mathcal{S}(s, x)$ consists of those substrings of s that are bounded by x or by the ends of s. Here, we call s *sample string* and x *cleavage character*, while the elements $y \in \mathcal{S}(s, x)$ will be called *fragments* of s (under x).

We use special characters $\mathbf{0}, \mathbf{1}$ to uniquely identify start and end of the sample string, what reduces the symmetry of the problem. The set of all strings in Σ^* with attached prefix $\mathbf{0}$ and suffix $\mathbf{1}$ is denoted $\mathbf{0}\Sigma^*\mathbf{1} := \{\mathbf{0}s\mathbf{1} : s \in \Sigma^*\}$. The use of special characters $\mathbf{0}, \mathbf{1}$ is motivated by characteristics of the underlying biochemistry: using base-specific cleavage, terminal fragments in general differ in mass from inner fragments with otherwise identical sequence.[2]

For $k \in \mathbb{N} \cup \{\infty\}$, we define the k-string spectrum of s, x by:

$$\mathcal{S}_k(s,x) := \{y \in \mathcal{S}(s,x) : \operatorname{ord}_x(y) \le k\} \tag{2}$$

The integer k is called *(spectrum) order.*

[2]The mass of a fragment is the summed mass of its characters, *plus* a correction factor that depends on the molecule's terminals: For example, RNAse A cleavage produces 5' hydroxyl and 3' phosphate terminals. Inner fragments have been cleaved by RNAse A on both sides, and we have to correct their mass by adding 18 Dalton (H_2O). This correction is different for the first and last fragment that have been cleaved on one side only.

Example 1. For the string $s = \mathbf{0}\mathrm{RYRBBGRBY}\mathbf{1}$ over $\Sigma = \{\mathrm{B, G, R, Y}\}$ from Fig. 1, we have:

$$\mathcal{S}_0(s,\mathrm{B}) = \{\mathbf{0}\mathrm{RYR}, \epsilon, \mathrm{GR}, \mathrm{Y}\mathbf{1}\} \qquad \mathcal{S}_0(s,\mathrm{G}) = \{\mathbf{0}\mathrm{RYRBB}, \mathrm{RBY}\mathbf{1}\}$$
$$\mathcal{S}_0(s,\mathrm{R}) = \{\mathbf{0}, \mathrm{Y}, \mathrm{BBG}, \mathrm{BY}\mathbf{1}\} \qquad \mathcal{S}_0(s,\mathrm{Y}) = \{\mathbf{0}\mathrm{R}, \mathrm{RBBGRB}, \mathbf{1}\}$$

For spectrum order $k = 0$ the string spectrum resembles the Fragmind puzzle. Higher orders give us more fragment information: for example,

$$\mathcal{S}_1(s,\mathrm{B}) = \{\mathbf{0}\mathrm{RYR}, \epsilon, \mathrm{GR}, \mathrm{Y}\mathbf{1}, \mathbf{0}\mathrm{RYRB}, \mathrm{BGR}, \mathrm{GRBY}\mathbf{1}\}.$$

As mentioned above, we cannot recover the order of characters inside a fragment from its mass, what is represented by the second rule of the Fragmind puzzle. To this end, we define a *compomer* c to be a "string without order:" Formally, c is a map from the alphabet Σ to the set of natural numbers including 0. To simplify notation, we use subindices for the number of characters in a compomer, omitting those characters with subindex 0: For example, $c = \mathrm{A}_2\mathrm{G}_1$ denotes the compomer $c(\mathrm{A}) = 2$, $c(\mathrm{C}) = 0$, $c(\mathrm{G}) = 1$, and $c(\mathrm{T}) = 0$. Finally, we write 0 for the *empty compomer*.

Clearly, we can map a string s to a compomer by counting the number of characters of each type in s. Let comp be this function: for example, $\mathrm{comp}(\mathrm{ACCTA}) = \mathrm{A}_2\mathrm{C}_2\mathrm{T}_1$. Finally, given two compomers c and c' over the alphabet Σ, we write $c \preceq c'$ if and only if $c(\sigma) \le c'(\sigma)$ holds for all $\sigma \in \Sigma$. So, $\preceq$ is a partial order on the set of compomers.

The *k-compomer spectrum* $\mathcal{C}_k(s,x)$ of s consists of the compomers of all fragments in the string spectrum $\mathcal{S}_k(s,x)$:

$$\mathcal{C}_k(s,x) := \mathrm{comp}\big(\mathcal{S}_k(s,x)\big) = \big\{\mathrm{comp}(y) : y \in \mathcal{S}(s,x),\ \ \mathrm{ord}_x(y) \le k\big\} \tag{3}$$

Example 2. For s from Example 1 and spectrum order $k = 0$ we calculate:

$$\begin{aligned} &\mathcal{C}_0(s,\mathrm{B}) = \{\mathbf{0}\,\mathrm{R}_2\mathrm{Y}_1, 0, \mathrm{G}_1\mathrm{R}_1, \mathrm{Y}_1\mathbf{1}\} \qquad &&\mathcal{C}_0(s,\mathrm{G}) = \{\mathbf{0}\,\mathrm{B}_2\mathrm{R}_2\mathrm{Y}_1, \mathrm{B}_1\mathrm{R}_1\mathrm{Y}_1\mathbf{1}\} \\ &\mathcal{C}_0(s,\mathrm{R}) = \{\mathbf{0}, \mathrm{Y}_1, \mathrm{B}_2\mathrm{G}_1, \mathrm{B}_1\mathrm{Y}_1\mathbf{1}\} \qquad &&\mathcal{C}_0(s,\mathrm{Y}) = \{\mathbf{0}\,\mathrm{R}_1, \mathrm{B}_3\mathrm{G}_1\mathrm{R}_2, \mathbf{1}\}\end{aligned} \tag{4}$$

Finally, we note that for Fragmind puzzles we do not only know the compomers of all fragments but also their multiplicities! To this end, let $\mathcal{M}_k(s,x)$ be the *multiset compomer spectrum* of s, x of spectrum order k, where we modify equations (1–3) to be multisets instead of "simple" sets. For a set X and $j \in \mathbb{N}$ we denote by $j \cdot X$ the multiset containing every element $x \in X$ exactly j times. As an example, for $s := \mathbf{0}\mathrm{GRGRGRGRG}\mathbf{1}$ and $x = \mathrm{R}$ the multiset compomer spectrum of order $k = 0$ is $\mathcal{M}_0(s,\mathrm{R}) = \{\mathbf{0}\mathrm{G}_1, \mathrm{G}_1, \mathrm{G}_1, \mathrm{G}_1, \mathrm{G}_1\mathbf{1}\} = \{\mathbf{0}\mathrm{G}_1\} \cup 3\cdot\{\mathrm{G}_1\} \cup \{\mathrm{G}_1\mathbf{1}\}$.

3. The Sequencing From Compomers Problem

Let $k \in \mathbb{N} \cup \{\infty\}$ be the fixed spectrum order, and let $S \subseteq \mathbf{0}\Sigma^*\mathbf{1}$ (or $S \subseteq \Sigma^*$) be the set of sample string candidates.[3] Now, the question is: Can we uniquely recover a string $s \in \mathbf{0}\Sigma^*\mathbf{1}$ from its (multiset) compomer spectra? To this end, we define the Sequencing From Compomers (SFC) Problem in four different flavors:

SFC$_{\mathbf{me}}$: Find all $s \in S$ with $\mathcal{M}_k(s,x) = \mathcal{M}_x$ for given multisets $\mathcal{M}_x$, $x \in \Sigma$.
SFC$_{\mathbf{mi}}$: Find all $s \in S$ with $\mathcal{M}_k(s,x) \subseteq \mathcal{M}_x$ for given multisets $\mathcal{M}_x$, $x \in \Sigma$.
SFC$_{\mathbf{se}}$: Find all $s \in S$ such that $\mathcal{C}_k(s,x) = \mathcal{C}_x$ holds for given sets $\mathcal{C}_x$, $x \in \Sigma$.
SFC$_{\mathbf{si}}$: Find all $s \in S$ such that $\mathcal{C}_k(s,x) \subseteq \mathcal{C}_x$ holds for given sets $\mathcal{C}_x$, $x \in \Sigma$.

Indices "me, mi, se, si" stand for multiset/set equality/inequality, respectively. SFC$_{\mathrm{me}}$ for $k = 0$ corresponds to the Fragmind puzzle described in the introduction, where multiplicities of all compomers are known. SFC$_{\mathrm{si}}$ and SFC$_{\mathrm{mi}}$ correspond to the more realistic setting of sequencing DNA using mass spectrometry data, where a detected peak will potentially be interpreted as many different compomers even though only one of these interpretations is correct. In addition, mass spectra contain false positive peaks (so-called "noise" peaks). So, it seems favorable to include all compomer interpretations of detected peaks, and to search for strings that satisfy the inclusion condition only. Furthermore, it is currently not viable to deduce the multiplicity of a fragment from the intensity of the corresponding peak in a measured mass spectrum. Hence, SFC$_{\mathrm{si}}$ is best suited for modeling experimental mass spectrometry data.

The corresponding SFC$_m$ *decision problem* is as follows, for $m \in \{\mathrm{me, mi, se, si}\}$: Given sets of compomers $\mathcal{C}_x$, or multisets of compomers $\mathcal{M}_x$, for $x \in \Sigma$, is there at least one string $s \in S$ that satisfies the conditions of SFC$_m$? In [1] we show that the SFC$_{\mathrm{si}}$ decision problem is NP-hard. But below we will see that the — seemingly less complex — SFC$_{\mathrm{me}}$ and SFC$_{\mathrm{mi}}$ decision problems are also NP-hard.

[3]A straightforward choice for the set of sample string candidates is the set of all strings with length in some interval, because we can often determine the correct sample string length up-front, and take into account the measurement inaccuracy.

Finally, we define the SFC$_m$ *reconstruction problem*: Given a set of sample string candidates $S \subseteq \mathbf{0}\Sigma^*\mathbf{1}$ and some sample string $s \in S$, set $\mathcal{C}_x := \mathcal{C}_k(s,x)$ and $\mathcal{M}_x := \mathcal{M}_k(s,x)$. Find the subset of sample strings $S' \subseteq S$ such that every $s' \in S'$ satisfies the conditions of SFC$_m$. Clearly, $s \in S'$ holds. In particular, we are interested in those strings $s \in S$ that can be uniquely recovered, that is $S' = \{s\}$.

Even though experimental mass spectrometry data cannot be modeled by an instance of SFC$_{\text{me}}$, it is reasonable to study SFC$_{\text{me}}$ because this problem is better suited for a combinatorial analysis. Also note that a string s that does not have "unique" multiset compomer spectra $\mathcal{M}(s,x)$ also has non-unique ("simple" set) compomer spectra $\mathcal{C}(s,x)$, for $x \in \Sigma$.

In this context, the following question arises: Given sample strings that can be uniquely recovered using equality of multisets (SFC$_{\text{me}}$), can those sample strings also be recovered in the more realistic SFC$_{\text{si}}$ model? How much discriminative power do we gain by adding information about compomer multiplicities and restricting the search by demanding set equality? We will address these questions in Section 8 below.

Remark. Given a string $s \in S$, let $S_m \subseteq S$ be the set of solutions for the SFC$_m$ reconstruction problem, for $m \in \{\text{me}, \text{mi}, \text{se}, \text{si}\}$. Then, $S_{\text{me}} \subseteq S_{\text{mi}} \subseteq S_{\text{si}}$ and $S_{\text{me}} \subseteq S_{\text{se}} \subseteq S_{\text{si}}$.

Example 3. For the string $s = \mathbf{0}$RYRBBGRBY$\mathbf{1}$ from Example 2 we set $\mathcal{C}_x := \mathcal{C}_0(s,x)$ as shown in (4). Can we uniquely recover s from this information?

Regarding SFC$_{\text{se}}$, we can show that s is the unique string in $\mathbf{0}\Sigma^*\mathbf{1}$ such that $\mathcal{C}_0(s,x) = \mathcal{C}_x$ holds for all $x \in \Sigma$: Let s' be such a string. We know that its first character is $\mathbf{0}$, and we can iteratively deduce longer prefixes of s'. Regarding the second character, we infer that $\mathbf{0}$R is a prefix: Otherwise, $\mathbf{0} \notin \mathcal{C}_0(s',\text{R})$ holds since the character $\mathbf{0}$ cannot appear somewhere else in s'. This is a contradiction to $\mathbf{0} \in \mathcal{C}_{\text{R}}$. Next, $\mathbf{0}\text{R}_1 \in \mathcal{C}_{\text{Y}}$ implies that $\mathbf{0}$RY is a prefix of s'. For the third character, $\text{Y}_1 \in \mathcal{C}_{\text{R}}$ allow us to append an R, but is not a sufficient condition: We can also "generate" this compomer further upstream in s'. But $\mathbf{0}\text{R}_1\text{Y}_1 \notin \mathcal{C}_{\text{B}}$, $\mathbf{0}\text{R}_1\text{Y}_1 \notin \mathcal{C}_{\text{G}}$, and $0 \notin \mathcal{C}_{\text{Y}}$ prohibits to append characters B, G, or Y, so $\mathbf{0}$RYR is a prefix of s'. Continuing in this way, we see that $\mathbf{0}$RYRBBGRB is a prefix of s'.

At this point, $0 \in \mathcal{C}_{\text{B}}$ allows us to append a "wrong" character B. So, assume that $\mathbf{0}$RYRBBGRBB is a prefix: At some point, we will append either Y or $\mathbf{1}$ to s', so there exists $c \in \mathcal{C}_0(s',\text{Y})$ with $\text{B}_4\text{G}_1\text{R}_2 \preceq c$. But no such compomer is an element of $\mathcal{C}_{\text{Y}}$, a contradiction. Hence, we cannot append B, and by the above reasoning we conclude that $\mathbf{0}$RYRBBGRBY is a prefix of s' and, finally, that $s' = s$.

In Section 7 we will formalize the reasoning of Example 3, leading to a recursive algorithm with two branch-and-bound conditions.

4. Transformations that do not change multiset compomer spectra

We now focus on the SFC$_{\text{me}}$ reconstruction problem: Given a sample string s, we want to find (all) transformations of s that do not change the multiset compomer spectra of s. Clearly, if s is a solution of an instance of SFC$_m$ then the transformed string is also a solution of this instance, for $m \in \{\text{me}, \text{mi}, \text{se}, \text{si}\}$. In finding all such transformations, we can solve the SFC$_{\text{me}}$ reconstruction problem.

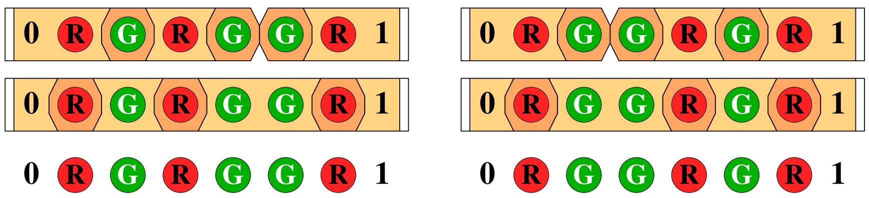

FIGURE 2. Puzzle with two solutions for $k = 0$. We added special characters $\mathbf{0}$, $\mathbf{1}$ to the game setup in accordance with the definitions.

Consider Fig. 2: For $\Sigma = \{\text{G,R}\}$ we define the multisets $\mathcal{M}_{\text{G}} := \{\mathbf{0}\,\text{R}_1, 0, \text{R}_1, \text{R}_1\mathbf{1}\}$ and $\mathcal{M}_{\text{R}} := \{\mathbf{0}, \text{G}_1, \text{G}_2, \mathbf{1}\}$. Then there exist two strings $s, s' \in \mathbf{0}\Sigma^*\mathbf{1}$ satisfying $\mathcal{M}_0(s,x) = \mathcal{M}_0(s',x) = \mathcal{M}_x$ for $x \in \Sigma$: namely, $s = \mathbf{0}$RGRGGR$\mathbf{1}$ and $s' = \mathbf{0}$RGGRGR$\mathbf{1}$. This example can easily be generalized for an arbitrary threshold k: We define $s := \mathbf{0}(\text{RG})^{k+1}\text{RG}(\text{GR})^{k+1}\mathbf{1}$ and set $\mathcal{M}_x := \mathcal{M}_k(s,x)$ for $x \in \Sigma$. One can easily check that the string $s' := \mathbf{0}(\text{RG})^{k+1}\text{GR}(\text{GR})^{k+1}\mathbf{1}$ also satisfies $\mathcal{M}_k(s',x) = \mathcal{M}_x$ for both $x \in \Sigma$.

For a string $s = s_1 \dots s_n$, let $s^{-1} := s_n \dots s_1$ denote the *inverse* string. The following proposition follows directly from $\text{comp}(s) = \text{comp}(s^{-1})$:

Proposition 1. *For a sample string $s \in \Sigma^*$, a cleavage character $x \in \Sigma$, and $k \in \mathbb{N} \cup \{\infty\}$, we have $\mathcal{M}_k(s,x) = \mathcal{M}_k(s^{-1},x)$.*

We will now present two simple transformations that do not change the multiset compomer spectra of a string. For $s \in \Sigma^*$ let $\Sigma_k(s) := \{\sigma \in \Sigma : \mathrm{ord}_\sigma(s) \geq k\}$ be the set of characters in s that appear at least k times. Lemma 2 generalizes the construction of Fig. 2, its proof is based on the observation $(yzy^{-1})^{-1} = yz^{-1}y^{-1}$ and on Proposition 1.

Lemma 2 (Flip transformation)**.** *Let $y \in \Sigma^*$ be a string, and let $z \in (\Sigma_{k+1}(y))^*$ be a non-empty string using only those characters that appear at least $k+1$ times in y.*

If yzy^{-1} is a substring of s, we can write $s = azb$ with strings a, b where y is a suffix of a, and y^{-1} is a prefix of b. Then, $s' := az^{-1}b$ has the same multiset compomer spectrum as s, that is $\mathcal{M}_k(s,x) = \mathcal{M}_k(s',x)$ for $x \in \Sigma$.

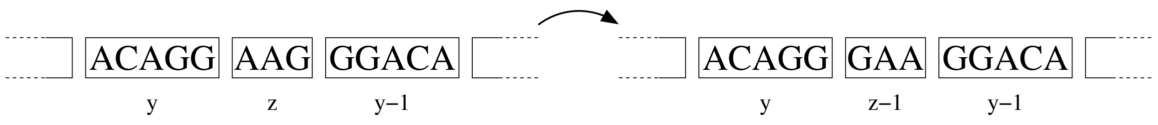

Figure 3. Example of a Flip transformation for $k = 1$.

The second transformation allows us to swap substrings that are enclosed by suitable strings y, y' as in the previous transformation:

Lemma 3 (Swap transformation)**.** *Let $y, y' \in \Sigma^*$ be strings, set $\Sigma' := \Sigma_{k+1}(y) \cap \Sigma_{k+1}(y')$, and let $z_1, z_2 \in (\Sigma')^*$ be non-empty strings using only those characters that appear at least $k+1$ times in y and y', and satisfy* $\mathrm{comp}(z_1) = \mathrm{comp}(z_2)$.

Let $a, b, c \in \Sigma^*$ be strings with $s = az_1bz_2c$ such that y is a suffix of a and of b, while y' is a prefix of b and of c. Then, $s' := az_2bz_1c$ has the same multiset compomer spectrum as s, that is $\mathcal{M}_k(s,x) = \mathcal{M}_k(s',x)$ for $x \in \Sigma$.

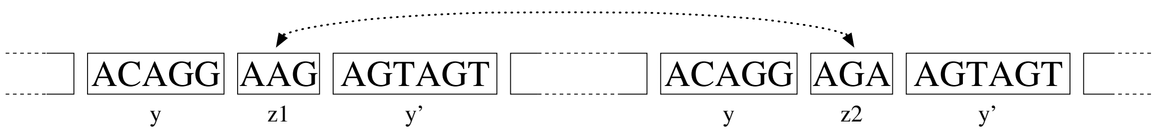

Figure 4. Example of a Swap transformation for $k = 1$.

The strings y, y^{-1} and y, y' act as boundaries so that for $x \in \Sigma_{k+1}(y)$ (or $x \in \Sigma'$, respectively) no fragment $w \in \mathcal{S}_k(s,x)$ stretches over, say, y: that is, no fragment w contains y as a substring.

We omit the formal proofs of these lemmata for the sake of brevity. There exist transformations preserving the multiset compomer spectra that cannot be composed of a sequence of Flip and Swap transformations: Regarding Lemma 3, if y, y' are *tight* boundaries with $\Sigma' = \Sigma_1(s)$, then we can do a Swap transformation even when $\mathrm{comp}(z_1) \neq \mathrm{comp}(z_2)$. Here, the task is to find a minimal set of transformations with the property of generating all preserving transformations. Work on this question is in progress.

5. Complexity of SFC$_{\mathrm{me}}$ and SFC$_{\mathrm{mi}}$

Let Σ be an arbitrary finite alphabet, and let $k \geq 0$ be an arbitrary but fixed spectrum order. We show below that it is computationally hard to decide whether there exists at least *one* (non-trivial) solution of SFC$_{\mathrm{me}}$ or SFC$_{\mathrm{mi}}$. Here we define the *problem order* of the SFC Problem to be the number of bits needed to represent all compomer sets $\mathcal{C}_x$ or $\mathcal{M}_x$, for $x \in \Sigma$ — not to be confused with the spectrum order k. Note that the length of strings solving some instance of SFC, can be exponential in the problem order, as we need only $\log n$ bits to store the compomer σ_n corresponding to a string of length n. In [1] we show that SFC$_{\mathrm{si}}$ is NP-hard — so it is likely that the same holds for SFC$_{\mathrm{se}}$.

Theorem 1. *For $|\Sigma| \geq 3$ and spectrum order $k \geq 0$, we are given a set of candidate strings $S \subseteq \mathbf{0}\Sigma^*\mathbf{1}$. Then it is NP-hard to decide whether the* SFC$_{\mathrm{mi}}$ *Problem with spectrum order k has at least one solution $s \in S$ satisfying $\mathcal{M}_k(s,x) \subseteq \mathcal{M}_x$ for all $x \in \Sigma$.*

Theorem 2. *For $|\Sigma| \geq 3$ and spectrum order $k \geq 0$, it is NP-hard to decide whether the* SFC$_{\mathrm{me}}$ *Problem with spectrum order k has at least one solution $s \in S := \mathbf{0}\Sigma^*\mathbf{1}$ such that equality $\mathcal{M}_k(s,x) = \mathcal{M}_x$ for all $x \in \Sigma$ is achieved.*

We will prove the latter theorem only, but one can prove Theorem 1 analogously by limiting the set S to strings of length $nb + 4n(k+1) + (n-1)(k+1)$. As we cannot check whether a string of exponential size is a solution, SFC$_{\mathrm{mi}}$ and SFC$_{\mathrm{me}}$ are not in NP.

For our complexity result, we make use of the 3-PARTITION Problem known to be NP-complete [4]:

3-PARTITION. *Given a set M of $3n$ elements, a bound $b \in \mathbb{N}$, and a size $\varphi(m) \in \mathbb{N}$ for each $m \in M$ with $b/4 < \varphi(m) < b/2$ and $\sum_{m \in M} \varphi(m) = nb$, can M be partitioned into n disjoint sets $M_1, \dots, M_n$ such that, for $1 \le i \le n$, $\sum_{m \in M_i} \varphi(m) = b$ holds?*

Given a solution of 3-PARTITION, then the constraints on $\varphi(m)$ imply that every set $M_i \subseteq M$ contains exactly three elements.

The formal proof of Theorem 2 can be found in the Appendix.

6. Sequencing graphs

Sequencing graphs were introduced in [1] and can be used to solve the SFC Problem. These directed graphs are subgraphs of the well-known de Bruijn graph, but the interesting twist here is that the alphabet underlying the de Bruijn graph is not the usual DNA alphabet, but instead a set of compomers over the DNA alphabet. We have slightly modified the definitions of [1] to allow a simpler incorporation of a source vertex in these graphs, see [2] for details.

A *directed graph* consists of a set V of vertices and a set $E \subseteq V^2 = V \times V$ of edges. An edge (v, v) for $v \in V$ is called a *loop*. We limit our attention to finite directed graphs with finite vertex sets. A *walk* in G is a finite sequence $p = (p_0, p_1, \dots, p_n)$ of elements from V with $(p_{i-1}, p_i) \in E$ for all $i = 1, \dots, n$, and $|p| := n$ is the *length* of p.

We start this section by an example, and show below how to formalize the construction.

Example 4. Let

$$s = \mathbf{0}\text{ACAATCAGCTATGGGCATACTACTCGCATCCGGAGT}\mathbf{1} \quad \in \mathbf{0}\Sigma^*\mathbf{1}$$

be our sample string over the DNA alphabet Σ. We set $\mathcal{C}_x := \mathcal{C}_1(s, x)$ for $x \in \Sigma$; for example,

$$\mathcal{C}_\text{T} = \{\mathbf{0}\,\text{A}_3\text{C}_1,\ \mathbf{0}\,\text{A}_4\text{C}_3\text{G}_1\text{T}_1,\ \text{A}_1\text{C}_2\text{G}_1,\ \dots,\ \text{A}_1\text{C}_2\text{G}_3,\ \text{A}_1\text{C}_2\text{G}_3\text{T}_1\mathbf{1},\ \mathbf{1}\}.$$

Next, we set $\Sigma_x := \{c \in \mathcal{C}_x : c(x) = 0\} \cup \{*\}$ for all $x \in \Sigma$ as the set of compomers with no uncleaved cut character, see (5) below; for example,

$$\Sigma_\text{T} = \{\mathbf{0}\,\text{A}_3\text{C}_1,\ \text{A}_1\text{C}_2\text{G}_1,\ \text{A}_1,\ \text{A}_1\text{C}_1\text{G}_3,\ \text{A}_1\text{C}_1,\ \text{A}_1\text{C}_2\text{G}_3,\ \mathbf{1},\ *\}.$$

For every $x \in \Sigma$, we define the directed graph $G_x = (V_x, E_x)$ by $V_x := \Sigma_x$, and $(u, v) \in E_x$ holds for $u, v \ne *$ if and only if $u + \text{comp}(x) + v \in \mathcal{C}_x$. Note that if (u, v) is an edge of G_x then (v, u) is an edge, too: In fact, these edges can be viewed as undirected. Furthermore, we include directed edges $(*, v)$ for all vertices $v \ne *$ that satisfy $v(\mathbf{0}) = 1$. We have displayed the four graphs resulting from this construction in Fig. 5. As an example, the graph G_T contains edges $(\text{A}_1\text{C}_2\text{G}_1, \text{A}_1)$ and $(\text{A}_1, \text{A}_1\text{C}_2\text{G}_1)$ since $\text{A}_1\text{C}_2\text{G}_1 + \text{T}_1 + \text{A}_1 = \text{A}_2\text{C}_2\text{G}_1\text{T}_1 \in \mathcal{C}_\text{T}$ holds; and it contains the loop $(\text{A}_1\text{C}_1, \text{A}_1\text{C}_1)$ because $\text{A}_1\text{C}_1 + \text{T}_1 + \text{A}_1\text{C}_1 = \text{A}_2\text{C}_2\text{T}_1 \in \mathcal{C}_\text{T}$ holds

What can we do with the graphs of Example 4? Consider cleavage character $x = \text{T}$, where $*$ is the source vertex of G_T. The first fragment of s under T is $\mathbf{0}$ACAA with compomer $\mathbf{0}\,\text{A}_3\text{C}_1$, and $(*, \mathbf{0}\,\text{A}_3\text{C}_1)$ is an edge of G_T. The second fragment of s under T is CAGC with compomer $\text{A}_1\text{C}_2\text{G}_1$, and $(\mathbf{0}\,\text{A}_3\text{C}_1, \text{A}_1\text{C}_2\text{G}_1)$ is an edge of G_T, and so on. So, the fragments of s under x correspond to a walk in G_x — and if we cannot find such a walk, then s cannot be a solution of our problem. Note that there exist edges in these graphs that do *not* correspond to subsequent fragments of the sample string, such as $(\text{A}_1\text{C}_2\text{G}_3, \text{A}_1)$ in G_T.

We will now formalize the graph construction of Example 4, but first we have to introduce some more definitions:

The *de Bruijn graph* of order $k \ge 1$ over the alphabet Σ' is a directed graph whose vertex set consists of all k-tuples over Σ', and (u, v) is an edge of the de Bruijn graph if the last $k - 1$ elements of u agree with the first $k - 1$ elements of v, that is, $u_{j+1} = v_j$ for $j = 1, \dots, k - 1$ [3]. We denote the de Bruijn graph of order k over Σ' by $B_k(\Sigma')$, and we denote an edge $\big((e_1, \dots, e_k), (e_2, \dots, e_{k+1})\big)$ by $(e_1, \dots, e_{k+1})$ for short.

In the following, we assume that we are given an instance of the Sequencing From Compomers Problem for set inclusion, SFC_si. In particular, for every cleavage character x we are given a set of compomers $\mathcal{C}_x$ as input. The *compomer alphabet* Σ_x consists of those compomers in $\mathcal{C}_x$ that do not contain the character x, plus a special *source character* $*$:

$$\Sigma_x := \{c \in \mathcal{C}_x : c(x) = 0\} \cup \{*\} \tag{5}$$

Note that we can componentwise add compomer "characters" $c, c' \in \Sigma_x$: For $* \in \Sigma_x$ we formally define $c + * = * + c = *$ for every compomer c. In the following, Σ_x will be the alphabet underlying the de Bruijn graph construction.

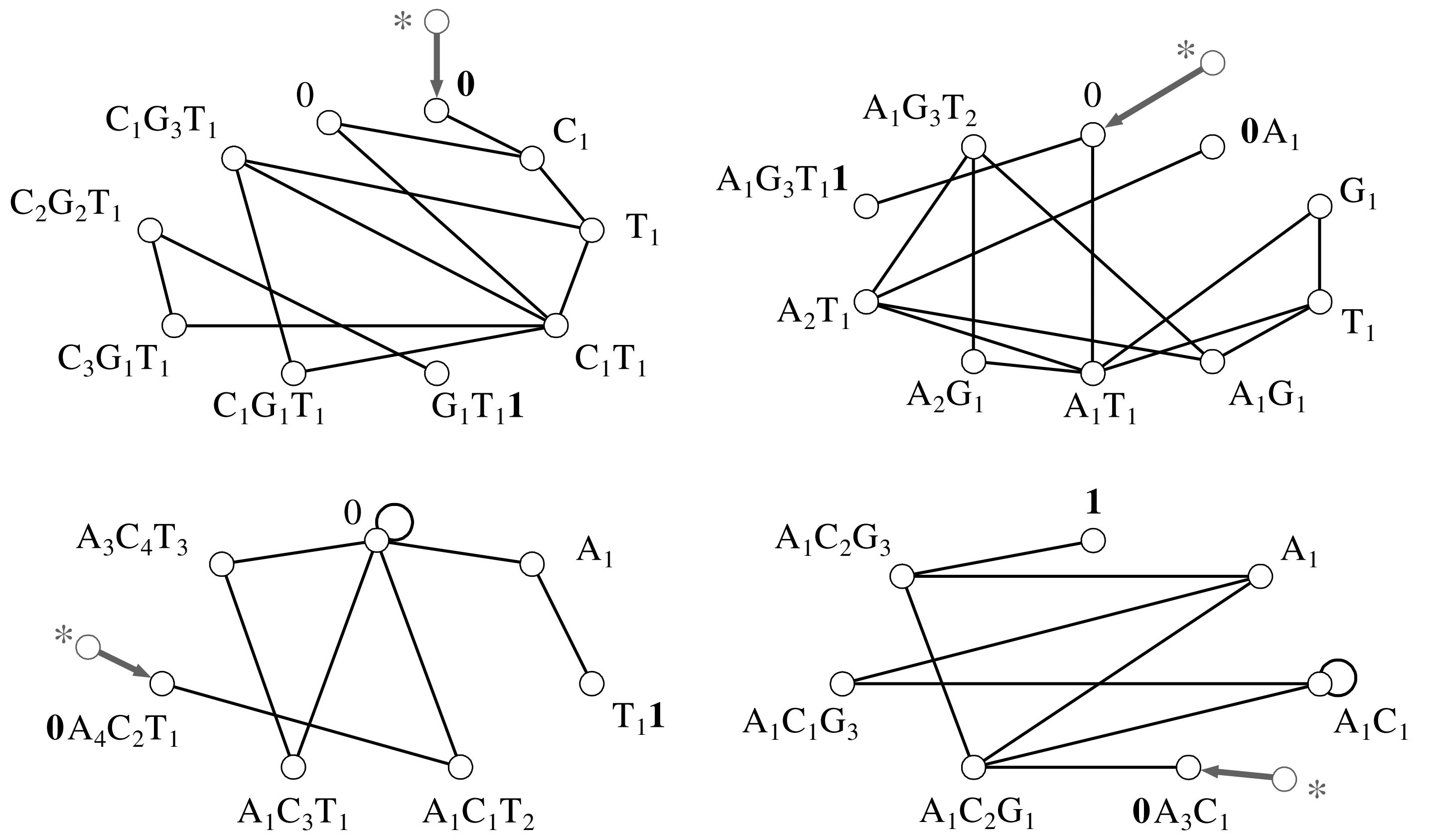

FIGURE 5. The four sequencing graphs of Example 4: the graph G_x for $x = \mathrm{A}$ (top left), $x = \mathrm{C}$ (top right), $x = \mathrm{G}$ (bottom left), and $x = \mathrm{T}$ (bottom right). Undirected edges represent directed edges in both directions.

Strings $s_0, \ldots, s_l$ form an *x-partitioning* of a string s if none of the strings $s_0, \ldots, s_l$ contains the character x, and $s = s_0 x s_1 x s_2 x \ldots x s_l$ holds. Clearly, there exists exactly one x-partitioning for every string s. For example, $(\mathbf{0}\mathrm{RYR}, \epsilon, \mathrm{GR}, \mathrm{Y}\mathbf{1})$ is the unique B-partitioning of $s = \mathbf{0}\mathrm{RYRBBGRBY}\mathbf{1}$.

Finally, a string s is called *compatible* with a walk $p = p_0 \ldots p_{|p|}$ in the de Bruijn graph $B_k(\Sigma_x)$ if the x-partitioning $s_0, \ldots, s_l$ of s satisfies $l = |p|$ and

$$p_j = \big(c_{j-k+1}, c_{j-k+2}, \ldots, c_j\big) \quad \text{for } j = 0, \ldots, l\,, \tag{6}$$

where $c_j := \mathrm{comp}(s_j)$ for $j = 0, \ldots, l$, and $c_{-j} := *$ for all integers $j > 0$. By definition, $(*, \ldots, *)$ is the first vertex of such p.

For every string s that is a solution of the $\mathrm{SFC_{si}}$ Problem, there exists a unique walk p in the de Bruijn graph that is compatible with s, see [1, Lemma 3]. For $s = \mathbf{0}\mathrm{RYRBBGRBY}\mathbf{1}$ from Example 1, the unique compatible walk in $B_1(\Sigma_\mathrm{B})$ is $(*), (\mathbf{0}\,\mathrm{R}_2\mathrm{Y}_1), (0), (\mathrm{G}_1\mathrm{R}_1), (\mathrm{Y}_1\mathbf{1})$. We now "thin out" the de Bruijn graph so that the resulting graph still contains all walks compatible with any solution of $\mathrm{SFC_{si}}$, but we remove those edges that cannot be part of any such walk.

For an edge $e = (e_1, \ldots, e_{k+1})$ of the de Bruijn graph $B_k(\Sigma_x)$, we use the notation

$$e_{[i,j]} := e_i + \mathrm{comp}(x) + e_{i+1} + \mathrm{comp}(x) + \cdots + e_{j-1} + \mathrm{comp}(x) + e_j \tag{7}$$

for $1 \le i \le j \le k+1$; recall that the alphabet underlying the de Bruijn graph is a set of compomers.[4]

Now, we define the *sequencing graph* $G_k(\mathcal{C}_x, x)$ of order $k \ge 1$ as follows: This is an edge-induced sub-graph of $B_k(\Sigma_x)$ where $\Sigma_x = \{c \in \mathcal{C}_x : c(x) = 0\} \cup \{*\}$. An edge $e = (e_1, \ldots, e_{k+1})$ of $B_k(\Sigma_x)$ belongs to the sequencing graph if and only if

- $e_{[i,j]} \in \mathcal{C}_x \cup \{*\}$ holds for all $1 \le i \le j \le k+1$, and
- $e_j = *$ for some j implies $e_i = *$ for all $1 \le i \le j$.

The first condition assures that we keep only those edges that might be needed for some compatible walk, whereas the second condition only prevents us from using the source character "in the middle" of some vertex. In Example 4,

[4]Note that $e_{[i,j]} = *$ holds if and only if there exists an index $i' \in [i,j]$ such that $e_{i'} = *$.

we have in fact constructed the four sequencing graphs $G_1(\mathcal{C}_x, x)$ for $x \in \{\mathrm{A}, \mathrm{C}, \mathrm{G}, \mathrm{T}\}$, omitting superfluous edges $(*, v)$ for those vertices v that do not contain the initial character $\mathbf{0}$.

Lemma 3 in [1] guarantees that s satisfies $\mathcal{C}_k(s, x) \subseteq \mathcal{C}_x$ if and only if there exists a walk p in the sequencing graph $G_k(\mathcal{C}_x, x)$ such that s is compatible with p. Hence, we can find all solutions to the $\mathrm{SFC_{si}}$ Problem by constructing walks in the sequencing graphs.

But what about the initial Fragmind problem with spectrum order $k = 0$? Here, we define the sequencing graph $G_0(\mathcal{C}_x, x)$ to be the complete directed graph with vertex set $\Sigma_x := \{c \in \mathcal{C}_x : c(x) = 0\} \cup \{*\}$, minus superfluous edges $(v, *)$ for $v \in \Sigma_x$. The high density of this graph indicates why the resolving power of $\mathrm{SFC_{me}}$ for $k = 0$ is extremely limited.

7. Walking the sequencing graphs

Suppose we are given an instance of the $\mathrm{SFC_{si}}$ Problem: the spectrum order k and sets of compomers $\mathcal{C}_x$ for $x \in \Sigma$. Our task is to find all sample strings $s \in \mathbf{0}\Sigma^*\mathbf{1}$ satisfying $\mathcal{C}_k(s, x) \subseteq \mathcal{C}_x$ for all $x \in \Sigma$. We limit our search to those strings s with length in a given interval.

We sketch a runtime-heuristic for this problem, see [1] for a detailed description. This depth-first search backtracks through string space, and implicitly builds walks in the directed sequencing graphs that are compatible with the constructed strings: To this end, suppose that strings $\hat{s}$ and $\bar{s}$ are compatible with walks in the sequencing graph $G_x := G_k(\mathcal{C}_x, x)$, and that $\hat{s}, \bar{s}$ share a common prefix s. Then, the walks compatible to $\hat{s}, \bar{s}$ also share a "prefix:" Let $(s_1, \ldots, s_l)$ be the x-partitioning of the prefix s, then both walks must start with the vertex sequence

$$(*, \ldots, *),\ (*, \ldots, *, c_1),\ (*, \ldots, *, c_1, c_2),\ \ldots,\ (c_{l-k}, \ldots, c_{l-1})$$

where $c_j := \mathrm{comp}(s_j)$ are the compomers of the partitioning. In our algorithm, we mark the last vertex $(c_{l-k}, \ldots, c_{l-1})$ of the common prefix walk with a token. In case sx is also a prefix of the strings, we can elongate this prefix walk and move our token from $(c_{l-k}, \ldots, c_{l-1})$ to $(c_{l-k+1}, \ldots, c_l)$, because $(s_1, \ldots, s_l, \epsilon)$ is the x-partitioning of sx.

Assume that we have built the sequencing graphs $G_x := G_k(\mathcal{C}_x, x)$ for $x \in \Sigma$. We initialize our algorithm by putting a token onto the source vertex $v_x \leftarrow (*, \ldots, *)$ in every sequencing graph G_x for $x \in \Sigma$. We recursively construct prefixes s of solutions, and we initialize $s \leftarrow \mathbf{0}$.

Let s be the current prefix, and for all $x \in \Sigma$, a token is currently placed on vertex v_x in the sequencing graph G_x. Let s_x be rightmost fragment of the x-partitioning of s, and set $c_x := \mathrm{comp}(s_x)$.

If we append a character $x \in \Sigma$ to s, can the resulting string sx still be a prefix of a solution? In this case, the following two conditions must be satisfied:

- By the above reasoning, we have to move our token in G_x from the current vertex $v_x = (v_1, \ldots, v_k)$ to $(v_2, \ldots, v_k, c_x)$. If the latter is not a vertex of G_x, or if no such edge exists, we cannot append x to s.
- Second, we check the following for all characters $\sigma \neq x$: Can we move our token in G_σ in the future? As we continue to append characters to sx, we will at some point append either σ or the terminal character $\mathbf{1}$ for the first time. At that point, we have to move our token in G_σ from $v_\sigma = (v_1, \ldots, v_k)$ to a new vertex $(v_2, \ldots, v_k, c)$, and the above implies $\mathrm{comp}(s_x x) \preceq c$.

  So, we check if there exists at least one edge $e = (v_1, \ldots, v_k, c)$ in G_σ leaving v_σ such that $c_x + \mathrm{comp}(x) \preceq c$ holds — if no such edge exists, we cannot move our token in G_σ in the future and, hence, we cannot append x to s.

To prevent infinite loops, we test whether the prefix length is below some upper bound before appending the next character. In case the prefix length is above some lower bound, we also test if we can append the terminal character $\mathbf{1}$: Doing so, we trigger an edge transition in all sequencing graphs simultaneously. If we can move our tokens accordingly in all graphs, then we output $s\mathbf{1}$ as a solution.

If there exists exactly one admissible character $x \in \Sigma$, we update the prefix, move the token in G_x, and continue. If there exist two or more admissible characters, we have to recursively test all alternatives. If there is no admissible character, we return to the previous level of recursion. Theorem 1 of [1] guarantees that this algorithm indeed returns all strings s with $\mathcal{C}_k(s, x) \subseteq \mathcal{C}_x$ for all $x \in \Sigma$.

Clearly, the algorithm is a runtime heuristic; but in view of the hardness of the decision problem, there is little hope that a polynomial runtime algorithm exists. Also, the number of solutions can be quite large, for example exponential in the fixed length of reconstructed strings [1, Example 3].

To solve $\mathrm{SFC_{mi}}$ or $\mathrm{SFC_{me}}$ Problems using this approach, we also store for every edge of a sequencing graphs, what compomers were tested in (7) with $j = k+1$ during construction. We cannot attach counters directly to the edges of a sequencing graph, because different edges may "use" the same compomers. In the recursion step, we

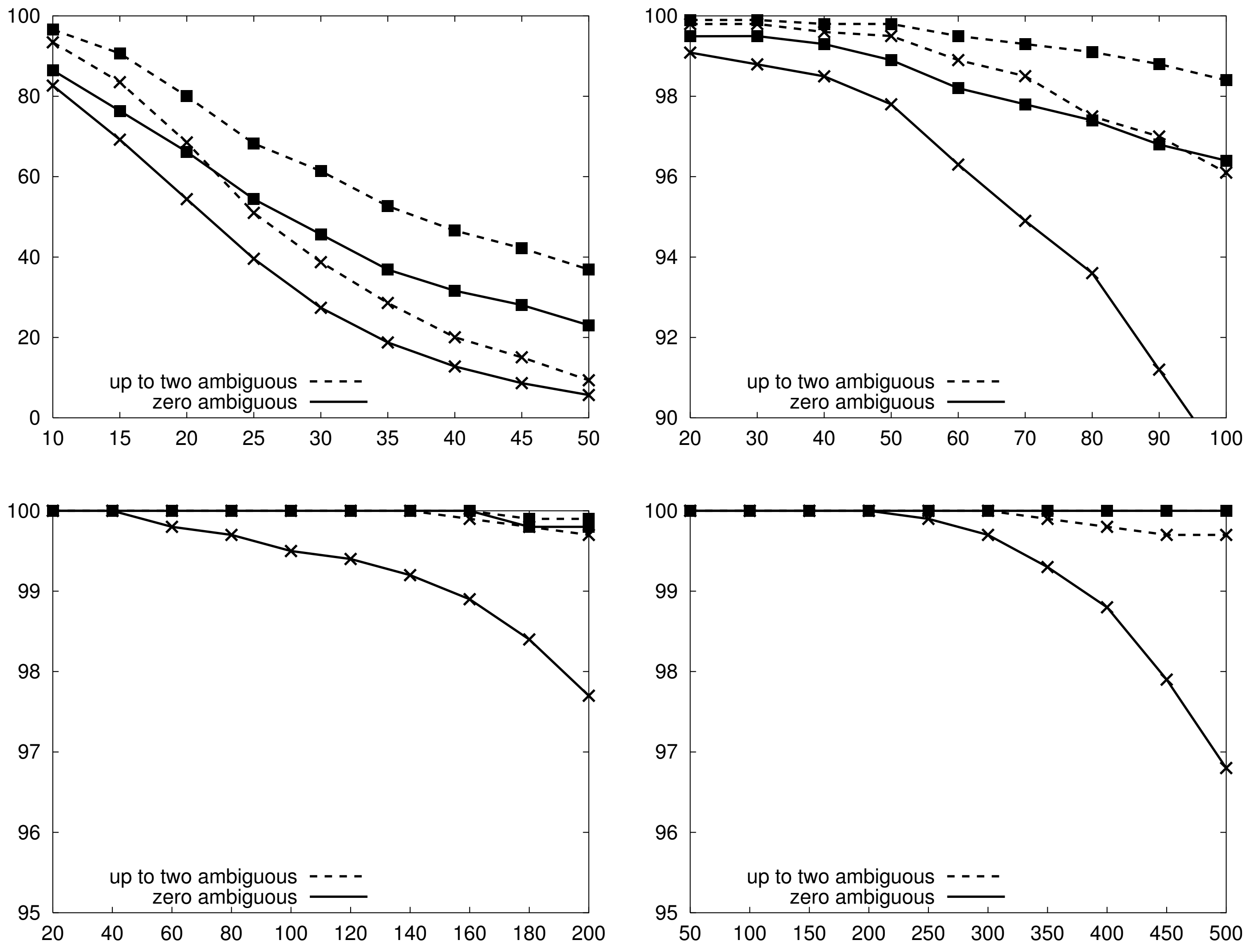

FIGURE 6. Simulation results: Percentage of strings that are uniquely reconstructed (solid line) or reconstructed with up to two ambiguous characters (dashed line) when varying string lengths (x-axis). Results for SFC_{me} (boxes) and SFC_{si} (crosses). Spectrum orders are $k = 0, 1$ (top, left and right) and $k = 2, 3$ (bottom, left and right). Observe the different scalings of the axes in the figures.

keep track how often every compomer has been generated by the current prefix, and stop as soon as appending a character exceeds the given limit.

8. Simulation results

We performed simulations comparing the resolving power of SFC_{me} and SFC_{si}. To this end, we generate random strings s of varying length, compute the (multiset) compomer spectra $\mathcal{C}_x := \mathcal{C}_k(s, x)$ and $\mathcal{M}_x := \mathcal{M}_k(s, x)$ for all $x \in \Sigma$, and finally search for all strings $s' \in \mathbf{0}\Sigma^*\mathbf{1}$ that are solutions to SFC_{me} and SFC_{si}. Since we can deduce the length of a solution string from the multiset compomer spectra, we search only for those solutions of SFC_{si} that have exactly the same length as the input string s, to allow a "fair" comparison of the resolving powers.

For spectrum orders $k = 0, 1, 2, 3$ we report the results of our simulations in Fig. 6. As expected, the knowledge of compomer multiplicities greatly enhances the chances of unique string reconstruction. Although it is unrealistic to believe that the experimental settings will ever allow to exactly estimate fragment multiplicities, it should be understood that even a rough estimate of these multiplicities increases the resolving power of our approach.

We do not only report the percentage of strings that were uniquely reconstructed from their (multiset) compomer spectra: For 2+ solutions we align all solutions without gaps and count the number of columns where the strings disagree. This is the number of *ambiguous characters*. In Fig. 6 we also report the percentage of strings that allowed reconstruction with up to two ambiguous characters. There were no strings that allowed reconstruction with a *single* ambiguous character.

As one can see, the resolving power for $k = 0$ corresponding to the original Fragmind puzzle and to complete cleavage, is very limited even if we take into account compomer multiplicities. But the resolving power increases by an order of magnitude for every increment of the spectrum order k.

9. Discussion

The Sequencing From Compomers Problem is a formal representation as well as a generalization of the Fragmind puzzle game. While SFC is computationally hard, we have introduced sequencing graphs that allow us to solve SFC in practice using a runtime-heuristic. The Fragmind puzzle and SFC are derived from the analysis of mass spectrometry data from base-specific cleavage experiments, and the presented approach might allow de-novo sequencing of DNA molecules with several hundred nucleotides. Various combinatorial problems in the context of SFC, at the same time challenging and relevant for applications, await their solution.

Acknowledgments

Sebastian Böcker is currently supported by "Deutsche Forschungsgemeinschaft" (BO 1910/1-1) within the Computer Science Action Program. Additional programming provided by Matthias Steinrücken. Simulations performed by Marcel Martin. I want to thank Zsuzsanna Lipták for proofreading earlier versions of this manuscript.

References

[1] S. Böcker. Sequencing from compomers: Using mass spectrometry for DNA de-novo sequencing of 200+ nt. *J Comput Biol*, 11(6):1110–1134, 2004.
[2] S. Böcker. Weighted sequencing from compomers: DNA de-novo sequencing from mass spectrometry data in the presence of false negative peaks. In *Proc. of German Conference on Bioinformatics (GCB 2004)*, volume P-53 of *Lecture Notes in Informatics*, pages 13–23, 2004.
[3] N. G. de Bruijn. A combinatorial problem. In *Indagationes Mathematicae*, volume VIII, pages 461–467. Koninklijke Nederlandse Akademie van Wetenschappen, Amsterdam: North Holland, 1946.
[4] M. R. Garey and D. S. Johnson. Complexity results for multiprocessor scheduling under resource constraints. *SIAM J Comput*, 4:397–411, 1975.
[5] C. P. Rodi, B. Darnhofer-Patel, P. Stanssens, M. Zabeau, and D. van den Boom. A strategy for the rapid discovery of disease markers using the MassARRAY system. *Biotechniques*, 32:S62–S69, 2002.
[6] J. Rosenblatt and P. D. Seymour. The structure of homometric sets. *SIAM J Algebra Discr*, 3:343–350, 1982.
[7] S. S. Skiena and G. Sundaram. A partial digest approach to restriction site mapping. *Bull Math Biol*, 56:275–294, 1994.
[8] F. von Wintzingerode, S. Böcker, C. Schlötelburg, N. H. Chiu, N. Storm, C. Jurinke, C. R. Cantor, U. B. Göbel, and D. van den Boom. Base-specific fragmentation of amplified 16S rRNA genes and mass spectrometry analysis: A novel tool for rapid bacterial identification. *Proc Natl Acad Sci U S A*, 99(10):7039–7044, 2002.

Appendix

Proof of Theorem 2. Recall that k is the fixed spectrum order. We may assume that $\{\mathrm{A}, \mathrm{B}, \mathrm{C}\} \subseteq \Sigma$ holds. We further assume that $b/4 \geq k+1$. Otherwise there exists a minimal constant $k' \in \mathbb{N}$ satisfying $k' \cdot b/4 \geq k+1$ and we can replace b by $k'b$ and φ by $k' \cdot \varphi$.

Given an instance of 3-PARTITION, we define an instance of $\mathrm{SFC}_{\mathrm{me}}$ by representing the values $\varphi(m)$ in the string solution by substrings $\mathrm{A}^{\varphi(m)}$. These substrings are separated by substrings of the form B^{k+1}, see (12). Finally, we use the character C to assure that all elements $m \in M$ are partitioned into sets M_i as desired, see (13). We prove that the instance of 3-PARTITION has a solution if and only if our instance of $\mathrm{SFC}_{\mathrm{me}}$ has a solution.

Recall that for a set X and $j \in \mathbb{N}$, we denote by $j \cdot X$ the multiset containing every element $x \in X$ exactly j times. We define multisets $\mathcal{M}_x$ for $x \in \Sigma$, where all unions over κ run $\kappa = 0, \dots, k$:

$$\mathcal{M}_{\mathrm{A}} := \bigcup_{\kappa} (2n(\kappa+1)+2) \cdot \{\mathrm{A}_\kappa \mathrm{B}_{k+1}\} \cup \bigcup_{\kappa} (n-1)(\kappa+1) \cdot \{\mathrm{A}_\kappa \mathrm{B}_{2(k+1)} \mathrm{C}_{k+1}\} \cup \bigcup_{m \in M, \kappa} (\varphi(m) - 1 - \kappa) \cdot \{\mathrm{A}_\kappa\} \tag{8}$$

$$\mathcal{M}_{\mathrm{B}} := \bigcup_{\kappa} (\kappa+1) \cdot \{\mathrm{A}_{\varphi(m)} \mathrm{B}_\kappa : m \in M\} \cup \bigcup_{\kappa} (n-1)(\kappa+1) \cdot \{\mathrm{B}_\kappa \mathrm{C}_{k+1}\} \cup \bigcup_{\kappa} 4n(k-\kappa) \cdot \{\mathrm{B}_\kappa\} \tag{9}$$

$$\mathcal{M}_{\mathrm{C}} := \bigcup_{\kappa} ((n-2)(\kappa+1)+2) \cdot \{\mathrm{A}_b \mathrm{B}_{4(k+1)} \mathrm{C}_\kappa\} \cup \bigcup_{\kappa} (n-1)(k-\kappa) \cdot \{\mathrm{C}_\kappa\} \tag{10}$$

$$\mathcal{M}_x := \{\mathrm{A}_{nb} \mathrm{B}_{4n(k+1)} \mathrm{C}_{(n-1)(k+1)}\} \quad \text{for } x \in \Sigma \setminus \{\mathrm{A}, \mathrm{B}, \mathrm{C}\} \tag{11}$$

First, we want to show that any solution $M_1, \ldots, M_n$ of 3-PARTITION yields a solution to the above instance of $\mathrm{SFC_{me}}$: For $i = 1, \ldots, n$ we set $M_i = \{m_i, m_i', m_i''\}$. We define the string

$$s_i := \mathrm{B}^{k+1}\mathrm{A}^{\varphi(m_i)}\mathrm{B}^{k+1}\mathrm{A}^{\varphi(m_i')}\mathrm{B}^{k+1}\mathrm{A}^{\varphi(m_i'')}\mathrm{B}^{k+1} \tag{12}$$

and compute for cleavage character $x = \mathrm{A}$:

$$\begin{aligned}\mathcal{M}_k(s_i, \mathrm{A}) &= \{\mathrm{A}_\kappa \mathrm{B}_{k+1} : \kappa = 0, \ldots, k\} \;\cup \mathcal{M}_\mathrm{A}^i \;\cup \{\mathrm{A}_\kappa \mathrm{B}_{k+1} : \kappa = 0, \ldots, k\}\\ \text{with } \mathcal{M}_\mathrm{A}^i &:= \bigcup_{\kappa=0,\ldots,k} 2(\kappa+1)\cdot\{\mathrm{A}_\kappa \mathrm{B}_{k+1}\} \;\cup \bigcup_{\substack{m \in M_i\\ \kappa=0,\ldots,k}} (\varphi(m) - 1 - \kappa)\cdot\{\mathrm{A}_\kappa\},\end{aligned}$$

where the sets $\{\mathrm{A}_\kappa \mathrm{B}_{k+1} : \kappa = 0, \ldots, k\}$ correspond to the beginning and end of s_i. For cleavage character $x = \mathrm{B}$ we compute

$$\mathcal{M}_k(s_i, \mathrm{B}) = \bigcup_{\kappa=0,\ldots,k} (\kappa+1)\cdot\{\mathrm{A}_{\varphi(m)}\mathrm{B}_\kappa : m \in M_i\} \;\cup \bigcup_{\kappa=0,\ldots,k} 4(k-\kappa)\cdot\{\mathrm{B}_\kappa\}.$$

From $\sum_{m \in M_i} \varphi(m) = b$, we infer $\mathrm{comp}(s_i) = \mathrm{A}_b\mathrm{B}_{4(k+1)}$. For the string

$$s := s(M_1, \ldots, M_n) := s_1\mathrm{C}^{k+1}s_2\mathrm{C}^{k+1}\cdots\mathrm{C}^{k+1}s_n \tag{13}$$

we can now conclude:

$$\begin{aligned}\mathcal{M}_k(s, \mathrm{A}) &= 2\cdot\{\mathrm{A}_\kappa\mathrm{B}_{k+1} : \kappa = 0, \ldots, k\} \;\cup \bigcup_{i=1,\ldots,n} \mathcal{M}_\mathrm{A}^i\\ &\quad \cup \bigcup_{\kappa=0,\ldots,k} (n-1)(\kappa+1)\cdot\{\mathrm{A}_\kappa\mathrm{B}_{2(k+1)}\mathrm{C}_{k+1}\} \quad = \mathcal{M}_\mathrm{A}\\ \mathcal{M}_k(s, \mathrm{B}) &= \bigcup_{i=1}^{n} \mathcal{M}_k(s_i, \mathrm{B}) \;\cup \bigcup_{\kappa=0,\ldots,k} (n-1)(\kappa+1)\cdot\{\mathrm{B}_\kappa\mathrm{C}_{k+1}\} \quad = \mathcal{M}_\mathrm{B}\\ \mathcal{M}_k(s, \mathrm{C}) &= \{\mathrm{A}_b\mathrm{B}_{4(k+1)}\mathrm{C}_\kappa : \kappa = 0, \ldots, k\} \;\cup \bigcup_{\kappa=0,\ldots,k} (n-2)(\kappa+1)\cdot\{\mathrm{A}_b\mathrm{B}_{4(k+1)}\mathrm{C}_\kappa\}\\ &\quad \cup \bigcup_{\kappa=0,\ldots,k} (n-1)(k-\kappa)\cdot\{\mathrm{C}_\kappa\} \;\cup \{\mathrm{A}_b\mathrm{B}_{4(k+1)}\mathrm{C}_\kappa : \kappa = 0, \ldots, k\} \quad = \mathcal{M}_\mathrm{C}\\ \mathcal{M}_k(s, x) &= \{\mathrm{A}_{nb}\mathrm{B}_{4n(k+1)}\mathrm{C}_{(n-1)(k+1)}\} \quad = \mathcal{M}_x \quad \text{for } x \in \Sigma \setminus \{\mathrm{A}, \mathrm{B}, \mathrm{C}\}\end{aligned}$$

So, $\mathcal{M}_k(s, x) = \mathcal{M}_x$ holds for all $x \in \Sigma$ as desired. This shows that given a solution $M_1, \ldots, M_n$ of 3-PARTITION, then $s(M_1, \ldots, M_n)$ is a solution of the above instance of $\mathrm{SFC_{me}}$.

Now, let us suppose that s is an arbitrary solution to the instance of $\mathrm{SFC_{me}}$ defined in Equations (8–11). Since

$$\{c \in \mathcal{M}_\mathrm{C} : c(\mathrm{C}) = 0\} = n\cdot\{\mathrm{A}_b\mathrm{B}_{4(k+1)}\} \;\cup (n-1)k\cdot\{0\} \tag{14}$$

we infer $\mathrm{comp}(s)(\mathrm{A}) = nb$ and $\mathrm{comp}(s)(\mathrm{B}) = 4n(k+1)$. Analogously, we conclude from (9) that $\mathrm{comp}(s)(\mathrm{C}) = (n-1)(k+1)$, so $|s| = nb + 4n(k+1) + (n-1)(k+1)$ is polynomial in b, n. We also derive from (14) that s has the form

$$s = c_0\, s_1\, c_1\, s_2\, c_2 \ldots c_{n-2}\, s_{n-1}\, c_{n-1} s_n\, c_n$$

where c_i are strings from $\{\mathrm{C}\}^*$, and $s_1, \ldots, s_n \in \{\mathrm{A}, \mathrm{B}\}^*$ such that $\mathrm{comp}(s_i)(\mathrm{A}) = b$ and $\mathrm{comp}(s_i)(\mathrm{B}) = 4(k+1)$ for all $i = 1, \ldots, n$ holds.

Now, we derive from (9) that

$$\{c \in \mathcal{M}_\mathrm{B} : c(\mathrm{B}) = 0\} = \{\mathrm{A}_{\varphi(m)} : m \in M\} \;\cup (n-1)\cdot\{\mathrm{C}_{k+1}\} \;\cup 4nk\cdot\{0\}$$

and, in particular, that $\mathrm{B}\mathrm{A}^{\varphi(m)}\mathrm{B}$ is a substring of s for all $m \in M$ — more precisely, for $y = \mathrm{B}\mathrm{A}^l\mathrm{B}$, we have $\mathrm{ord}_y(s) = |\{m \in M : \varphi(m) = l\}|$. For the sake of brevity, we ignore the case that A is a prefix or a suffix of s, that we can solve analogously. Since $\sum_{m \in M} \varphi(m) = nb = \mathrm{comp}(s)(\mathrm{A})$ we conclude that substrings of A's are always bounded by B's.

We iteratively define sets $M_1, \ldots, M_n$ that form a solution to 3-PARTITION. For $i = 1, \ldots, n$ let L_i be the multiset

$$L_i := \{l \in \mathbb{N} : \mathrm{B}\mathrm{A}^l\mathrm{B} \text{ is a substring of } s_i\}$$

where the multiplicity of $l \in L_i$ equals $\mathrm{ord}_y(s_i)$ for $y = \mathrm{BA}^l\mathrm{B}$. From $\mathrm{comp}(s_i)(\mathrm{A}) = b$ we conclude $\sum_{l \in L_i} l = b$. From

$$\sum_{i=1}^{n} \sum_{l \in L_i} l = n \cdot b = \mathrm{comp}(s)(\mathrm{A}) \tag{15}$$

we infer that $l \in L_i$ implies $l = \varphi(m)$ for some $m \in M$: Otherwise, the existence of $l \in L_i$ with $l \neq \varphi(m)$ for all $m \in M$ would contradict (15). So, we have $b/4 < l < b/2$ and, hence, $|L_i| = 3$. Choose $M_i \subseteq M$ with $|M_i| = 3$ such that $\varphi(M_i) = \{\varphi(m) : m \in M_i\} = L_i$, and set $M \leftarrow M \setminus M_i$. The existence of such set $M_i \subseteq M$ can be guaranteed due to the properties we derived above.

One can easily see that the resulting sets $M_1, \dots, M_n$ form a solution of the instance of 3-PARTITION. This concludes our proof that $\mathrm{SFC}_{\mathrm{me}}$ is NP-hard. □

AG Genominformatik, Technische Fakultät, Universität Bielefeld, PF 100 131, 33501 Bielefeld, Germany
Email address: boecker@CeBiTec.uni-bielefeld.de