Genomic Problems Involving Copy Number Profiles: Complexity and Algorithms

Manuel Lafond
Department of Computer Science, Universite de Sherbrooke, Quebec J1K 2R1, Canada
manuel.lafond@usherbrooke.ca

Binhai Zhu
Gianforte School of Computing, Montana State University, Bozeman, MT 59717, USA
bhz@montana.edu

Peng Zou
Gianforte School of Computing, Montana State University, Bozeman, MT 59717, USA
peng.zou@student.montana.edu

Abstract

Recently, due to the genomic sequence analysis in several types of cancer, genomic data based on *copy number profiles* (*CNP* for short) are getting more and more popular. A CNP is a vector where each component is a non-negative integer representing the number of copies of a specific segment of interest. The motivation is that in the late stage of certain types of cancer, the genomes are progressing rapidly by segmental duplications and deletions, and hence obtaining the exact sequences becomes difficult. Instead, the number of copies of important segments can be predicted from expression analysis and carries important biological information. Therefore, significant research has recently been devoted to the analysis of genomic data represented as CNP's.

In this paper, we present two streams of results. The first is the negative results on two open problems regarding the computational complexity of the Minimum Copy Number Generation (MCNG) problem posed by Qingge et al. in 2018. The *Minimum Copy Number Generation* (MCNG) is defined as follows: given a string S in which each character represents a gene or segment, and a CNP C, compute a string T from S, with the minimum number of segmental duplications and deletions, such that $cnp(T) = C$. It was shown by Qingge et al. that the problem is NP-hard if the duplications are tandem and they left the open question of whether the problem remains NP-hard if arbitrary duplications and/or deletions are used. We answer this question affirmatively in this paper; in fact, we prove that it is NP-hard to even obtain a constant factor approximation. This is achieved through a general-purpose lemma on set-cover reductions that require an exact cover in one direction, but not the other, which might be of independent interest. We also prove that the corresponding parameterized version is W[1]-hard, answering another open question by Qingge et al.

The other result is positive and is based on a new (and more general) problem regarding CNP's. The *Copy Number Profile Conforming (CNPC)* problem is formally defined as follows: given two CNP's C_1 and C_2, compute two strings S_1 and S_2 with $cnp(S_1) = C_1$ and $cnp(S_2) = C_2$ such that the distance between S_1 and S_2, $d(S_1, S_2)$, is minimized. Here, $d(S_1, S_2)$ is a very general term, which means it could be any genome rearrangement distance (like reversal, transposition, and tandem duplication, etc). We make the first step by showing that if $d(S_1, S_2)$ is measured by the breakpoint distance then the problem is polynomially solvable. We expect that this will trigger some related research along the line in the near future.

2012 ACM Subject Classification Theory of computation

Keywords and phrases Computational genomics, cancer genomics, copy number profiles, NP-hardness, approximation algorithms, FPT algorithms

Digital Object Identifier 10.4230/LIPIcs.CPM.2020.22

Related Version A full version of this paper is available at `https://arxiv.org/abs/2002.04778`.

Funding *Manuel Lafond*: ML was supported by NSERC of Canada, and BZ was partially supported by NNSF of China under project 61628207.

1 Introduction

In cancer genomics research, intra-tumor genetic heterogeneity is one of the central problems [15, 16, 21]. Understanding the origins of cancer cell diversity could help cancer prognostics [4, 14] and also help explain drug resistance [3, 6]. It is known for some types of cancers, such as high-grade serous ovarian cancer (HGSOC), that heterogeneity is mainly acquired through genome rearrangements and endoreduplications, the replication of the genome without the usual mitosis reproduction cycle. These result in aberrant *copy number profiles (CNPs)* – nonnegative integer vectors representing the numbers of genes occurring in a genome [17]. To understand how the cancer progresses, an evolutionary tree is certainly desirable, and inferring such a tree based on these genomic data becomes a new problem. In [20], Schwarz et al. proposed a way to construct a phylogenetic tree directly from integer copy number profiles, the underlying problem being to convert CNPs into one another using the minimum number of duplications/deletions [22]. This was recently followed with several other distances measures between CNPs that can be used to reconstruct cancer phylogenies [26, 9, 5, 19, 25].

In [8], a more complex distance computation was used as a subroutine to compute an ancestor profile given a set of k profiles. The problem was shown to be NP-hard, though an ILP formulation was given. In fact, Chowdhury *et al.* considered copy number changes at different levels, from single gene, single chromosome to whole genome, to enhance the tumor phylogeny reconstruction [2]. In [18], another fundamental problem was proposed. The motivation is that in the early stages of cancer, when large numbers of endoreduplications are still rare, genome sequencing is still possible. However, in the later stage we might only be able to obtain genomic data in the form of CNPs. This leads to the problem of comparing a sequenced genome with a genome with only copy-number information.

Given a genome G represented as a string and a copy number profile $\vec{c}$, the *Minimum Copy Number Generation (MCNG)* problem asks for the minimum number of deletions and duplications needed to transform G into any genome in which each character occurs as many times as specified by $\vec{c}$. Qingge et al. proved that the problem is NP-hard when the duplications are restricted to be tandem and posed several open questions: (1) Is the problem NP-hard when the duplications are arbitrary and/or deletions are allowed? (2) Does the problem admit a decent approximation? (3) Is the problem fixed-parameter tractable (FPT)? In this paper, we answer all these three open questions. We show that MCNG is NP-hard to approximate within any constant factor, and that it is W[1]-hard when parameterized by the solution size. The inapproximability follows from a new general-purpose lemma on set-cover reductions that require an exact cover in one direction, but not the other. The W[1]-hardness uses a new set-cover variant in which every optimal solution is an exact cover. These set-cover extensions can make reductions easier, and may be of independent interest.

We also consider a new fundamental problem called *Copy Number Profile Conforming (CNPC)*, which is defined as follows. Given two CNP's $\vec{c}_1$ and $\vec{c}_2$, compute two strings/genomes S_1 and S_2 with $cnp(S_1) = \vec{c}_1$ and $cnp(S_2) = \vec{c}_2$ such that the distance between S_1 and S_2, $d(S_1, S_2)$, is minimized. The distance $d(S_1, S_2)$ could be general, which means it could be any genome rearrangement distance (such as reversal, transposition, and tandem duplication, etc). We make the first step by showing that if $d(S_1, S_2)$ is measured by the breakpoint distance then the problem is polynomially solvable.

2 Preliminaries

A genome G is a string, i.e. a sequence of characters, all of which belong to some alphabet Σ (the characters of G can be interpreted as genes or segments – in this paper we assume the latter, i.e., Σ is a set of segments). We use genome and string interchangeably in this paper,

when the context is clear. A *substring* of G is a sequence of contiguous characters that occur in G, and a *subsequence* is a string that can be obtained from G by deleting some characters. We write $G[p]$ to denote the character at position p of G (the first position being 1), and we write $G[i..j]$ for the substring of G from positions i to j, inclusively. For $s \in \Sigma$, we write $G - s$ to denote the subsequence of G obtained by removing all occurrences of s.

We represent an alphabet as an ordered list $\Sigma = (s_1, s_2, \ldots, s_m)$ of distinct characters. Slightly abusing notation, we may write $s \in \Sigma$ if s is a member of this list. We write $n_s(G)$ to denote the number of occurrences of $s \in \Sigma$ in a genome G. A *Copy-Number Profile (or CNP)* on Σ is a vector $\vec{c} = \langle c_1, \ldots, c_{|\Sigma|} \rangle$ that associates each character s_i of the alphabet with a non-negative integer $c_i \in \mathbb{N}$ [1]; formally,

$$cnp(G) = \langle n_{s_1}(G), n_{s_2}(G), \ldots, n_{s_m}(G) \rangle.$$

We may write $\vec{c}(s)$ to denote the number associated with $s \in \Sigma$ in $\vec{c}$. We write $\vec{c} - s$ to denote the CNP obtained from $\vec{c}$ by setting $\vec{c}(s) = 0$. An an example, if $\Sigma = (a, b, c)$ and $G = abbcbbcca$, then $cnp(G) = \langle 2, 4, 3 \rangle$ and $\vec{c}(a) = 2$.

Deletions and duplications on strings

We now describe the two string events of *deletion* and *duplication*. Both are illustrated in Figure 1.

Sequence	*Operations*
$G_1 = abbc \cdot \sout{cab} \cdot cab$	$del(5, 7)$
$G_2 = a \cdot \underline{bbcc} \cdot ab$	$dup(2, 5, 6)$
$G_3 = abbcca \cdot \underline{bbcc} \cdot b$	

Figure 1 Three strings (or toy genomes), G_1, G_2 and G_3. From G_1 to G_2, a deletion is applied to $G_1[5..7]$. From G_2 to G_3, a duplication is applied to $G_2[2..5]$, with the copy inserted after position 6.

Given a genome G, a *deletion* on G takes a substring of G and removes it. Deletions are denoted by a pair (i, j) of the positions of the substring to remove. Applying deletion (i, j) to G transforms G into $G[1..i-1]G[j+1..n]$.

A *duplication* on G takes a substring of G, copies it and inserts the copy anywhere in G, except inside the copied substring. A duplication is defined by a triple (i, j, p) where $G[i..j]$ is the string to duplicate and $p \in \{0, 1, \ldots, i-1, j, \ldots n\}$ is the position *after* which we insert (inserting after 0 prepends the copied substring to G). Applying duplication (i, j, p) to G transforms G into $G[1..p]G[i..j]G[p+1..n]$.

An *event* is either a deletion or a duplication. If G is a genome and e is an event , we write $G\langle e \rangle$ to denote the genome obtained by applying e on G. Given a sequence $E = (e_1, \ldots, e_k)$ of events, we define $G\langle E \rangle = G\langle e_1 \rangle \langle e_2 \rangle \ldots \langle e_k \rangle$ as the genome obtained by successively applying the events of E to G. We may also write $G\langle e_1, \ldots, e_k \rangle$ instead of $G\langle (e_1, \ldots, e_k) \rangle$.

The most natural application of the above events is to compare genomes.

▶ **Definition 1.** *Let G and G' be two strings over alphabet Σ. The* Genome-to-Genome *distance between G and G', denoted $d_{GG}(G, G')$, is the size of the smallest sequence of events E satisfying $G\langle E \rangle = G'$.*

[1] Note that in the theory of formal languages, the CNP of a string is called the *Parikh vector*.

Note that d_{GG} has recently been considered in [11, 13], the latter in the case of tandem duplications only. We also define a distance between a genome G and a CNP $\vec{c}$, which is the minimum number of events to apply to G to obtain a genome with CNP $\vec{c}$.

▶ **Definition 2.** *Let G be a genome and $\vec{c}$ be a CNP, both over alphabet Σ. The* Genome-to-CNP *distance between G and $\vec{c}$, denoted $d_{GCNP}(G, \vec{c})$, is the size of the smallest sequence of events E satisfying $cnp(G\langle E\rangle) = \vec{c}$.*

The above definition leads to the following problem, which was first studied in [18].
The Minimum Copy Number Generation (MCNG) problem:
Instance: a genome G and a CNP $\vec{c}$ over alphabet Σ.
Task: compute $d_{GCNP}(G, \vec{c})$.

Qingge et al. proved that the MCNG problem is NP-hard when all the duplications are restricted to be tandem [18]. In the next section, we prove that this problem is not only NP-hard, but also NP-hard to approximate within any constant factor.

3 Hardness of Approximation for MCNG

In this section, we show that the d_{GCNP} distance is hard to approximate within any constant factor. This result actually holds if only deletions on G are allowed. This restriction makes the proof significantly simpler, so we first analyze the deletions-only case. We then extend this result to deletions *and* duplications.

Both proofs rely on a reduction from SET-COVER. Recall that in SET-COVER, we are given a collection of sets $\mathcal{S} = \{S_1, S_2, \ldots, S_n\}$ over universe $U = \{u_1, u_2, \ldots, u_m\} = \bigcup_{S_i \in \mathcal{S}} S_i$, and we are asked to find a set cover of $\mathcal{S}$ having minimum cardinality (a set cover of $\mathcal{S}$ is a subset $\mathcal{S}^* \subseteq \mathcal{S}$ such that $\bigcup_{S \in \mathcal{S}^*} S = U$). If $\mathcal{S}'$ is a set cover in which no two sets intersect, then $\mathcal{S}'$ is called an *exact cover*.

There is one interesting feature (or constraint) of our reduction g, which transforms a SET-COVER instance $\mathcal{S}$ into a MCNG instance $g(\mathcal{S})$. A set cover $\mathcal{S}^*$ only works on $g(\mathcal{S})$ if $\mathcal{S}^*$ is actually an exact cover, and a solution for $g(\mathcal{S})$ can be turned into a set cover for $\mathcal{S}^*$ that is not necessarily exact. Thus we are unable to reduce directly from either SET-COVER nor its exact version. We provide a general-purpose lemma for such situations, and our reductions serve as an example of its usefulness.

The proof relies on a result on t-SET-COVER, the special case of SET-COVER in which every given set contains at most t elements. It is known that for any constant $t \geq 3$, the t-SET-COVER problem is hard to approximate within a factor $\ln t - c \ln \ln t$ for some constant c not depending on t [23].

▶ **Lemma 3.** *Let $\mathcal{B}$ be a minimization problem, and let g be a function that transforms any SET-COVER instance $\mathcal{S}$ into an instance $g(\mathcal{S})$ of $\mathcal{B}$ in polynomial time. Assume that both the following statements hold:*

- *any exact cover $\mathcal{S}^*$ of $\mathcal{S}$ of cardinality at most k can be transformed in polynomial time into a solution of value at most k for $g(\mathcal{S})$;*
- *any solution of value at most k for $g(\mathcal{S})$ can be transformed in polynomial time into a set cover of $\mathcal{S}$ of cardinality at most k.*

Then unless $P = NP$, there is no constant factor approximation algorithm for $\mathcal{B}$.

Proof. Suppose for contradiction that $\mathcal{B}$ admits a factor b approximation for some constant b. Choose any constant t such that t-SET-COVER is hard to approximate within factor $\ln t - c \ln \ln t$, and such that $b < \ln t - c \ln \ln t$. Note that t might be exponentially larger than b, but is still a constant.

Now, let $\mathcal{S}$ be an instance of t-SET-COVER over the universe $U = \{u_1, \ldots, u_m\}$. Consider the intermediate reduction g' that transforms $\mathcal{S}$ into another t-SET-COVER instance $g'(\mathcal{S}) = \{S' \subseteq S : S \in \mathcal{S}\}$. Since t is a constant, $g(\mathcal{S})$ has $O(|\mathcal{S}|)$ sets and this can be carried out in polynomial time.

Now define $\mathcal{S}' = g'(\mathcal{S})$ and consider the instance $B = g(\mathcal{S}') = g(g'(\mathcal{S}))$. By the assumptions of the lemma, a solution for B of value k yields a set cover $\mathcal{S}^*$ for $\mathcal{S}'$. Clearly, $\mathcal{S}^*$ can be transformed into a set cover for instance $\mathcal{S}$: for each $S' \in \mathcal{S}^*$, there exists $S \in \mathcal{S}$ such that $S' \subseteq S$, so we get a set cover for $\mathcal{S}$ by adding this corresponding superset for each $S \in \mathcal{S}^*$. Thus B yields a set cover of $\mathcal{S}$ with at most k sets.

In the other direction, consider a set cover $\mathcal{S}^* = \{S_1, \ldots, S_k\}$ of $\mathcal{S}$ with k sets. This easily translates into an *exact* cover of $\mathcal{S}'$ with k sets by taking the collection

$$\{S_1, S_2 \setminus S_1, S_3 \setminus (S_1 \cup S_2), \ldots, S_k \setminus \bigcup_{i=1}^{k-1} S_i\}.$$

By the assumptions of the lemma, this exact cover can then be transformed into a solution of value at most k for instance B.

Therefore, $\mathcal{S}$ has a set cover of cardinality at most k if and only if B has a solution of value at most k. By this correspondence, a factor b approximation for $\mathcal{B}$ would provide a factor $b < \ln t - c \ln \ln t$ approximation for t-SET-COVER. ◀

3.1 Constructing genomes and CNPs from SET-COVER instances

All of our hardness results rely on Lemma 3. We need to provide a reduction from SET-COVER to MCNG and prove that both assumptions of the lemma are satisfied.

This reduction is the same for deletions-only and deletions-and-duplications. Given $\mathcal{S}$ and U, we construct a genome G and a CNP $\vec{c}$ as follows (an example is illustrated in Figure 2). The alphabet is $\Sigma = \Sigma_{\mathcal{S}} \cup \Sigma_U$, where $\Sigma_{\mathcal{S}} := \{\langle\beta_{S_i}\rangle : S_i \in \mathcal{S}\}$ and $\Sigma_U := \{\alpha_{u_i} : u_i \in U\}$. Thus, there is one character for each set of $\mathcal{S}$ and each element of U. Here, each $\langle\beta_{S_i}\rangle$ is a character that will serve as a separator between characters to delete. For a set $S_i \in \mathcal{S}$, define the string $q(S_i)$ as any string that contains each character of $\{\alpha_u : u \in S_i\}$ exactly once (in any fixed order, say by their indices). We put

$$G = \langle\beta_{S_1}\rangle q(S_1) \langle\beta_{S_2}\rangle q(S_2) \ldots \langle\beta_{S_n}\rangle q(S_n),$$

i.e. G is the concatenation of the strings $\langle\beta_{S_i}\rangle q(S_i)$. As for the CNP $\vec{c}$, put

- $\vec{c}(\langle\beta_{S_i}\rangle) = 1$ for each $S_i \in \mathcal{S}$;
- $\vec{c}(\alpha_u) = f(u) - 1$ for each $u \in U$, where $f(u) = |\{S_i \in \mathcal{S} : u \in S_i\}|$ is the number of sets from $\mathcal{S}$ that contain u.

Notice that in G, each $\langle\beta_S\rangle$ already has the correct copy-number, whereas each α_u needs exactly one less copy. Our goal is thus to reduce the number of each α_u by 1. This concludes the construction of MCNG instances from SET-COVER instances. We now focus on the hardness of the deletions-only case.

$$S_1 = \{1,2,3\} \quad S_2 = \{1,3,4\} \quad S_3 = \{2,3,5\}$$

$$G = \langle\beta_{S_1}\rangle\alpha_1\alpha_2\alpha_3\langle\beta_{S_2}\rangle\alpha_1\alpha_3\alpha_4\langle\beta_{S_3}\rangle\alpha_2\alpha_3\alpha_5$$

$$\vec{c}(\alpha_1) = \vec{c}(\alpha_2) = 1 \quad \vec{c}(\alpha_3) = 2 \quad \vec{c}(\alpha_4) = \vec{c}(\alpha_5) = 0$$

Figure 2 An example of our construction, with $\mathcal{S} = \{S_1, S_2, S_3\}$ and $U = \{1, 2, 3, 4, 5\}$.

3.2 Warmup: the deletions-only case

Suppose that we are given a set cover instance $\mathcal{S}$ and U, and let G and $\vec{c}$ be the genome and CNP, respectively, as constructed above.

▶ **Lemma 4.** *Given an exact cover $\mathcal{S}^*$ for $\mathcal{S}$ of cardinality k, one can obtain a sequence of k deletions transforming G into a genome with CNP $\vec{c}$.*

Proof. Denote $\mathcal{S}^* = \{S_{i_1}, \ldots, S_{i_k}\}$. Consider the sequence of k deletions that deletes the substrings $q(S_{i_1}), \ldots, q(S_{i_k})$ (i.e. the sequence first deletes the substring $q(S_{i_1})$, then deletes $q(S_{i_2})$, and so on until $q(S_{i_k})$ is deleted). Since $S_{i_1}, \ldots, S_{i_k}$ is an exact cover, this sequence removes exactly one copy of each $\alpha_u \in \Sigma_U$ and does not affect the $\langle\beta_S\rangle$ characters. Thus the k deletions transform G into a genome with the desired CNP $\vec{c}$. ◀

▶ **Lemma 5.** *Given a sequence of k deletions transforming G into a genome with CNP $\vec{c}$, one can obtain a set cover for $\mathcal{S}$ of cardinality at most k.*

Proof. Suppose that the deletion events $E = (e_1, \ldots, e_k)$ transform G into a genome G^* with CNP $\vec{c}$. Note that no e_i deletion is allowed to delete a set-character $\langle\beta_{S_i}\rangle \in \Sigma_{\mathcal{S}}$, as there is only one occurrence of $\langle\beta_{S_i}\rangle$ in G and $\vec{c}(\langle\beta_{S_i}\rangle) = 1$. Thus all deletions remove only α_u characters. In other words, each e_j in E either deletes a substring of G between some $\langle\beta_{S_i}\rangle$ and $\langle\beta_{S_{i+1}}\rangle$ with $1 \leq i < n$, or e_j deletes a substring after $\langle\beta_{S_n}\rangle$. Moreover, exactly one of each α_u occurrences gets deleted from G.

Call $\langle\beta_{S_i}\rangle \in \Sigma_{\mathcal{S}}$ *affected* if there is some event of E that deletes at least one character between $\langle\beta_{S_i}\rangle$ and $\langle\beta_{S_{i+1}}\rangle$ with $1 \leq i < n$, and call $\langle\beta_{S_n}\rangle$ affected if some event of E deletes characters after $\langle\beta_{S_n}\rangle$. Let $\mathcal{S}^* := \{S_i \in \mathcal{S} : \langle\beta_{S_i}\rangle \text{ is affected}\}$. Then $|\mathcal{S}^*| \leq k$, since each deletion affects at most one $\langle\beta_{S_i}\rangle$ and there are k deletion events. Moreover, $\mathcal{S}^*$ must be a set cover, because each $\alpha_u \in \Sigma_U$ has at least one occurrence that gets deleted and thus at least one set containing u that is included in $\mathcal{S}^*$. This concludes the proof. ◀

We have shown that all the assumptions required by Lemma 3 are satisfied. The inapproximability follows.

▶ **Theorem 6.** *Assuming $P \neq NP$, there is no polynomial-time constant factor approximation algorithm for MCNG when only deletions are allowed.*

We mention without proof that the reduction can be adaptable to the duplication-only case, by putting $\vec{c}(\alpha_u) = f(u) + 1$ for each $u \in U$.

The real deal: deletions and duplications

We now consider both deletions and duplications. The reduction uses the same construction as in Section 3.1. Thus we assume that we have a SET-COVER instance $\mathcal{S}$ over U, and a corresponding instance of MCNG with genome G and CNP $\vec{c}$. In that case, we observe that Lemma 4 still holds whether we allow deletion only, or both deletions and duplications. Thus we only need to show that the second assumption of Lemma 3 holds.

Unfortunately, this is not as simple as in the deletions-only case. The problem is that some duplications may copy some α_u and $\langle\beta_{S_i}\rangle$ occurrences, and we lose control over what gets deleted, and over what $\langle\beta_{S_i}\rangle$ each α_u corresponds to (in particular, some $\langle\beta_{S_i}\rangle$ might now get deleted, which did not occur in the deletions-only case).

Nevertheless, the analogous result can be shown. That is, using the above reduction, our goal is to show that, given a sequence of k events (deletions and duplications) transforming G into a genome with CNP $\vec{c}$, one can obtain a set cover for $\mathcal{S}$ of cardinality at most k.

We need some new notation and intermediate results beforehand. Let $E = (e_1, \ldots, e_k)$ be a sequence of events transforming genome G into another genome G'. We would like to distinguish each position of G in order to know which specific character of G is at the origin of a character of G'.

We augment each individual character of G with a unique identifier, which is its position in G. That is, let $G = g_1 g_2 \ldots g_n$, define a new alphabet $\hat{\Sigma} = (g_1^1, g_2^2, \ldots, g_n^n)$ and define the genome $\hat{G} = g_1^1 g_2^2 \ldots g_n^n$. Here, two characters g_i and g_j may be identical, but g_i^i and g_j^j are two distinct characters. We call $\hat{\Sigma}$ the *augmented alphabet* and $\hat{G}$ the *augmented genome* of G. For instance if $G = aabcb$ and $\Sigma = (a, b, c)$, then $\hat{\Sigma} = (a^1, a^2, b^3, c^4, b^5)$ and $\hat{G} = a^1 a^2 b^3 c^4 b^5$.

Since G and $\hat{G}$ have the same length, we may apply the sequence E on $\hat{G}$, resulting in a genome $\hat{G}' := \hat{G}\langle E\rangle$ on alphabet $\hat{\Sigma}$. Now $\hat{G}'$ may contain some characters of $\hat{\Sigma}$ multiple times owing to duplications, but if we remove the superscript identifier from the characters of $\hat{G}'$, we obtain G'. The idea is that the identifiers on the characters of $\hat{G}'$ tell us precisely where each character of $\hat{G}'$ "comes from" in $\hat{G}$ (and thus G).

▶ **Definition 7.** *Let G and G' be genomes and let E an event sequence such that $G' = G\langle E\rangle$. Let $\hat{G}$ be the augmented genome of G and let $\hat{G}[i] = g^i$ be the character at position i.*

If there is at least one occurrence of g^i in $\hat{G}\langle E\rangle$, then position i is called important *with respect to E. Otherwise, position i is called* unimportant *with respect to E.*

Roughly speaking, position i is unimportant if it eventually gets deleted, and any character that was copied from position i from a duplication also gets deleted, as well as a copy of this copy, and so on – in other words, position i has no "descendant" in G' when applying E.

First, we prove some general properties that will be useful. Recall that $G - s$ removes all occurrences of s from G, and $\vec{c} - s$ puts $\vec{c}(s) = 0$.

▶ **Proposition 8.** *Let G be a genome over alphabet Σ, let $\vec{c}$ be a CNP and let $s \in \Sigma$. Then $d_{GCNP}(G - s, \vec{c} - s) \leq d_{GCNP}(G, \vec{c})$.*

The next technical lemma states that if a genome alternates between positions to keep and positions to delete n times, then we need n events to remove the unimportant ones.

▶ **Lemma 9.** *Let $\Sigma = X \cup Y$ be an alphabet defined by two disjoint sets $X = \{x_1, \ldots, x_n\}$ and Y. Let $G = Y_0 x_1 Y_1 x_2 Y_2 \ldots x_n Y_n$ be a genome on Σ, where for all $i \in [n]$, Y_i is a non-empty string over alphabet Y and Y_0 is a possibly empty string on alphabet Y. Moreover let $\vec{c}$ be a CNP such that $\vec{c}(x_i) = 1$ for all $x_i \in X$ and $\vec{c}(y) = 0$ for all $y \in Y$. Then $d_{GCNP}(G, \vec{c}) \geq n$, with equality when Y_0 is empty.*

The proof is surprisingly technical and can be found in the full version. We may now prove the second assumption of Lemma 3.

▶ **Lemma 10.** *Let $\mathcal{S}$ be a SET-COVER instance, and let G and $\vec{c}$ be the corresponding MCNG instance. Given a sequence of k events (deletions and duplications) transforming G into a genome with CNP $\vec{c}$, one can obtain a set cover for $\mathcal{S}$ of cardinality at most k.*

Proof. Suppose that the events $E = (e_1, \ldots, e_k)$ transform G into a genome G^* with CNP $\vec{c}$. We construct a set cover for $\mathcal{S}$ of cardinality k. For a position p with $G[p] = \alpha_u \in \Sigma_U$, define $pred(p)$ as the first $\Sigma_{\mathcal{S}}$ character to the left of position p. To be precise, if p' is the largest integer satisfying $G[p'] \in \Sigma_{\mathcal{S}}$ and $p' < p$, then $pred(p) = G[p']$. Note that since $G[1] = \langle\beta_{S_1}\rangle$, $pred(p)$ is well-defined. Notice that by construction, if $G[p] = \alpha_u$ and $\langle\beta_S\rangle = pred(p)$, then $u \in S$. The set of $pred(p)$ of unimportant positions p will correspond to our set cover, which we now prove by separate claims.

▷ Claim 11. For each $u \in U$, there is at least one position p of G such that $G[p] = \alpha_u$ and such that p is unimportant w.r.t. E.

Proof. If we assume this is not the case, then each of the $f(u)$ positions p of G having $G[p] = \alpha_u$ has a descendant in G^*, implying that G^* has at least $f(u)$ copies of α_u and thereby contradicting that G^* complies with $\vec{c}(\alpha_u) = f(u) - 1$. ◁

Recall that $U = \{u_i, \ldots, u_m\}$. Given that the claim holds, let $P = \{p_1, \ldots, p_m\}$ be any set of positions of G such that for each $i \in [m]$, $G[p_i] = \alpha_{u_i}$ and p_i is unimportant w.r.t. E (choosing arbitrarily if there are multiple choices for p_i). Define $\Sigma_P = \{pred(p_i) : p_i \in P\}$ and $\mathcal{S}^* = \{S_i \in \mathcal{S} : \langle \beta_{S_i} \rangle \in \Sigma_P\}$.

▷ Claim 12. $\mathcal{S}^*$ is a set cover.

Proof. For each $u_i \in U$, there is an unimportant position $p_i \in P$ such that $G[p_i] = \alpha_{u_i}$. Moreover, $pred(p_i)$ is some character $\langle \beta_S \rangle$ such that $\langle \beta_S \rangle \in \Sigma_P$ and such that $u_i \in S$. Since $S \in \mathcal{S}^*$, it follows that each u_i is covered. ◁

It remains to show that $\mathcal{S}^*$ has at most k sets. Denote $P' = P \cup \{p : G[p] \in \Sigma_P\}$. Let $\tilde{G}$ be the subsequence of G obtained by keeping only positions in P' (i.e. if we denote $P' = \{p'_1, \ldots, p'_l\}$ with $p'_1 < p'_2 < \ldots < p'_l$, then $\tilde{G} = G[p'_1]G[p'_2] \ldots G[p'_l]$). Furthermore, define the CNP $\vec{c}_0$ such that $\vec{c}_0(\langle \beta_{S_i} \rangle) = 1$ for all $\langle \beta_{S_i} \rangle \in \Sigma_P$, $\vec{c}_0(\langle \beta_{S_i} \rangle) = 0$ for all $\langle \beta_{S_i} \rangle \in \Sigma_{\mathcal{S}} \setminus \Sigma_P$, and $\vec{c}_0(\alpha_u) = 0$ for all $\alpha_u \in \Sigma_U$. Note that $\tilde{G}$ has the form $\langle \beta_{S_{i_1}} \rangle D_1 \langle \beta_{S_{i_2}} \rangle D_2 \ldots \langle \beta_{S_{i_r}} \rangle D_r$ for some r, where the D_i's are substrings over alphabet Σ_U. This is form of Lemma 9.

▷ Claim 13. $d_{GCNP}(\tilde{G}, \vec{c}_0) \leq k$.

Proof. Let G' be the genome obtained by replacing every position p of G by some dummy character λ, except for the positions of P' (thus if we remove all the λ occurrences we obtain $\tilde{G}$). Since G and G' have the same length, we can apply the E events on G'. Let $G'' := G'\langle E \rangle$, and let l be the number of occurrences of λ in G''. Recall that P' contains only positions p such that $G[p] \in \Sigma_P$, or such that p is unimportant w.r.t E and $G[p] \in \Sigma_U$. It follows that if a position q is important w.r.t. E, then $G'[q] \in \Sigma_P \cup \{\lambda\}$. Moreover, for any $\langle \beta_S \rangle \in \Sigma_P$, G'' has as many occurrences of $\langle \beta_S \rangle$ as in $G\langle E \rangle$. In other words, G'' has one occurrence of each $\langle \beta_S \rangle \in \Sigma_P$ and the rest is filled with λ.

Let $\vec{c}_1$ be the CNP satisfying $\vec{c}_1(\lambda) = l$, $\vec{c}_1(\langle \beta_{S_i} \rangle) = \vec{c}_0(\langle \beta_{S_i} \rangle) = 1$ for every $\langle \beta_{S_i} \rangle \in \Sigma_P$, and $\vec{c}_1(x) = 0$ for any other character x. Then clearly, $\vec{c}_1 = cnp(G'')$, which implies $d_{GCNP}(G', \vec{c}_1) \leq k$ since E transforms G' into G''. Moreover by Proposition 8, $d_{GCNP}(G' - \lambda, \vec{c}_1 - \lambda) \leq d_{GCNP}(G', \vec{c}_1) \leq k$. The claim follows from the observation that $\tilde{G} = G' - \lambda$ and $\vec{c}_0 = \vec{c}_1 - \lambda$. ◁

Observe that $\tilde{G}$ and $\vec{c}_0$ have the required form for Lemma 9 (with $|\Sigma_P|$ important positions), and so $d_{GCNP}(\tilde{G}, \vec{c}_0) \geq |\Sigma_P|$. It follows from Claim 13 that $k \geq d_{GCNP}(\tilde{G}, \vec{c}_0) \geq |\Sigma_P| = |\mathcal{S}^*|$. We thus have a set cover $\mathcal{S}^*$ for $\mathcal{S}$ of cardinality at most k, completing the proof. ◀

We arrive to our main inapproximability result, which again follows from Lemma 3.

▶ **Theorem 14.** *Assuming $P \neq NP$, there is no polynomial-time constant factor approximation algorithm for MCNG.*

In the next section, we prove that the MCNG problem, parameterized by the solution size, is W[1]-hard. This answers another open question in [18]. We refer readers for more details on FPT and W[1]-hardness to the book by Downey and Fellows [7].

4 W[1]-hardness for MCNG

Since SET-COVER is W[2]-hard, naturally we would like to use the above reduction to prove the W[2]-hardness of MCNG. However, the fact that we use t-SET-COVER with constant t in the proof of Lemma 3 is crucial, and t-SET-COVER is in FPT. On the other hand, the property that is really needed in the instance of this proof, and in our MCNG reduction, is that we can transform any set cover instance into an exact cover. We capture this intuition and show that SET-COVER instances that have this property are W[1]-hard to solve.

An instance of SET-COVER-with-EXACT-COVER, or SET-COVER-EC for short, is a pair $I = (\mathcal{S}, k)$ where k is an integer and $\mathcal{S}$ is a collection of sets forming a universe U. In this problem, we require that $\mathcal{S}$ satisfies the property that *any* set cover for $\mathcal{S}$ of size at most k is also an exact cover. We are asked whether there exists a set cover for $\mathcal{S}$ of size at most k (in which case this set cover is also an exact cover).

▶ **Lemma 15.** *The SET-COVER-EC problem is W[1]-hard for parameter k.*

Proof. We show W[1]-hardness using the techniques introduced by Fellows et al. which is coined as MULTICOLORED-CLIQUE [10]. In the MULTICOLORED-CLIQUE problem, we are given a graph G, an integer k and a coloring $c : V(G) \to [k]$ such that no two vertices of the same color share an edge. We are asked whether G contains a clique of k vertices (noting that such a clique must have a vertex of each color). This problem is W[1]-hard w.r.t. k.

Given an instance (G, k, c) of MULTICOLORED-CLIQUE, we construct an instance $I = (\mathcal{S}, k')$ of SET-COVER-EC. We put $k' = k + \binom{k}{2}$. For $i \in [k]$, let $V_i = \{v \in V(G) : c(v) = i\}$ and for each pair $i < j \in [k]$, let $E_{ij} = \{uv \in E(G) : u \in V_i, v \in V_j\}$. The universe U of the SET-COVER-EC instance has one element for each color i, one element for each pair $\{i, j\}$ of distinct colors, and two elements for each edge, one for each direction of the edge. That is,

$$U = [k] \cup \binom{[k]}{2} \cup \{(u, v) \in V(G) \times V(G) : uv \in E(G)\}$$

Thus $|U| = k + \binom{k}{2} + 2|E(G)|$. For two colors $i < j \in [k]$, we will denote $U_{ij} = \{(u, v), (v, u) : u \in V_i, v \in V_j, uv \in E_{ij}\}$, i.e. we include in U_{ij} both elements corresponding to each $uv \in E_{ij}$. Now, for each color class $i \in [k]$ and each vertex $u \in V_i$, add to $\mathcal{S}$ the set

$$S_u = \{i\} \cup \{(u, v) : v \in N(u)\}$$

where $N(u)$ is the set of neighbors of u in G. Then for each $i < j \in [k]$, and for each edge $uv \in E_{ij}$, add to $\mathcal{S}$ the set

$$S_{uv} = \{\{i, j\}\} \cup \{(x, y) \in U_{ij} : x \notin \{u, v\}\}$$

The idea is that S_{uv} can cover every element of U_{ij}, except those ordered pairs whose first element is u or v. Then if we do decide to include S_{uv} in a set cover, it turns out that we will need to include S_u and S_v to cover these missing ordered pairs. See Figure 3 for an example. For instance if we include S_{u_2,v_3} in a cover, the uncovered (u_2, v_3) and (v_3, u_2) can be covered with S_{u_2} and S_{v_3}. We show that G has a multicolored clique of size k if and only if $\mathcal{S}$ admits a set cover of size k'. Note that we have not shown yet that $(\mathcal{S}, k')$ is an instance of SET-COVER-EC, i.e. that any set cover of size at most k' is also an exact cover. This will be a later part of the proof.

First suppose that G has a multi-colored clique $C = \{v_1, \ldots, v_k\}$, where $v_i \in V_i$ for each $i \in [k]$. Consider the collection

$$\mathcal{S}^* = \{S_{v_1}, \ldots, S_{v_k}\} \cup \{S_{v_i v_j} : v_i, v_j \in C, 1 \leq i < j \leq k\},$$

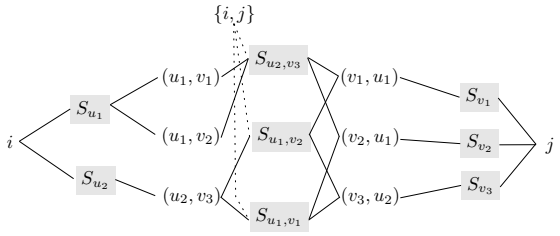

Figure 3 A graphical example of the constructed sets for the U_{ij} elements of a graph (not shown) with $E_{ij} = \{u_1v_1, u_1v_2, u_2v_3\}$, where the u_l's are in V_i and the v_l's in V_j (sets have a gray background, edges represent containment, the $\{i, j\}$ lines are dotted only for better visualization).

the cardinality of $\mathcal{S}^*$ is $k + \binom{k}{2} = k'$. Each element $i \in U \cap [k]$ is covered since we include a set S_{v_i} for each color. Each element $\{i, j\} \in U \cap \binom{[k]}{2}$ is covered since we include a set $S_{v_iv_j}$ for each color pair i, j with $i < j$. Consider an element $(x_i, y_j) \in U \cap (V(G) \times V(G))$, where $x_i \in V_i$ and $y_i \in V_j$. Note that either $i < j$ or $j < i$ is possible, and that $v_iv_j \in E(G)$. If $x_i \notin \{v_i, v_j\}$, then $S_{v_iv_j}$ covers (x_i, y_j). If $x_i = v_i$, then S_{v_i} covers (x_i, v_j) and if $x_i = v_j$, then S_{v_j} covers (x_i, v_j). Thus $\mathcal{S}^*$ is a set cover, and is of size at most k'.

For the converse direction, suppose that $\mathcal{S}^*$ is a set cover for $\mathcal{S}$ of size at most $k' = k + \binom{k}{2}$. Note that to cover the elements of $U \cap [k]$, $\mathcal{S}^*$ must have at least one set S_u such that $u \in V_i$ for each color class $i \in [k]$. Moreover, to cover the elements of $U \cap \binom{[k]}{2}$, $\mathcal{S}^*$ must have at least one set S_{uv} such that $u \in V_i, v \in V_j$ for each $i, j \in [k]$ pair. We deduce that $\mathcal{S}^*$ has exactly $k + \binom{k}{2}$ sets. Hence for color $i \in [k]$, there is *exactly* one set S_u in $\mathcal{S}^*$ for which $u \in V_i$, and for each $\{i, j\}$ pair, there is *exactly* one S_{uv} set in $\mathcal{S}^*$ for which $u \in V_i, v \in V_j$.

We claim that $C = \{u : S_u \in \mathcal{S}^*\}$ is a multi-colored clique. We already know that C contains one vertex of each color. Now, suppose that some $u, v \in C$ do not share an edge, where $u \in V_i, v \in V_j$ and $i < j$. Let S_{xy} be the set of $\mathcal{S}^*$ that covers $\{i, j\}$, with $x \in V_i, y \in V_j$. Since uv is not an edge but xy is, we know that $u \neq x$ or $v \neq y$ (or both). Moreover, S_{xy} does not cover the (x, y) and (y, x) elements of U_{ij}, and we know that at least one of these is not covered by S_u nor S_v (if $u \neq x$, then none covers (x, y), if $v \neq y$, then none covers (y, x)). But $(x, y) \in U_{ij}$, and S_u, S_v and S_{xy} are the only sets of $\mathcal{S}^*$ that have elements of U_{ij}, contradicting that $\mathcal{S}^*$ is a set cover. This shows that C is a multi-colored clique.

It remains to show that $\mathcal{S}^*$ is an exact cover. Observe that no two distinct S_u and S_v sets in $\mathcal{S}^*$ can intersect because u and v must be of a different color, and no two distinct S_{uv} and S_{xy} sets in $\mathcal{S}^*$ can intersect because $\{u, v\}$ and $\{x, y\}$ must be from two different color pairs. Suppose that $S_u, S_{xy} \in \mathcal{S}^*$ do intersect, and say that $x \in V_i, y \in V_j$ and $i < j$. Then all elements in $S_u \cap S_{xy}$ are of the form (u, v) for some v. Choose any such (u, v). If u is of color i, then $u \neq x$ since otherwise by construction S_{xy} could not contain (u, v). But when $u \neq x$, no set of $\mathcal{S}^*$ covers the element (x, y) (it is not S_u nor S_{xy}, the only two possibilities). If u is of color j, then $u \neq y$ since again S_{xy} could not contain (u, v). In this case, no set of $\mathcal{S}^*$ covers (y, x). We reach a contradiction and deduce that $\mathcal{S}^*$ is an exact cover. ◀

It is now almost immediate that MCNG is W[1]-hard with respect to the natural parameter, namely the number of events to transform a genome G into a genome with a given profile $\vec{c}$ (the detailed proof can be found in the full version).

▶ **Theorem 16.** *The MCNG problem is W[1]-hard.*

We do not know whether SET-COVER-EC or MCNG are also in W[1], i.e. whether they are W[1]-complete. We have finished presenting the negative results on MCNG. An

immediate question is whether we could obtain some positive result on a related problem. In the next section, we present some positive result for an interesting variation of MCNG.

5 The Copy Number Profile Conforming Problem

We define the more general *Copy Number Profile Conforming (CNPC)* problem as follows:

▶ **Definition 17.** *Given two CNP's* $\vec{c}_1 = \langle u_1, u_2, ..., u_n \rangle$ *and* $\vec{c}_2 = \langle v_1, v_2, ..., v_n \rangle$*, with* $u_i, v_i \geq 0$ *and* $u_i, v_i \in \mathbb{N}$*, the CNPC problem asks to compute two strings* S_1 *and* S_2 *with* $cnp(S_1) = \vec{c}_1$ *and* $cnp(S_2) = \vec{c}_2$ *such that the distance between* S_1 *and* S_2*,* $d(S_1, S_2)$*, is minimized.*

Let $\sum_i u_i = m_1, \sum_i v_i = m_2$, we assume that m_1 and m_2 are bounded by a polynomial of n. (This assumption is needed as the solution of our algorithm could be of size $\max\{m_1, n_2\}$.) We simply say $\vec{c}_1, \vec{c}_2$ are polynomially bounded. Note that $d(S_1, S_2)$ is a very general distance measure, i.e., it could be any genome rearrangement distance (like reversal, transposition, and tandem duplication, etc, or their combinations, e.g. tandem duplication + deletion). In this paper, we use the breakpoint distance and the adjacency number. Our definitions of these notions are adapted from Angibaud et al. [1] and Jiang et al. [12], which generalize the corresponding concepts on permutations [24].

Given two sequences $A{=}a_1a_2\cdots a_n$ and $B{=}b_1b_2\cdots b_m$, if $\{a_i, a_{i+1}\} = \{b_j, b_{j+1}\}$ we say that a_ia_{i+1} and b_jb_{j+1} are matched to each other (in the graph theory terminology, they share an edge). Consider a maximum cardinality matching between length 2 substrings of A and B. A matched pair is called an *adjacency*, and an unmatched pair is called a *breakpoint* in A and B respectively. Then, the multiset of 2-substrings of A (resp. B) that belong to a breakpoint is denoted as $b_A(A, B)$ (resp. $d_B(A, B)$) and the corresponding number is $d_b(A, B)$ (resp. $d_b(B, A)$), and the number of common adjacencies between A and B is denoted as $a(A, B)$. Note that $d_b(A, B), d_b(B, A)$ and $a(A, B)$ do not depend on a particular choice of maximum matching. We illustrate the above definitions in Fig. 4.

sequence A	$=$	$\langle a\ c\ b\ d\ c\ b\ \rangle$
sequence B	$=$	$\langle a\ b\ c\ d\ a\ b\ c\ d\ \rangle$
matched pairs	$:$	$(cb \leftrightarrow bc), (dc \leftrightarrow cd), (cb \leftrightarrow bc)$
$a(A, B)$	$=$	$\{bc, bc, cd\}$
$b_A(A, B)$	$=$	$\{ac, bd\}$
$b_B(A, B)$	$=$	$\{ab, da, ab, cd\}$

Figure 4 Example for adjacency and breakpoint definitions, with $d_b(A, B) = 2$ and $d_b(B, A) = 4$.

Coming back to our problem, we define $d(S_1, S_2) = d_b(S_1, S_2) + d_b(S_2, S_1)$. From the definitions, we have

$$d_b(S_1, S_2) + d_b(S_2, S_1) + 2 \cdot a(S_1, S_2) = (m_1 - 1) + (m_2 - 1),$$

or,

$$d_b(S_1, S_2) + d_b(S_2, S_1) = m_1 + m_2 - 2 \cdot a(S_1, S_2) - 2.$$

Hence, the problem is really to maximize $a(S_1, S_2)$.

▶ **Definition 18.** *Given n-dimensional vectors* $\vec{u} = \langle u_1, u_2, ..., u_n \rangle$ *and* $\vec{w} = \langle w_1, w_2, ..., w_n \rangle$, *with* $u_i, w_i \geq 0$, *and* $u_i, w_i \in \mathbb{N}$, *we say* $\vec{w}$ *is a* sub-vector *of* $\vec{u}$ *if* $w_i \leq u_i$ *for* $i = 1, ..., n$, *also denote this relation as* $\vec{w} \leq \vec{u}$.

Henceforth, we simply call $\vec{u}, \vec{w}$ integer vectors, with the understanding that no item in a vector is negative.

▶ **Definition 19.** *Given two n-dimensional integer vectors* $\vec{u} = \langle u_1, u_2, ..., u_n \rangle$ *and* $\vec{v} = \langle v_1, v_2, ..., v_n \rangle$, *we say* $\vec{w}$ *is a* common sub-vector *of* $\vec{u}$ *and* $\vec{v}$ *if* $\vec{w}$ *is a sub-vector of* $\vec{u}$ *and* $\vec{w}$ *is also a sub-vector of* $\vec{v}$ *(i.e.,* $\vec{w} \leq \vec{u}$ *and* $\vec{w} \leq \vec{v}$*). Finally,* $\vec{w}$ *is the* maximum common sub-vector *of* $\vec{u}$ *and* $\vec{v}$ *if there is no common sub-vector* $\vec{w}' \neq \vec{w}$ *of* $\vec{u}$ *and* $\vec{v}$ *which satisfies* $\vec{w} \leq \vec{w}' \leq \vec{u}$ *or* $\vec{w} \leq \vec{w}' \leq \vec{v}$.

An example is illustrated as follows. We have $\vec{u} = \langle 3, 2, 1, 0, 5 \rangle$, $\vec{v} = \langle 2, 1, 3, 1, 4 \rangle$, $w' = \langle 2, 1, 0, 0, 3 \rangle$ and $\vec{w} = \langle 2, 1, 1, 0, 4 \rangle$. Both $\vec{w}$ and $\vec{w}'$ are common sub-vectors for $\vec{u}$ and $\vec{v}$, $\vec{w}'$ is not the maximum common sub-vector of $\vec{u}$ and $\vec{v}$ (since $\vec{w}' \leq \vec{w}$) while $\vec{w}$ is.

Given a CNP $\vec{u} = \langle u_1, u_2, ..., u_n \rangle$ and alphabet $\Sigma = (x_1, x_2, ..., x_n)$, for $i \in \{1, 2, ..., n\}$, we use $S(\vec{u})$ to denote the multiset of letters (genes) corresponding to $\vec{u}$; more precisely, u_i denotes the number of x_i's in $S(\vec{u})$. Similarly, given a multiset of letters Z, we use $s(Z)$ to denote a string where all the letters in Z appear exactly once (counting multiplicity; i.e, $|Z| = |s(Z)|$). $s(Z)$ is similarly defined when Z is a CNP. We present **Algorithm 1**:.

1. Compute the maximum common sub-vector $\vec{v}$ of $\vec{c}_1$ and $\vec{c}_2$.
2. Given the gene alphabet Σ, compute $S(\vec{v})$, $S(\vec{c}_1)$ and $S(\vec{c}_2)$. Let $X = S(\vec{c}_1) - S(\vec{v})$ and $Y = S(\vec{c}_2) - S(\vec{v})$.
3. If $S(\vec{v}) = \emptyset$, then return two arbitrary strings $s(\vec{c}_1)$ and $s(\vec{c}_2)$ as S_1 and S_2, exit; otherwise, continue.
4. Find $\{x, y\}$, $x, y \in \Sigma$ and $x \neq y$, such that $x \in S(\vec{v})$ and $y \in S(\vec{v})$, and exactly one of x, y is in X (say $x \in X$), and the other is in Y (say $y \in Y$). If such an $\{x, y\}$ cannot be found then return two strings S_1 and S_2 by concatenating letters in X and Y arbitrarily at the ends of $s(\vec{v})$ respectively, exit; otherwise, continue.
5. Compute an arbitrary sequence $s(\vec{v})$ with the constraint that the first letter is x and the last letter is y. Then obtain $s_1 = s(\vec{v}) \circ x$ and $s_2 = y \circ s(\vec{v})$ ($\circ$ denotes concatenation).
6. Finally, insert all the elements in $X - \{x\}$ arbitrarily at the two ends of s_1 to obtain S_1, and insert all the elements in $Y - \{y\}$ arbitrarily at the two ends of s_2 to obtain S_2.
7. Return S_1 and S_2.

Let $\Sigma = \{a, b, c, d, e\}$. Also let $\vec{c}_1 = \langle 2, 2, 2, 4, 1 \rangle$ and $\vec{c}_2 = \langle 4, 4, 1, 1, 1 \rangle$. We walk through the algorithm using this input as follows.

1. The maximum common sub-vector $\vec{v}$ of $\vec{c}_1$ and $\vec{c}_2$ is $\vec{v} = \langle 2, 2, 1, 1, 1 \rangle$.
2. Compute $S(\vec{v}) = \{a, a, b, b, c, d, e\}$, $S(\vec{c}_1) = \{a, a, b, b, c, c, d, d, d, d, e\}$ and $S(\vec{c}_2) = \{a, a, a, a, b, b, b, b, c, d, e\}$. Compute $X = \{c, d, d, d\}$ and $Y = \{a, a, b, b\}$.
3. Identify d and a such that $d \in S(\vec{v})$ and $a \in S(\vec{v})$, and $d \in X$ while $a \in Y$.
4. Compute $s(\vec{v}) = dabbcea$, $s_1 = dabbcea \cdot d$ and $s_2 = a \cdot dabbcea$.
5. Insert elements in $X - \{d\} = \{c, d, d\}$ arbitrarily at the right end of s_1 to obtain S_1, and insert all the elements in $Y - \{a\} = \{a, b, b\}$ at the right end of s_2 to obtain S_2.
6. Return $S_1 = dabbcea \cdot d \cdot cdd$ and $S_2 = a \cdot dabbcea \cdot abb$.

▶ **Theorem 20.** *Let* $\vec{c}_1, \vec{c}_2$ *be polynomially bounded. The number of common adjacencies generated by Algorithm 1 is optimal with a value either* n^* *or* $n^* - 1$, *where* $n^* = \sum_{i=1}^{n} v_i$ *with the maximum common sub-vector of* $\vec{c}_1$ *and* $\vec{c}_2$ *being* $\vec{v} = \langle v_1, v_2, ..., v_n \rangle$.

Proof. First, note that if $\vec{v}$ is a 0-vector (or $S(\vec{v}) = \emptyset$) then there will not be any adjacency in S_1 and S_2. Henceforth we discuss $S(\vec{v}) \neq \emptyset$.

Notice that a common adjacency between S_1 and S_2 must come from two letters which are both in $S(\vec{v})$. That naturally gives us $n^* - 1$ adjacencies, where $n^* = |S(\vec{v})|$, which can be done by using the letters in $S(\vec{v})$ to form two arbitrary strings S_1 and S_2 (for which $s(\vec{v})$ is a common substring). If $\{x, y\}$ can be found such that $x, y \in S(\vec{v})$ and $x \neq y$, and one of them is in X (say $x \in X$), and the other is in Y (say $y \in Y$), then, obviously we could obtain $s_1 = s(\vec{v}) \circ x$ and $s_2 = y \circ s(\vec{v})$ which are substrings of S_1 and S_2 respectively. Clearly, there are $n^* = |S(\vec{v})|$ adjacencies between s_1 and s_2 (and also S_1 and S_2).

To see that this is optimal, first suppose that no $\{x, y\}$ pair as above can be found. This can only occur when there are no two components $i < j$ in $\vec{c}_1 = \langle c_{1,1}, ..., c_{1,i}, ..., c_{1,j}, ..., c_{1,n}\rangle$, $\vec{c}_2 = \langle c_{2,1}, ..., c_{2,i}, ..., c_{2,j}, ..., c_{2,n}\rangle$, and in the maximum common sub-vector $\vec{v} = \langle v_1, ..., v_i, ..., v_j, ..., v_n\rangle$ of $\vec{c}_1$ and $\vec{c}_2$ which satisfy that $\min\{c_{1,i}, c_{2,i}\} = v_i \neq 0$ and $\max\{c_{1,i}, c_{2,i}\} \neq v_i$, and $\min\{c_{1,j}, c_{2,j}\} = v_j \neq 0$ and $\max\{c_{1,j}, c_{2,j}\} \neq v_j$. If this condition holds, then all the components i in $s(\vec{c}_1 - \vec{v})$ and $s(\vec{c}_2 - \vec{v})$, i.e., $c_{1,i} - v_i$ and $c_{2,i} - v_i$, have the property that at least one of the two is zero and $v_i = 0$. Therefore, except for the letters corresponding to $\vec{v}$, no other adjacency can be formed. As any string with CNP $\vec{v}$ has n^* characters, at most $n^* - 1$ adjacencies can be formed. If an $\{x, y\}$ pair can be found, let $b \in \Sigma$, and let v_b be the minimum copy-number of b in $\vec{c}_1$ or $\vec{c}_2$, i.e., $v_b = \min\{c_{1,b}, c_{2,b}\}$. Assume this minimum occurs in $\vec{c}_1$, w.l.o.g. There can be at most $2v_b$ adjacencies involving b in $\vec{c}_1$, and thus at most $2v_b$ adjacencies in common involving v_b. Summing over every $b \in \Sigma$, the sum of common adjacencies, counted for each character individually, is at most $\sum_{b\in\Sigma} 2v_b = 2n^*$. Since each adjacency is counted twice in this sum, the number of common adjacencies is at most n^*. ◀

Note that if we only want the breakpoint distance between S_1 and S_2, then the polynomial boundness condition of $\vec{c}_1$ and $\vec{c}_2$ can be withdrawn as we can decide whether $\{x, y\}$ exists by searching directly in the CNPs (vectors).

6 Concluding Remarks

In this paper, we answered two recent open questions regarding the computational complexity of the Minimum Copy Number Generation problem. Our technique could be used for other optimization problems where the solution involves Set Cover whose solution must also be an exact cover. We also present a polynomial time algorithm for the Copy Number Profile Conforming (CNPC) problem when the distance is the classical breakpoint distance. The breakpoint distance is static, and we leave open the question for solving or approximating CNPC with dynamic rearrangement distance such as reversal, duplication+deletion, etc.

References

1 Sebastien Angibaud, Guillaume Fertin, Irena Rusu, Annelyse Thevenin, and Stephane Vialette. On the approximability of comparing genomes with duplicates. *J. Graph Algorithms and Applications*, 13(1):19–53, 2009.

2 Salim Chowdhury, Stanley Shackney, Kerstin Heselmeyer-Haddad, Thomas Ried, Alejandro Shaeffer, and Russell Schwartz. Algorithms to model single gene, single chromosome, and whole genome copy number changes jointly in tumor phylogenetics. *PLOS Computational Biology*, 10(7), 2014.

3 SL Cooke, J Temple, S Macarthur, MA Zahra, LT Tan, RAF Crawford, CKY Ng, M Jimenez-Linan, E Sala, and JD Brenton. Intra-tumour genetic heterogeneity and poor chemoradiotherapy response in cervical cancer. *British Journal of Cancer*, 104(2):361, 2011.

4 Susanna L Cooke and James D Brenton. Evolution of platinum resistance in high-grade serous ovarian cancer. *The Lancet Oncology*, 12(12):1169–1174, 2011.

5 Garance Cordonnier and Manuel Lafond. Comparing copy-number profiles under multi-copy amplifications and deletions. *BMC genomics*, 21(2):1–12, 2020.

6 Prue A Cowin, Joshy George, Sian Fereday, Elizabeth Loehrer, Peter Van Loo, Carleen Cullinane, Dariush Etemadmoghadam, Sarah Ftouni, Laura Galletta, Michael S Anglesio, et al. Lrp1b deletion in high-grade serous ovarian cancers is associated with acquired chemotherapy resistance to liposomal doxorubicin. *Cancer Research*, 72(16):4060–4073, 2012.

7 Rodney Downey and Michael Fellows. *Parameterized complexity*. Springer Science & Business Media, 2012.

8 Mohammed El-Kebir, Benjamin Raphael, Ron Shamir, Roded Sharan, Simone Zaccaria, Meirav Zehavi, and Ron Zeira. Copy-number evolutions: complexity and algorithms. In *Proceedings of WABI'2016, LNCS*, volume 9838, pages 137–149. Springer, 2016.

9 Mohammed El-Kebir, Benjamin J Raphael, Ron Shamir, Roded Sharan, Simone Zaccaria, Meirav Zehavi, and Ron Zeira. Complexity and algorithms for copy-number evolution problems. *Algorithms for Molecular Biology*, 12(1):13, 2017.

10 Michael Fellows, Danny Hermelin, Frances Rosamond, and Stephane Vialette. On the parameterized complexity of multiple-interval graph problems. *Theoretical Computer Science*, 410(1):53–61, 2009.

11 Patrick Holloway, Krister Swenson, David Ardell, and Nadia El-Mabrouk. Ancestral genome organization: an alignment approach. *Journal of Computational Biology*, 20(4):280–295, 2013.

12 Haitao Jiang, Chunfang Zheng, David Sankodd, and Binhai Zhu. Scaffold filling under the breakpoint and related distances. *IEEE/ACM Trans. Bioinformatics and Comput. Biology*, 9(4):1220–1229, 2012.

13 Manuel Lafond, Binhai Zhu, and Peng Zou. The tandem duplication distance is np-hard. In *Proceedings of STACS'2020, LIPIcs*, volume 154, pages 15:1–15:15. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2020.

14 Carlo C Maley, Patricia C Galipeau, Jennifer C Finley, V Jon Wongsurawat, Xiaohong Li, Carissa A Sanchez, Thomas G Paulson, Patricia L Blount, Rosa-Ana Risques, Peter S Rabinovitch, et al. Genetic clonal diversity predicts progression to esophageal adenocarcinoma. *Nature Genetics*, 38(4):468, 2006.

15 Andriy Marusyk, Vanessa Almendro, and Kornelia Polyak. Intra-tumour heterogeneity: a looking glass for cancer? *Nature Reviews Cancer*, 12(5):323, 2012.

16 Nicholas Navin, Alexander Krasnitz, Linda Rodgers, Kerry Cook, Jennifer Meth, Jude Kendall, Michael Riggs, Yvonne Eberling, Jennifer Troge, Vladimir Grubor, et al. Inferring tumor progression from genomic heterogeneity. *Genome Research*, 20(1):68–80, 2010.

17 Cancer Genome Atlas Research Network et al. Integrated genomic analyses of ovarian carcinoma. *Nature*, 474(7353):609, 2011.

18 Letu Qingge, Xiaozhou He, Zhihui Liu, and Binhai Zhu. On the minimum copy number generation problem in cancer genomics. In *Proceedings of ACM BCB'2018*, pages 260–269. ACM, 2018.

19 Gryte Satas, Simone Zaccaria, Geoffrey Mon, and Benjamin J Raphael. Scarlet: Single-cell tumor phylogeny inference with copy-number constrained mutation losses. *Cell Systems*, 10(4):323–332, 2020.

20 Roland F Schwarz, Anne Trinh, Botond Sipos, James D Brenton, Nick Goldman, and Florian Markowetz. Phylogenetic quantification of intra-tumour heterogeneity. *PLoS Computational Biology*, 10(4):e1003535, 2014.

21 Sohrab P Shah, Ryan D Morin, Jaswinder Khattra, Leah Prentice, Trevor Pugh, Angela Burleigh, Allen Delaney, Karen Gelmon, Ryan Guliany, Janine Senz, et al. Mutational evolution in a lobular breast tumour profiled at single nucleotide resolution. *Nature*, 461(7265):809, 2009.

22 Ron Shamir, Meirav Zehavi, and Ron Zeira. A linear-time algorithm for the copy number transformation problem. In *Proceedings of CPM'2016, LIPIcs*, volume 54, pages 16:1–16:13. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2016.

23 Luca Trevisan. Non-approximability results for optimization problems on bounded degree instances. In *Proceedings of 33rd ACM Symp. on Theory of Comput. (STOC'01)*, pages 453–461. ACM, 2001.

24 G.A. Watterson, W.J. Ewens, T.E. Hall, and A. Morgan. The chromosome inversion problem. *J. Theoretical Biology*, 99(1):1–7, 1982.

25 Ruofan Xia, Yu Lin, Jun Zhou, Tieming Geng, Bing Feng, and Jijun Tang. Phylogenetic reconstruction for copy-number evolution problems. *IEEE/ACM transactions on computational biology and bioinformatics*, 16(2):694–699, 2018.

26 Simone Zaccaria, Mohammed El-Kebir, Gunnar W Klau, and Benjamin J Raphael. Phylogenetic copy-number factorization of multiple tumor samples. *Journal of Computational Biology*, 25(7):689–708, 2018.