Algorithms for the uniqueness of the longest common subsequence

Yue Wang[1,*]

[1]Department of Computational Medicine, University of California,
Los Angeles, California, United States of America
[*]E-mail address: yuew@g.ucla.edu. ORCID: 0000-0001-5918-7525

Abstract

Given several number sequences, determining the longest common subsequence is a classical problem in computer science. This problem has applications in bioinformatics, especially determining transposable genes. Nevertheless, related works only consider how to find one longest common subsequence. In this paper, we consider how to determine the uniqueness of the longest common subsequence. If there are multiple longest common subsequences, we also determine which number appears in all/some/none of the longest common subsequences. We focus on four scenarios: (1) linear sequences without duplicated numbers; (2) circular sequences without duplicated numbers; (3) linear sequences with duplicated numbers; (4) circular sequences with duplicated numbers. We develop corresponding algorithms and apply them to gene sequencing data.

KEY WORDS: longest common subsequence, algorithm, graph, transposable gene

1 Introduction

Given some number sequences, a common subsequence is a number sequence which appears in all these sequences (not necessarily consecutive). Determining the longest common subsequence (LCS) for some number sequences is a classical problem in computer science. LCS is a common tool to evaluate the difference among different sequences. For example, LCS can be applied to computational linguistics [39, 61, 60]. In biology, it is common to use the length of LCS as a quantitative score for comparing DNA sequences [12, 26, 84]. LCS has also been used to define ultraconserved elements [55] or remove incongruent markers in DNA sequences [16].

Various scenarios for the LCS problem have been studied. Here we list Scenarios A-E, where the first two are more commonly studied. For more works in these scenarios, readers may refer to more thorough reviews [5, 24, 77].

Scenario A considers two sequences with possibly repeated numbers, and the sequence length is n. The goal is to find the LCS. If a number appears multiple time in a common subsequence, all appearances are counted when calculating the length of this common subsequence. This can be solved by dynamic programming with $\mathcal{O}(n^2)$ time complexity and $\mathcal{O}(n)$ space complexity [23], but $\mathcal{O}(n^{2-\epsilon})$ time complexity for any $\epsilon > 0$ is impossible [4]. This also can be solved with $o(n)$ space complexity and $\mathcal{O}(n^3)$ time complexity [35].

In Scenario B, there are m sequences with possibly repeated numbers, and the sequence length is n. The goal is to find the LCS. If a number appears multiple time in a common subsequence, all appearances are counted when calculating the length of this common subsequence. A standard dynamic programming algorithm has $\mathcal{O}(n^m)$ time complexity [7]. There have been other faster algorithms [67, 45, 27]. This scenario is equivalent to the maximum clique problem in graph theory, which is NP-hard [40], but has relatively fast exact and heuristic algorithms [30, 37, 72].

Scenario C considers 2 sequences with possibly repeated numbers, and the sequence length is n. The goal is to find the LCS, where each number appears at most once. This scenario is NP-hard [1].

Scenario D is similar to Scenario B, but only consider common subsequences that contain or do not contain certain strings [69, 47].

In Scenario E, the sequences are arc-annotated, and LCS should have the same arc annotation in original sequences [31].

In this paper, the motivation of studying the LCS problem is to apply it to compare gene sequences. Assume we have some gene sequences from different individuals of the same species or different species. Some genes are relatively unstable, and they can change their relative locations in the gene sequence (transposable). An unstable gene might also be duplicated or deleted. Therefore, these gene sequences from different individuals are not identical. Then we can find the LCS, which is useful for measuring the stability of genes. Genes in the LCS should be more stable, and genes not in the LCS should be transposable.

Due to the motivation of comparing gene sequences, we consider four scenarios that are different from the previously studied LCS problems. These four scenarios are determined by two factors: whether the considered species has linear or circular gene sequences, and whether genes have multiple copies. When genes have multiple copies, we only consider common subsequences that consist of all or none of copies of the same gene. Scenario 1 has linear sequences without duplicated genes; Scenario 2 has circular sequences without duplicated genes; Scenario 3 has linear sequences with duplicated genes; Scenario 4 has circular sequences with duplicated genes.

Most known methods only aim at finding one LCS. Since we concern the stability of genes, the uniqueness of LCS should be determined. When the LCS is not unique, we also need to classify whether a gene appears in all/some/none of the LCSs. A gene that appears in all the LCSs is highly stable; a gene that appears in some LCSs is moderately stable; a gene that appears in no LCS is unstable. Determining all LCSs is too time-consuming, since there might be exponentially many LCSs. For example, consider two sequences $(1, 2, 3, 4, 5, 6, \ldots, 2n-1, 2n)$ and $(2, 1, 4, 3, 6, 5, \ldots, 2n, 2n-1)$. Although the sequence length is $2n$, and the LCS length is n, the number of LCSs is 2^n. To determine the relationship between genes and LCSs, we develop corresponding algorithms with polynomial time complexities for Scenarios 1, 2 (Algorithms 2, 4). To our knowledge, there are no other determinations of whether genes appear in all LCSs with polynomial complexities. Scenarios 3, 4 only consider subsequences that consist of all or none copies of the same gene, and calculate the length by genes. Therefore, they are different from the classic Scenario B. We develop the equivalence of Scenario 3 with the maximum clique problems on graphs (Proposition 1). We prove that Scenario 4 is between the maximum clique problems on graphs and the maximum clique problems on 3-uniform hypergraphs (Propositions 2, 3). Although circular sequences are commonly studied in the context of genomic rearrangements, they are rare in the literature of the LCS problems. Therefore, our Algorithm 3 that finds one LCS for Scenario 2 should also be novel. We test Algorithms 1, 2, 3, 4 on the gene sequences of different *Escherichia coli* individuals and find some possible transposable genes.

If we only need to find one LCS, then Scenario 1 is a special case of Scenario B, and our method (Algorithm 1) can be easily derived from standard algorithms. Scenarios 3, 4 are equivalent to maximum clique problems in graphs and hypergraphs, which are NP-hard. These properties are also similar to Scenario B. Although there have been numerous algorithms for the maximum clique problem [78], for the sake of completeness, we design fast heuristic algorithms (Algorithms 5, 6) and test them to find that they only fail in rare cases.

We proposed the idea of using the LCS to find transposable genes and Algorithm 1 in a previous paper [32], where Algorithm 1 was applied to study the "core-gene-defined genome organizational framework" (the complement of transposable genes) in various bacteria, and it was found that for different species, the transposable gene distribution and developmental traits are correlated. This paper considers other situations (especially when the LCS is not unique), and can be regarded as a theoretical sequel of that previous paper. Algorithm 1 is contained in this paper for the sake of completeness.

In sum, our main contributions are Algorithms 2, 3, 4 in Scenarios 1, 2 and Propositions 1, 2, 3 in Scenarios 3, 4.

We first introduce the background of transposable genes in Section 2. Then we describe the setup for the LCS problem we study in Section 3. In Sections 4–7, we transform the LCS problem into corresponding graph theory problems and design algorithms. We finish with some discussions in Section 8 and conclusions in Section 9. All the algorithms in this paper have been implemented in Python. For the code and data files, see https://github.com/YueWangMathbio/Transposon.

2 Biological background of transposable genes

In this section, we review how gene sequences become different, and introduce the specific biological problem we want to study. We also explain how Scenarios 1–4 of the LCS problem are derived from the biological problem.

The nucleotide sequence can be changed by various events, such as inversion, insertion, deletion, and duplication [28]. Such rearrangement events lead to the existence of transposons (also called transposable elements or jumping genes), which are DNA sequences that can change their relative positions within the genome. Transposons were first discovered in maize by Barbara McClintock [42]. Transposons have various types: long terminal repeats (LTR) retrotransposons, Dictyostelium intermediate repeat sequence (DIRS)-like elements, Penelope-like elements (PLE), long interspersed elements (LINE), short interspersed elements (SINE), terminal inverted repeats (TIR), Helitrons, etc. [41].

Transposons are common in various species. For the human genome, the proportion of transposons is approximately 44%, although most of transposons are inactive [43]. Transposons can participate in controlling gene expression [83], and they are related to several diseases, such as cancer [13], hemophilia [33], and porphyria [46]. Transposons can drive rapid phenotypic variations, which cause complicated cell behaviors [81, 49, 48, 11, 29].

Transposons can be used to detect cancer drivers [50] and potential therapies [2]. Transposons are also essential for the development of *Oxytricha trifallax* [51], antibiotic resistance of bacteria [3], and the proliferation of various cells [54, 79, 14]. With the presence of transposons, the regulation between genes might be affected, which is a challenge for inferring the structures of gene regulatory networks [75] and general transcriptome analysis [59, 82].

When transposons have been determined, we can use them to compare the genomes of different species, and such comparisons can be combined with other measurements between species, such as metrics on developmental trees [71]. Such comparisons can be also extended to different tissues to help with the prediction of tissue transplantation experiments [76]. Besides, for some species, cells at different positions have different gene expression patterns, which might be related to transposons [73].

Many transposons are as short as $10^2 - 10^3$ base pairs, shorter than a general gene [53]. To determine such short transposons, one needs to analyze the original AGCT nucleotide sequences. There have been many algorithms developed to determine short transposons from nucleotide sequences, such as MELT (Mobile Element Locator Tool) [18], ERVcaller (Endogenous Retro-Virus caller) [10], and TEMP2 (Transposable Elements Movements Present 2) [80]. Different algorithms may only determine certain types of transposons. For more details, readers may refer to other papers [52, 20]. They use raw DNA sequencing data, which only contain imperfect information about the true DNA sequence, and the data quality depends on some factors that vary across different datasets [17]. Besides, they need a corresponding genome or reference transposon libraries.

There are gross DNA changes that associate with many genes, also called genomic rearrangements [21]. Such rearrangements include inversion, transposition, fusion, and fission [8]. To determine such gross genomic rearrangements, one first needs to convert nucleotide sequences into gene sequences by annotation. For two different gene sequences, the general idea of determining rearrangements is to calculate the minimal number of operations required for transforming one sequence into the other [63]. This defines an editing distance between gene sequences, which can be used to compare the evolution distance between species and construct the phylogenetic tree [62]. There have been many algorithms developed to determine genomic rearrangements. They consider different scenarios: whether the gene sequence is linear or circular, whether genes have unique labels, and what operations can be taken. Kececioglu and Sankoff only consider inversion for linear sequences with unique gene labels [34]; Blanchette et al. consider inversion and transposition for circular sequences with unique gene labels [6];

Tesler considers inversion, transposition, fusion, and fission for linear and circular sequences with unique gene labels [63]; Terauds and Sumner study circular sequences with representation theory tools [62]; Bohnenkämper et al. consider linear and circular sequences with possibly duplicated labels [8]. There are also systematic pipelines for determining rearrangements from whole-genome assemblies [19, 44]. Nevertheless, these methods consider large-scale rearrangements, and minimize the number of operations to transform one gene sequence into the other, not concrete genes that can change their locations. Besides, these methods only compare two gene sequences, not more. Their results depend on the set of possible operations, which is somewhat arbitrary.

In this paper, we consider a mesoscopic scenario between the genomic rearrangement situation and the short transposon situation: *Given accurately annotated gene sequences (not nucleotide sequences) from different individuals, determine individual genes (not short nucleotide segments or long gene strands) that can change their locations (transposable).* This provides a qualitative description for the stability of genes, which can guide gene editing [68] and phylogenetics [32]. The proportion of fixed genes quantifies the robustness of the genome. We aim at minimizing the number of genes to move. When there are only two gene sequences, this is equivalent to calculating genomic arrangements, where the only allowed operation is single-gene transposition.

In the copy-paste (duplication) case and deletion case, we can compare the numbers of copies of genes for different individuals to determine the transposable genes that have changed their copy numbers. In the inversion case, we can check the direction of genes to determine transposable genes that have changed their orientations [38]. In the cut-paste (insertion) case, the compositions of gene sequences are the same, but the orders of genes differ. It is not straightforward to uniquely determine which genes have changed their relative locations. In this case, we need to introduce the LCS problem.

3 Problem setup

Given raw DNA sequencing data, the first step is to transform them into gene sequences. This can be done with various genome annotation tools [58, 9]. For simplicity, we replace the gene names by numbers $1, \ldots, n$.

For some species, the DNA is a line [57]. We can represent this DNA as a linear gene sequence of distinct numbers that represent genes: $(1, 2, 3, 4)$. If some genes change their transcriptional orientations, we can simply detect them and handle the remaining genes. Now a linear DNA naturally has a direction (from 5' end to 3' end), thus $(1,2,3,4)$ and $(4,3,2,1)$ are two different gene sequences.

Consider two linear gene sequences from different individuals: $(1,2,3,4)$ and $(1,4,2,3)$. We can intuitively detect that gene 4 changes its relative position, and should be regarded as a transposable gene. However, changing the positions of genes $2,3$ can also transform one sequence into the other. The reason that we think gene 4 (not genes $2,3$) changes its relative position is that the number of genes we need to move is smaller. Nevertheless, the number of genes that change their relative locations is difficult to determine. We can consider the complement of transposable genes, i.e., genes that do not change their relative positions. These fixed genes can be easily defined as the LCS of the given gene sequences. Here a common subsequence consists of some genes (not necessarily adjacent, different from a substring) that keep their relative orders in the original sequences. *Thus transposable genes are the complement of this LCS.* Notice that the LCS might not be unique. We classify genes by their relations with the LCS(s). The motivation of classifying transposable genes with respect to the intersection and union of LCSs is similar to defining essential variables with Markov boundaries in causal inference [74].

Definition 1. *A gene is **proper-transposable** if it is not contained in any LCS. A gene is **non-transposable** if it is contained in every LCS. A gene is **quasi-transposable** if it is contained in some but not all LCSs.*

In the example of $(1,2,3,4)$ and $(1,4,2,3)$, the unique LCS is $(1,2,3)$. Thus 4 is proper-transposable, and $1,2,3$ are non-transposable. In the following, we consider other scenarios, where the proper/quasi/non-transposable genes still follow Definition 1, but the definition of the LCS differs.

For some species, the DNA is a circle, not a line [66]. A circular DNA also has a natural direction (from 5' end to 3' end), and we use the clockwise direction to represent this natural direction. In the circular sequence scenario, a common subsequence is a circular sequence that can be obtained from each circular gene sequence by deleting some genes. See Fig. 1 for two circular gene sequences and their LCS. Notice that we can rotate each circular sequence for a better match.

A gene might have multiple copies (duplicated) in a gene sequence [25]. Notice that the definition of the transposable gene is a gene (specific DNA sequence) that has the ability to change its position, not a certain copy of a gene that changes its position. This means transposable genes should be

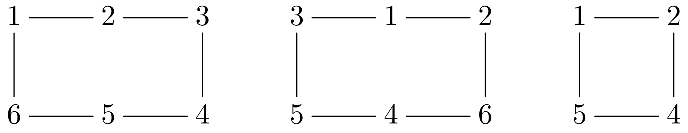

Figure 1: Two circular gene sequences without duplicated genes and their LCS, corresponding to Scenario 2.

defined for genes, not gene copies. Thus we should only consider common subsequences that consist of all or none copies of the same gene. When calculating the length of a common subsequence, we should count genes, not gene copies. Consider two linear sequences $(4, 1, 2, 1, 1, 3, 2, 4, 1, 1)$ and $(4, 1, 2, 3, 1, 1, 2, 1, 1, 4)$. If we consider any subsequences, the LCS is $(4, 1, 2, 1, 1, 2, 1, 1)$; if we only consider subsequences that contain all or none copies of the same gene, but count the length by copies, the LCS is $(1, 2, 1, 1, 2, 1, 1)$; if we only consider subsequences that contain all or none copies of the same gene, and count the length by genes, the unique LCS is $(4, 2, 3, 2, 4)$, and gene 1 is proper-transposable.

When we consider circular gene sequences with duplicated genes, we should still only consider subsequences that consist of all or none copies of the same gene, and calculate the length by genes. Notice that circular sequences can be rotated. See Fig. 2 for two circular gene sequences with duplicated genes and their LCS.

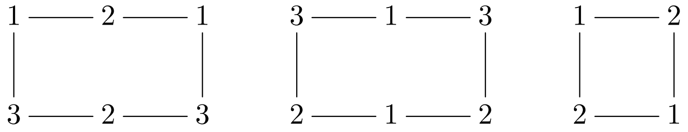

Figure 2: Two circular gene sequences with duplicated genes and their LCS, corresponding to Scenario 4.

We have turned the problem of determining transposable genes into finding the LCS of several gene sequences. Depending on whether the gene sequences are linear or circular, and whether genes have multiple copies, the problem can be classified into four scenarios:

Scenario 1: Consider m linear sequences of genes $1, \ldots, n$, where each gene has only one copy in each sequence. Determine the longest linear sequence that is a common subsequence of these m sequences.

Scenario 2: Consider m circular sequences of genes $1, \dots, n$, where each gene has only one copy in each sequence. Determine the longest circular sequence that is a common subsequence of these m sequences. Here circular sequences can be rotated.

Scenario 3: Consider m linear sequences of genes $1, \dots, n$, where each gene can have multiple copies in each sequence. Determine the longest linear sequence that is a common subsequence of these m sequences. Only consider subsequences that consist of all or none copies of the same gene, and calculate the length by genes.

Scenario 4: Consider m circular sequences of genes $1, \dots, n$, where each gene can have multiple copies in each sequence. Determine the longest circular sequence that is a common subsequence of these m sequences. Only consider subsequences that consist of all or none copies of the same gene, and calculate the length by genes. Here circular sequences can be rotated.

These four scenarios correspond to different algorithms, and will be discussed separately.

4 Linear sequences without duplicated genes

In Scenario 1, consider m linear gene sequences, where each sequence contains n genes $1, \dots, n$. Each gene has only one copy. For such permutations of $1, \dots, n$, we need to find the LCS.

4.1 A graph representation of the problem

Brute-force searching that tests whether each subsequence appears in all sequences is not applicable, since the time complexity is exponential in n. To develop a polynomial algorithm, we first design an auxiliary directed graph $\mathcal{G}$.

Definition 2. *For m linear sequences with n non-duplicated genes, the corresponding **auxiliary graph** $\mathcal{G}$ is a directed graph, where each vertex is a gene g_i, and there is a directed edge from g_i to g_j if and only if g_i appears before g_j in all m sequences.*

A directed path $g_1 \to g_2 \to g_3 \to \cdots \to g_4 \to g_5$ in $\mathcal{G}$ corresponds to a common subsequence $(g_1, g_2, g_3, \dots, g_4, g_5)$ of m sequences, and vice versa. We add 0 to the head of each sequence and $n+1$ to the tail. Then the LCS must start at 0 and end at $n+1$. *The problem of finding the LCS becomes finding the longest path from 0 to $n+1$ in $\mathcal{G}$.* See Fig. 3 for an example of using the auxiliary graph to determine transposable genes. This auxiliary graph $\mathcal{G}$ has no directed loop (acyclic). If there exists a loop $g_1 \to g_2 \to g_3 \to \cdots \to g_4 \to g_1$, then g_1 is prior to g_4 and g_4 is prior to g_1 in all sequences, a contradiction.

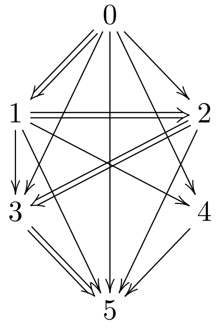

Figure 3: The auxiliary graph $\mathcal{G}$ of two sequences $([0], 1, 2, 3, 4, [5])$ and $([0], 1, 4, 2, 3, [5])$. The unique longest path (double arrows) from 0 to 5 is $0 \to 1 \to 2 \to 3 \to 5$, meaning that the unique longest common sequence is $([0], 1, 2, 3, [5])$. Thus $1, 2, 3$ are non-transposable, and 4 is proper-transposable.

4.2 Find the longest path

Determining the longest path between two vertices in a directed acyclic graph can be solved by a standard dynamic programming algorithm. For a vertex $g_i \in \{0, 1, \ldots, n\}$, consider the longest path from g_i to $n+1$. Since there exists an edge $g_i \to n+1$, and $\mathcal{G}$ is acyclic, this longest path exists. If the longest path is not unique, assign one arbitrarily.

Definition 3. *Define $F_+(g_i)$ to be the length of the longest path from g_i to $n+1$ in $\mathcal{G}$, and $H_+(g_i)$ to be the vertex next to g_i in this path.*

F_+ and H_+ can be calculated recursively: For one gene g_i, consider all genes g_j with an edge $g_i \to g_j$ in $\mathcal{G}$. The gene g_j with the largest $F_+(g_j)$ is assigned to be $H_+(g_i)$, and $F_+(g_i) = F_+(g_j) + 1$. If $g_l \to n+1$ is the only edge that starts from gene g_l, then $F_+(g_l) = 1$, and $H_+(g_l) = n+1$. In other words,

$$H_+(g_i) = \underset{\{g_j \text{ with } g_i \to g_j\}}{\operatorname{argmax}} F_+(g_j);$$

$$F_+(g_i) = 1 + F_+[H_+(g_i)].$$

Then $0 \to H_+(0) \to H_+^2(0) \to H_+^3(0) \to \cdots \to H_+^{f-1}(0) \to H_+^f(0) = n+1$, denoted by $\mathcal{L}_0$, is a longest path in $\mathcal{G}$. Here $f = F_+(0)$, and H_+^i is the ith iteration of H_+.

4.3 Test the uniqueness of the longest path

To test whether quasi-transposable genes exist, we need to check the uniqueness of this longest path.

Definition 4. *For $g_i \in \{1, \ldots, n, n+1\}$, define $F_-(g_i)$ to be the length of the longest path from 0 to g_i in $\mathcal{G}$, and $H_-(g_i)$ to be the vertex prior to g_i in this path.*

F_- and H_- can be calculated similar to F_+ and H_+. We can see that $F_+(g_i) + F_-(g_i)$ is the length of

$$0 = H_-^{F_-(g_i)}(g_i) \to H_-^{F_-(g_i)-1}(g_i) \to \cdots \to H_-(g_i) \to g_i$$

$$\to H_+(g_i) \to \cdots \to H_+^{F_+(g_i)-1}(g_i) \to H_+^{F_+(g_i)}(g_i) = n+1,$$

a longest path from 0 through g_i to $n+1$. For $g_i \notin \mathcal{L}_0$, if $F_+(g_i) + F_-(g_i) < F_+(0)$, then g_i is proper-transposable; if $F_+(g_i) + F_-(g_i) = F_+(0)$, then g_i is quasi-transposable. If every $g_i \notin \mathcal{L}_0$ is proper-transposable, then the LCS is unique, and all genes in $\mathcal{L}_0$ (excluding the auxiliary 0 and $n+1$) are non-transposable. The procedure of determining transposable genes stops here. Otherwise, the LCS is not unique, and we need to find quasi-transposable genes in $\mathcal{L}_0$.

4.4 Find quasi-transposable genes

When determining all quasi-transposable genes $g_1, \ldots, g_k$ not in $\mathcal{L}_0$, as described above, we construct corresponding longest paths $\mathcal{L}_1, \ldots, \mathcal{L}_k$ from 0 to $n+1$, where each $\mathcal{L}_i$ passes through g_i. We claim that a gene $g_j \in \mathcal{L}_0$ is non-transposable if and only if g_j is contained in all $\mathcal{L}_1, \ldots, \mathcal{L}_k$. To prove this, we need the following lemma.

Lemma 1. *In Scenario 1 of linear sequences without duplicated genes, each quasi-transposable gene g_i has a corresponding quasi-transposable gene g_j, so that no LCS can contain both g_i and g_j.*

If a gene $g_j \in \mathcal{L}_0$ is non-transposable, then it is contained in all $\mathcal{L}_1, \ldots, \mathcal{L}_k$. If $g_j \in \mathcal{L}_0$ is quasi-transposable, by Lemma 1, there is a quasi-transposable gene $g_l \notin \mathcal{L}_0$ which is mutual-exclusive with g_j, in the sense that g_l and g_j

cannot appear in the same LCS. The corresponding longest path $\mathcal{L}_l$ contains g_l, thus cannot contain g_j. This proves our approach to determine the quasi-transposable genes in $\mathcal{L}_0$.

Proof of Lemma 1. Fix a quasi-transposable gene g_i. It is contained in a longest path $\mathcal{L}_i$, which contains all non-transposable genes. Thus for each non-transposable gene g^*, there is an edge between g^* and g_i in $\mathcal{G}$. Assume g_i has no such mutual-exclusive quasi-transposable gene g_j. Then there is an edge (direction unknown) in $\mathcal{G}$ between g_i and each quasi-transposable gene g_j. Choose a longest path $\mathcal{L}^*$ in $\mathcal{G}$ that does not contain g_i. Whether $g_j \in \mathcal{L}^*$ is a non-transposable gene or a quasi-transposable gene, there is an edge between g_j and g_i. Determine the first gene g_k in $\mathcal{L}^*$ that has an edge $g_i \to g_k$. Since there is an edge $g_i \to n+1$, g_k exists. Since there is an edge $0 \to g_i$, $g_k \neq 0$. Denote the previous gene of g_k in $\mathcal{L}^*$ by g_l, then g_l exists, and there is an edge $g_l \to g_i$. Thus we construct a path $0 \to \cdots \to g_l \to g_i \to g_k \to \cdots \to n+1$, which is longer than the longest path, a contradiction. Thus g_i has a mutual-exclusive quasi-transposable gene g_j. □

4.5 Algorithms and complexities

We summarize the above method as Algorithms 1,2. If we have known that the LCS is unique, then we just need to apply Algorithm 1, so that genes in $\mathcal{L}_0$ are non-transposable, and genes not in $\mathcal{L}_0$ are proper-transposable. We have reported Algorithm 1 previously [32, 70]. Algorithm 1 is kept here to make the story complete. Assume we have m sequences with length n, and the length of the LCS is $n-k$. The time complexities of Steps 2-5 in Algorithm 1 are $\mathcal{O}(m)$, $\mathcal{O}(mn^2)$, $\mathcal{O}(n)$, $\mathcal{O}(n)$. The time complexities of Step 2 and Step 3 in Algorithm 2 are $\mathcal{O}(k)$ and $\mathcal{O}(kn)$. Since $k \leq n$, the overall time complexity of determining transposable genes in Scenario 1 by Algorithms 1,2 is $\mathcal{O}(mn^2)$. The space complexity is trivially $\mathcal{O}(mn+n^2)$.

4.6 Applications on experimental data

We test Algorithms 1,2 on *Escherichia coli* gene sequences. From NCBI sequencing database, we obtain gene sequences of three individuals of *E. coli* strain ST540 (GenBank CP007265.1, GenBank CP007390.1, GenBank CP007391.1) and three individuals of *E. coli* strain ST2747 (GenBank CP007392.1, GenBank CP007393.1, GenBank CP007394.1).

All three sequences of ST540 start with gene dnaA and end with gene rpmH. We can regard them as linear gene sequences. We remove genes that

```
1. Input
     $m$ linear sequences of genes $1, \dots, n$. No duplicated genes.
2. Modify the sequences:
     Add 0 to the head, and $n+1$ to the tail of each sequence
3. Construct the auxiliary graph $\mathcal{G}$:
     Vertices of $\mathcal{G}$ are all the genes $1, \dots, n$
     For each pair of genes $g_i, g_j$
       If $g_i$ is prior to $g_j$ in all $m$ sequences
         Add a directed edge $g_i \to g_j$ in $\mathcal{G}$
       End of if
     End of for
4. Calculate $F_+(\cdot)$ and $H_+(\cdot)$ for each gene $g_i$ in $0, 1, \dots, n$ recursively;
   calculate $F_-(\cdot)$ and $H_-(\cdot)$ for each gene $g_i$ in $1, \dots, n, n+1$
   recursively:
     $H_+(g_i) = \underset{\{g_j \text{ with } g_i \to g_j\}}{\operatorname{argmax}} F_+(g_j)$
     % If $g_j$ with $g_i \to g_j$ that maximizes $F_+(g_j)$ is not unique, choose
   one randomly
     $F_+(g_i) = 1 + F_+[H_+(g_i)]$
     $H_-(g_i) = \underset{\{g_j \text{ with } g_j \to g_i\}}{\operatorname{argmax}} F_-(g_j)$
     % If argmax is not unique, choose one randomly
     $F_-(g_i) = 1 + F_-[H_-(g_i)]$
5. Construct a longest path $\mathcal{L}_0$ from 0 to $n+1$:

   $0 \to H_+(0) \to H_+^2(0) \to H_+^3(0) \to \cdots \to H_+^{f-1}(0) \to H_+^f(0) = n+1$
     % Here $f = F_+(0)$, and $H_+^i$ is the $i$th iteration of $H_+$
6. Output $F_+(\cdot), H_+(\cdot), F_-(\cdot), H_-(\cdot), \mathcal{L}_0$
```

Algorithm 1: Detailed workflow of determining proper-transposable genes and quasi-transposable genes in Scenario 1, preparation stage.

```
1. Input
     $F_+(\cdot), H_+(\cdot), F_-(\cdot), H_-(\cdot), \mathcal{L}_0$ calculated from Algorithm 1
     Denote all genes not in $\mathcal{L}_0$ by $g_1, \ldots, g_k$

2. For each gene $g_i$ in $g_1, \ldots, g_k$
     If $F_+(g_i) + F_-(g_i) < F_+(0)$
       Output $g_i$ is a proper-transposable gene
     Else
       Output $g_i$ is a quasi-transposable gene
     End of if
   End of for

3. If all genes in $g_1, \ldots, g_k$ are proper-transposable
     Output all genes in $\mathcal{L}_0$ are non-transposable
   Else
     For each gene $g_i$ in $g_1, \ldots, g_k$
       Use $H_+(\cdot)$ and $H_-(\cdot)$ to construct $\mathcal{L}_i$, a longest path from 0 to $n+1$ that passes $g_i$.
     End of for
     For each gene $g_j$ in $\mathcal{L}_0$ (excluding auxiliary 0 and $n+1$)
       If $g_j$ is contained in all $\mathcal{L}_1, \ldots, \mathcal{L}_k$
         Output $g_j$ is non-transposable
       Else
         Output $g_j$ is quasi-transposable
       End of if
     End of for
   End of if

4. Output: whether each gene is proper/quasi/non-transposable
```

Algorithm 2: Detailed workflow of determining proper-transposable genes and quasi-transposable genes in Scenario 1, output stage.

appear more than once in one sequence, and remove genes that do not appear in all three sequences. After applying Algorithms 1,2 on these three sequences, there are 301 non-transposable genes, 4 quasi-transposable genes (hpaC, iraD, fbpC, psiB), and 263 proper-transposable genes. The reason for the large amount of proper-transposable genes is that sequence CP007265.1 is significantly different from the other two. After removing it and applying Algorithms 1,2 to the remaining two sequences (CP007390.1 and CP007391.1), there are 564 non-transposable genes and 4 quasi-transposable genes (hpaC, iraD, fbpC, psiB). Therefore, some genes in hpaC, iraD, fbpC, psiB are likely to translocate.

All three sequences of ST2747 start with gene glnG and end with gene hemG. We can regard them as linear gene sequences. We remove genes that appear more than once in one sequence, and remove genes that do not appear in all three sequences. After applying Algorithms 1,2 on these three sequences, all 573 genes are non-transposable.

5 Circular sequences without duplicated genes

In Scenario 2, consider m circular gene sequences, where each sequence contains n genes $1, \dots, n$. Each gene has only one copy in each sequence. For such circular permutations of $1, \dots, n$, we need to find the LCS. Assume the length of the LCS is $n - k$.

5.1 Find one LCS

We first randomly choose a gene g_i. Cut all circular sequences at g_i and expand them to be linear sequences. For example, the circular sequences in Fig. 1 cut at 1 are correspondingly $(1, 2, 3, 4, 5, 6)$ and $(1, 2, 6, 4, 5, 3)$. Using Algorithm 1, we can find $\mathcal{L}_i$ that begins with g_i, which is one LCS of all expanded linear sequences. In the above example, the longest common linear subsequence starting from 1 is $(1, 2, 4, 5)$. If g_i is a non-transposable gene or a quasi-transposable gene, then $\mathcal{L}_i$ (glued back to a circle) is a longest common circular subsequence. If g_i is a proper-transposable gene, then $\mathcal{L}_i$ is shorter than the longest common circular subsequence. In Fig. 1, gene 1 is non-transposable, and $(1, 2, 4, 5)$ (glued) is the longest common circular subsequence.

We do not know if $\mathcal{L}_i$ (glued) is an LCS (whether containing g_i or not) for all circular sequences. If there is a longer common subsequence, it should contain genes that are not in $\mathcal{L}_i$. Consider four variables $\mathcal{L}$, g, C, and $\mathcal{S}$, whose initial values are $\mathcal{L}_i$, g_i, the length of $\mathcal{L}_i$, and the complement of $\mathcal{L}_i$. These variables contain information on the longest common linear subsequence that we have found during this procedure.

Choose a gene g_j in $\mathcal{S}$, and cut all circular gene sequences at g_j. Apply Algorithm 1 to find $\mathcal{L}_j$, which is the longest in common subsequences that contain g_j. If the length of $\mathcal{L}_j$ is larger than C, set $\mathcal{L}$ to be $\mathcal{L}_j$, set g to be g_j, set C to be the length of $\mathcal{L}_j$, and set $\mathcal{S}$ to be the complement of $\mathcal{L}_j$. Otherwise, keep $\mathcal{L}$, g, C, and $\mathcal{S}$ still.

Choose another gene g_l in $\mathcal{S}$ which has not been chosen before, and repeat this procedure. This procedure terminates when all genes in $\mathcal{S}$ have been chosen and cut. Denote the final values of $\mathcal{L}$, g, C, and $\mathcal{S}$ by $\mathcal{L}_0$, g_0, C_0, and $\mathcal{S}_0$. Here $\mathcal{S}_0$ is the complement of $\mathcal{L}_0$.

During this procedure, if the current g is a proper-transposable gene, then $\mathcal{S}$ contains a non-transposable gene or a quasi-transposable gene, which has not been chosen. Thus $\mathcal{L}$, g, C, $\mathcal{S}$ will be further updated. If the current g is a non-transposable gene or a quasi-transposable gene, then C has reached its maximum, and $\mathcal{L}$, g, C, $\mathcal{S}$ will not be further updated. This means $\mathcal{L}_0$ is a longest common circular subsequence, and C_0 is the length of the LCS, $n-k$. Also, the total number of genes being chosen and cut is $k+1$. All k genes in $\mathcal{S}_0$ and g_0 are chosen and cut. A gene g_t in $\mathcal{L}_0$ (excluding g_0) is non-transposable or quasi-transposable, and cannot be chosen and cut. The reason is that it cannot be chosen before g_0 is chosen (only proper-transposable genes can be chosen before g_0 is chosen), and it cannot be chosen after g_0 is chosen ($g_t \notin \mathcal{S}_0$).

5.2 Determine quasi-transposable genes

For each gene $g_p \in \mathcal{S}_0$, apply Algorithm 1 to calculate C_p, the length of the LCS that contains g_p. If $C_p < C_0$, g_p is a proper-transposable gene. Otherwise, $C_p = C_0$ means g_p is a quasi-transposable gene. We have found all proper-transposable genes. If all genes in $\mathcal{S}_0$ are proper-transposable, then all genes in $\mathcal{L}_0$ are non-transposable, and the procedure terminates.

If $\mathcal{S}_0$ contains quasi-transposable genes, then $\mathcal{L}_0$ also has quasi-transposable genes. To determine quasi-transposable genes in $\mathcal{L}_0$, we need the following lemma.

Lemma 2. *In Scenario 2, choose a quasi-transposable gene g_p and cut the circular sequences at g_p to obtain linear sequences. A proper-transposable gene for the circular sequences is also a proper-transposable gene for the linear sequences; a non-transposable gene for the circular sequences is also a non-transposable gene for the linear sequences.*

Proof. Consider an LCS $\mathcal{L}_p$ for linear sequences cut at g_p. Since g_p is a quasi-transposable gene, the length of $\mathcal{L}_p$ is also $n - k$, meaning that $\mathcal{L}_p$ is also an LCS for circular sequences. Now, this lemma is proved by the definition of proper/quasi/non-transposable gene. □

If a gene g_r in $\mathcal{L}_0$ is non-transposable for the circular sequences, then g_r is a non-transposable gene for linear sequences cut at each quasi-transposable gene $g_q \in \mathcal{S}_0$. If a gene g_s in $\mathcal{L}_0$ is quasi-transposable for the circular sequences, then there is a longest common circular subsequence $\mathcal{L}_t$ that does not contain g_s, meaning that $\mathcal{L}_t$ contains a quasi-transposable gene g_t not in $\mathcal{L}_0$. Then g_s is a proper/quasi-transposable gene for linear sequences cut at g_t.

Therefore, we can use the following method to determine quasi-transposable genes in $\mathcal{L}_0$. For each quasi-transposable gene $g_q \in \mathcal{S}_0$, cut at g_q and apply Algorithms 1,2 to determine if each gene in $\mathcal{L}_0$ is proper/quasi/non-transposable for the linear gene sequences cut at g_q. A gene $g_r \in \mathcal{L}_0$ is non-transposable for the circular sequences if and only if it is non-transposable for linear sequences cut at any quasi-transposable gene $g_q \in \mathcal{S}_0$. A gene $g_s \in \mathcal{L}_0$ is quasi-transposable for the circular sequences if and only if it is proper/quasi-transposable for linear sequences cut at some quasi-transposable gene $g_q \in \mathcal{S}_0$.

When we have determined all quasi-transposable genes in $\mathcal{S}_0$, it might be tempting to apply a simpler approach to determine quasi-transposable genes in $\mathcal{L}_0$: For each quasi-transposable gene $g_q \in \mathcal{S}_0$, cut at g_q and apply Algorithm 1 to find an LCS $\mathcal{L}_q$. A gene in $\mathcal{L}_0$ is non-transposable if and only if it appears in all such $\mathcal{L}_q$. This approach is valid only if the following conjecture holds, which is similar to Lemma 1:

Conjecture 1. *In Scenario 2 of circular sequences without duplicated genes, each quasi-transposable gene g_i has a corresponding quasi-transposable gene g_j, so that no LCS can contain both g_i and g_j.*

However, Conjecture 1 does not hold. See Fig. 4 for a counterexample. All genes are quasi-transposable. Any two quasi-transposable genes are contained in an LCS (length 3). Thus the simplified approach above does not work.

We summarize the above method as Algorithms 3,4. If we have known that the LCS is unique, then we just need to apply Algorithm 3, so that genes in $\mathcal{S}_0$ are proper-transposable, and genes not in $\mathcal{S}_0$ are non-transposable. Assume we have m sequences with length n, and the length of the LCS is $n - k$. The time complexities of Step 2 and Step 3 in Algorithm 3 are

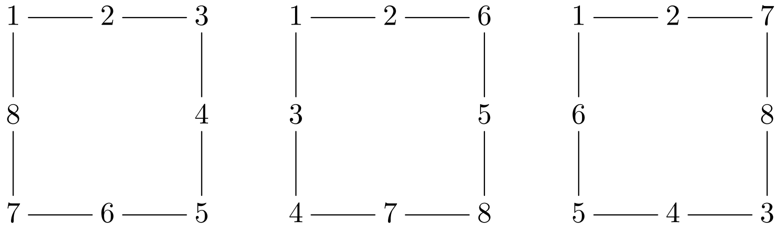

Figure 4: A counterexample with three circular sequences that fails Conjecture 1.

$\mathcal{O}(mn^2)$ and $\mathcal{O}(kmn^2)$. The time complexities of Step 2 in Algorithm 4 is $\mathcal{O}(kmn^2)$. The overall time complexity of determining transposable genes in Scenario 2 by Algorithms 3,4 is $\mathcal{O}(kmn^2)$. The space complexity is trivially $\mathcal{O}(mn + n^2)$.

5.3 Applications on experimental data

Similar to Subsection 4.6, we test Algorithms 3,4 on *Escherichia coli* gene sequences. From NCBI sequencing database, we obtain gene sequences of three individuals of *E. coli* strain ST540 (GenBank CP007265.1, GenBank CP007390.1, GenBank CP007391.1) and three individuals of *E. coli* strain ST2747 (GenBank CP007392.1, GenBank CP007393.1, GenBank CP007394.1).

We regard all three sequences of ST540 as circular gene sequences. We remove genes that appear more than once in one sequence, and remove genes that do not appear in all three sequences. After applying Algorithms 3,4 on these three sequences, there are 389 non-transposable genes, 50 quasi-transposable genes, and 129 proper-transposable genes. The reason for the large amount of proper-transposable genes is that sequence CP007265.1 is significantly different from the other two. After removing it and applying Algorithms 3,4 to the remaining two sequences (CP007390.1 and CP007391.1), there are 564 non-transposable genes and 4 quasi-transposable genes (hpaC, iraD, fbpC, psiB). Therefore, some genes in hpaC, iraD, fbpC, psiB are likely to translocate.

We regard all three sequences of ST2747 as circular gene sequences. We remove genes that appear more than once in one sequence, and remove genes that do not appear in all three sequences. After applying Algorithms 3,4 on these three sequences, all 573 genes are non-transposable genes.

```
1. Input
     $m$ circular sequences of genes $1, \dots, n$, where each gene has only
   one copy in each sequence

2. Choose a gene $g_i$ randomly
   Cut all circular sequences at $g_i$ and expand them to be linear
   sequences
   Apply Algorithm 1 to find $\mathcal{L}_i$, an LCS in the expanded linear
   sequences
   Set $C$ to be the length of $\mathcal{L}_i$, and set $\mathcal{S}$ to be the complement of $\mathcal{L}_i$

3. While $\mathcal{S}$ has a gene $g_j$ that has not been chosen and cut
     Cut all circular sequences at $g_j$ and apply Algorithm 1 to find $\mathcal{L}_j$
     Denote the length of $\mathcal{L}_j$ by $C_j$
     If $C_j > C$
       Update $C$ to be $C_j$, and update $\mathcal{S}$ to be the complement of $\mathcal{L}_j$
     End of if
   End of while
   Denote the final $C$ by $C_0$, and denote the final $\mathcal{S}$ by $\mathcal{S}_0$

4. Output $C_0$ and $\mathcal{S}_0$
```

Algorithm 3: Detailed workflow of determining proper-transposable genes and quasi-transposable genes in Scenario 2, preparation stage.

```
1. Input
     $m$ circular sequences of genes $1, \dots, n$, where each gene has only
   one copy in each sequence; $C_0$ and $\mathcal{S}_0$ calculated from Algorithm 3

2. For each gene $g_l \in \mathcal{S}_0$
     Cut all circular sequences at $g_l$ and expand them to be linear
   sequences
     Apply Algorithm 1 to find $\mathcal{L}_l$, an LCS in the expanded linear
   sequences.
     Denote the length of $\mathcal{L}_l$ by $C_l$
     If $C_l < C_0$
       Output $g_l$ is a proper-transposable gene
     Else
       Output $g_l$ is a quasi-transposable gene
       Cut all circular sequences at $g_l$ and apply Algorithms 1,2 to
   find all proper/quasi-transposable genes for linear gene sequences
   starting at $g_l$
       Output genes not in $\mathcal{S}_0$ but being proper/quasi-transposable for
   such linear sequences are quasi-transposable for circular sequences
     End of if
   End of for
   Output other genes that have not been determined to be
   proper/quasi-transposable are all non-transposable

3. Output: whether each gene is proper/quasi/non-transposable
```

Algorithm 4: Detailed workflow of determining proper-transposable genes and quasi-transposable genes in Scenario 2, output stage.

6 Linear sequences with duplicated genes

In Scenario 3, consider m linear gene sequences, where each sequence contains different numbers of copies of n genes $1, \ldots, n$. We need to find the LCS. Here we only consider common subsequences that consist of all or none copies of the same gene, and the subsequence length is calculated by genes, not gene copies.

6.1 A graph representation of the problem

Similar to Scenario 1, we construct an auxiliary graph $\mathcal{G}$, where each vertex is a gene (not a copy of a gene). However, in this case, the auxiliary graph is undirected: There is an undirected edge between gene g_i and gene g_j if and only if all the copies of g_i and g_j keep their relative locations in all sequences. For example, consider two sequences $(1, 2, 3, 2, 3, 4, 5)$ and $(2, 1, 3, 3, 2, 4, 5)$. For gene pair $1, 3$, the corresponding sequences are $(1, 3, 3)$ and $(1, 3, 3)$, meaning that there is an edge between 1 and 3. For gene pair $1, 2$, the corresponding sequences are $(1, 2, 2)$ and $(2, 1, 2)$, meaning that there is no edge between 1 and 2. See Fig. 5 for the auxiliary graph in this case.

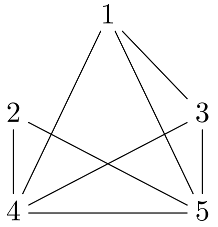

Figure 5: The auxiliary graph $\mathcal{G}$ of two sequences $(1, 2, 3, 2, 3, 4, 5)$ and $(2, 1, 3, 3, 2, 4, 5)$. The unique largest complete subgraph is $\{1, 3, 4, 5\}$, meaning that the unique longest common sequence is $(1, 3, 3, 4, 5)$. Thus $1, 3, 4, 5$ are non-transposable genes, and 2 is a proper-transposable gene.

Definition 5. *A subgraph of $\mathcal{G}$ consists of some genes $g_1, \ldots, g_l$ and the edges between them. In a subgraph, if there is an edge between any two genes, this subgraph is called a complete subgraph (also called a clique).*

Definition 6. *In graph $\mathcal{G}$, the degree of a gene g is the number of edges linking g. In a complete graph of p genes, where any two genes have an edge in between, each gene has degree $p - 1$.*

Definition 7. *If all copies of genes $g_1, \dots, g_l$ keep their relative locations in all linear sequences, we say that $g_1, \dots, g_l$ form a common subsequence.*

The following Lemma 3 shows that there is a bijection between common subsequences and complete subgraphs in $\mathcal{G}$. *The problem of determining the LCS now becomes determining the largest complete subgraph of $\mathcal{G}$.*

Lemma 3. *In Scenario 3, construct the auxiliary graph $\mathcal{G}$ from gene sequences. If $g_1, \dots, g_k$ form a complete subgraph in $\mathcal{G}$, then $g_1, \dots, g_k$ form a common subsequence, and vice versa.*

Proof. If $g_1, \dots, g_l$ form a common subsequence, then there is an edge in $\mathcal{G}$ between any two genes in $g_1, \dots, g_l$, meaning that they form a complete subgraph.

For the other direction, only consider copies of $g_1, \dots, g_k$ in these sequences. If $g_1, \dots, g_k$ do not form a common subsequence, find the first digit that such sequences differ. Assume g_p and g_q can both appear in this digit. Then g_p, g_q cannot form a common subsequence, and there is no edge between g_p and g_q.

We illustrate this proof with Fig. 5: For genes $2, 3, 4$, the sequences are $(2, 3, 2, 3, 4)$ and $(2, 3, 3, 2, 4)$. The third digit is different, where 2 and 3 can both appear. Then the sequences for genes $2, 3$, $(2, 3, 2, 3)$ and $(2, 3, 3, 2)$, cannot match, and there is no edge between 2 and 3. □

6.2 A heuristic algorithm

The above discussion shows that given gene sequences, we can construct an undirected graph $\mathcal{G}$, so that there is a bijection between common subsequences and complete subgraphs. The inverse also holds: We can construct corresponding gene sequences for a graph.

Lemma 4. *Given an undirected graph $\mathcal{G}$, we can construct two gene sequences, so that there is a bijection between common subsequences and complete subgraphs.*

Proof. Assume the graph has n genes. We start with two sequences $(1, 2, \dots, n)$ and$(1, 2, \dots, n)$. For each pair of genes g_i, g_j, if there is no edge between them in $\mathcal{G}$, add g_i, g_j to the end of the first sequence, and g_j, g_i to the end of the second sequence. Then g_i, g_j cannot both appear in a common subsequence, and this operation does not affect other gene pairs.

For example, corresponding to Fig. 5, we start with $(1, 2, 3, 4, 5)$ and $(1, 2, 3, 4, 5)$. Since there is no edge between $1, 2$, we add them to have

$(1, 2, 3, 4, 5, 1, 2)$ and $(1, 2, 3, 4, 5, 2, 1)$. Since there is no edge between $2, 3$, we add them to have $(1, 2, 3, 4, 5, 1, 2, 2, 3)$ and $(1, 2, 3, 4, 5, 2, 1, 3, 2)$. These two sequences corresponds to Fig. 5. □

Combining Lemma 3 and Lemma 4, we obtain the following result:

Proposition 1. *Finding the longest common sequence in Scenario 3 is equivalent to the maximum clique problem, which is NP-hard.*

Proof. For an undirected graph, we can use Lemma 4 to construct corresponding sequences. If we have the solution of finding the longest common sequence in Scenario 3, then we can find the largest complete subgraph in an extra polynomial time.

For gene sequences in Scenario 3, we can construct corresponding auxiliary graph. If we have the solution of finding the largest complete subgraph, then we can use Lemma 3 to find the longest common sequence in Scenario 3 in an extra polynomial time.

Therefore, finding the longest common sequence in Scenario 3 and finding the largest complete subgraph are equivalent. The problem of determining the largest complete subgraph is just the maximum clique problem, which is NP-hard [65]. Thus finding the longest common sequence in Scenario 3 is also NP-hard. This means it is not likely to design an algorithm that always correctly determines the LCS in polynomial time. □

We have transformed Scenario 3 into the maximum clique problem for a graph $\mathcal{G}$. There have been various algorithms for the maximum clique problem [30, 37, 72], and readers may refer to a review for more details [78]. For completeness, we propose a simple idea: In the auxiliary graph $\mathcal{G}$, repeatedly abandon the gene with the smallest degree (and also edges linking this gene) until the remaining genes form a complete subgraph. See Algorithm 5 for the details of this greedy heuristic method. This algorithm is easy to understand, and can provide some intuition. We do not claim that Algorithm 5 is comparable to other sophisticated algorithms.

We test Algorithm 5 on random graphs. Construct a random graph with n genes, and any two genes have probability 0.5 to have an edge in between. Use brute-force search to find the maximum clique, and compare its size with the result of Algorithm 5. For each $n \leq 15$, we repeat this for 10000 times, and every time Algorithm 5 returns the correct result. Therefore, for small random graphs, the 95% credible interval for the success rate of Algorithm 5 is $[0.9997, 1]$. We can claim that Algorithm 5 is a good heuristic algorithm that fails with a very small probability. Since finding the true maximum

```
1. Input
     $m$ linear sequences of genes $1, \ldots, n$, where each gene can have
   multiple copies

2. Construct the auxiliary graph $\mathcal{G}$:
     Vertices of $\mathcal{G}$ are all the genes $1, \ldots, n$ (not their copies)
     For each pair of genes $g_i, g_j$
       If all copies of $g_i$ and $g_j$ keep their relative locations in all $m$
   sequences
          Add an undirected edge between $g_i$ and $g_j$ in $\mathcal{G}$
       End of if
     End of for
   Calculate the degree for each gene in $\mathcal{G}$

3. While true
     Find a gene $g_i$ with the smallest degree $d_i$ in $\mathcal{G}$
     % If the minimal $g_i$ is not unique, choose one randomly
     If $d_i + 1$ is smaller than the number of genes in $\mathcal{G}$
       Delete $g_i$ and edges linking $g_i$ in $\mathcal{G}$
       Update the degrees of other genes
     Else
     % The remaining genes form a complete subgraph
       Break the while loop
     End of if
   End of while
   % The final $\mathcal{G}$ is a complete subgraph of the original $\mathcal{G}$, and it is
   likely to be the largest one

4. Output genes in the final $\mathcal{G}$ are not transposable, and genes not in
   the final $\mathcal{G}$ are transposable
```

Algorithm 5: A heuristic method for detecting transposable genes in Scenario 3.

clique requires exponentially slow brute-force search, we do not test on very large graphs.

Nevertheless, Algorithm 5 does not always produce the correct result. See Fig. 6 for a counterexample. Here genes $1, 2, 3, 4, 5, 6$ have degree 4, while genes $7, 8, 9, 10$ have degree 3. When applying Algorithm 5, genes $7, 8, 9, 10$ are first abandoned, and the final result just has three genes, such as $1, 3, 5$. However, the largest complete graph is $7, 8, 9, 10$. Besides, Algorithm 5 can only determine one (possibly longest) common subsequence. Thus we cannot determine the existence of quasi-transposable genes.

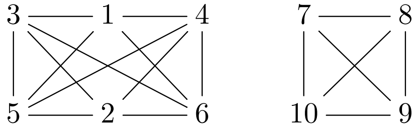

Figure 6: The auxiliary graph $\mathcal{G}$ of linear sequences $(7, 8, 9, 10, 1, 1, 2, 3, 3, 4, 5, 5, 6)$ and $(1, 2, 1, 3, 4, 3, 5, 6, 5, 7, 8, 9, 10)$. This counterexample fails Algorithm 5.

Assume we have m sequences with n genes. In general, the copy number of a gene is small, and we can assume the length of each sequence is $\mathcal{O}(n)$. The time complexities of Step 2 and Step 3 in Algorithm 5 are $\mathcal{O}(mn^2)$ and $\mathcal{O}(n^2)$, and the overall time complexity is $\mathcal{O}(mn^2)$. The space complexity is trivially $\mathcal{O}(mn + n^2)$.

7 Circular sequences with duplicated genes

In Scenario 4, consider m circular gene sequences, where each sequence contains different numbers of copies of n genes $1, \dots, n$. We need to find the LCS. Here we only consider common subsequences that consist of all or none copies of the same gene, and the subsequence length is calculated by genes, not gene copies.

We shall prove that finding the LCS in Scenario 4 is no easier than in Scenario 3. Thus Scenario 4 is also NP-hard.

Proposition 2. *Finding the LCS in Scenario 4 is NP-hard.*

Proof. From Proposition 1, Scenario 3 is NP-hard, meaning that any NP problem can be reduced to Scenario 3 in polynomial time. We just need to prove that Scenario 3 can be reduced to Scenario 4 in polynomial time.

Given m linear sequences with n genes in Scenario 3, add genes $n+1,\dots,2n+1$ to the end of each sequence, and glue each linear sequence into a circular sequence. The LCS for these circular sequence has the following properties: (1) it contains all genes $n+1,\dots,2n+1$; (2) after cutting at $n+1$ and removing genes $n+1,\dots,2n+1$, the remaining linear sequence is the LCS in Scenario 3.

(1) The LCS has at least $n+1$ genes $(n+1,\dots,2n+1)$. Therefore, at least one gene in $n+1,\dots,2n+1$ is included, such as $n+1$. Since gene $n+1$ aligned in all sequences, $n+2,\dots,2n+1$ are also aligned, meaning that they are also in the LCS.

(2) After cutting and removing $n+1,\dots,2n+1$, the remaining linear sequence is a common subsequence in Scenario 3. If there is a longer common subsequence, then that with $n+1,\dots,2n+1$ should be a longer common subsequence in Scenario 4, a contradiction.

Therefore, if we can find the LCS for these circular sequences, then we can find the LCS for linear sequences in polynomial time. □

Similar to Scenario 3, to find the LCS in Scenario 4, we want to reduce it to a maximum clique problem. However, Lemma 3 does not hold in Scenario 4. For example, we can consider a circular sequence $(1,2,3)$ and its mirror symmetry. These two sequences are different, but any two genes form a common subsequence. However, inspired by Lemma 3, we have the following conjecture, although we do not know if it is correct or not.

Conjecture 2. *In Scenario 4, if any three genes g_i, g_j, g_l in $g_1,\dots,g_k$ form a common subsequence, then $g_1,\dots,g_k$ form a common subsequence.*

To solve Scenario 4,construct a 3-uniform hypergraph $\mathcal{G}$ as following [15]: vertices are genes $1,\dots,n$; there is a 3-hyperedge (undirected) that links genes g_i, g_j, g_k if and only if they form a common subsequence.

Proposition 3. *If Conjecture 2 holds, then finding the longest common sequence in Scenario 4 can be reduced to the maximum clique problem for* 3*-uniform hypergraphs.*

Proof. If $g_1,\dots,g_k$ form a common subsequence, then any three genes g_i, g_j, g_l has a 3-hyperedge, and $g_1,\dots,g_k$ form a complete subgraph. If $g_1,\dots,g_k$ form a complete subgraph, then any three genes g_i, g_j, g_l form a common subsequence. By Conjecture 2 , this means $g_1,\dots,g_k$ form a common subsequence. Therefore, there is a bijection between common subsequence and complete subgraph. If we can find the maximum clique problem for 3-uniform hypergraphs, then it corresponds to the LCS. □

We have reduced Scenario 4 into the maximum clique problem for 3-uniform hypergraphs, which is also NP-hard [78]. There have been some algorithms for the maximum clique problem for 3-uniform hypergraphs [64, 56]. For completeness, we propose a simple idea: Repeatedly delete the gene that has the smallest degree, until we have a complete subgraph that any three genes have a 3-hyperedge that links them. We summarize this greedy heuristic method as Algorithm 6. This algorithm is easy to understand, and can provide some intuition. We do not claim that Algorithm 6 is comparable to other sophisticated algorithms.

We test Algorithm 6 on random graphs. Construct a random graph with n genes, and any two genes have probability 0.5 to have an edge in between. Use brute-force search to find the maximum clique, and compare its size with the result of Algorithm 6. For each $n \leq 15$, we repeat this for 10000 times, and every time Algorithm 6 returns the correct result. Therefore, for small random graphs, the 95% credible interval for the success rate of Algorithm 6 is $[0.9997, 1]$. We can claim that Algorithm 6 is a good heuristic algorithm that fails with a very small probability. Since finding the true maximum clique requires exponentially slow brute-force search, we do not test on very large graphs.

Nevertheless, Algorithm 6 does not always produce the correct result. See Fig. 7 for a counterexample. Here each gene in $1, 2, 3, 4, 5, 6$ has degree 4, while each gene in $7, 8, 9, 10$ has degree 3 .When applying Algorithm 6, genes $7, 8, 9, 10$ are first deleted, and the final result just has three genes, such as $(1, 3, 5)$. However, the LCS $(7, 8, 9, 10)$ has four genes.

Assume we have m sequences with n genes. In general, the copy number of a gene is small, and we can assume the length of each sequence is $\mathcal{O}(n)$. The time complexities of Step 2 and Step 3 in Algorithm 6 are $\mathcal{O}(mn^3)$ and $\mathcal{O}(n^3)$, and the overall time complexity is $\mathcal{O}(mn^3)$. The space complexity is trivially $\mathcal{O}(mn + n^3)$.

8 Discussion

A gene g_i might be missing in some sequences. Since g_i is not in any LCS, it should be a proper-transposable gene. This gene can be directly removed before applying corresponding algorithms.

We can adopt a stricter definition of transposable genes to exclude a gene which only changes its relative position in a few (no more than l, where l is small enough) sequences. Then we should consider the longest sequence which is a common subsequence of at least $m - l$ sequences. We can run

```
1. Input
     $m$ circular sequences of genes $1, \ldots, n$, where each gene can have multiple copies
2. Construct the auxiliary graph $\mathcal{G}$:
     Vertices of $\mathcal{G}$ are all the genes $1, \ldots, n$ (not their copies)
     For each gene triple $g_i, g_j, g_k$
       If all copies of $g_i, g_j, g_k$ keep their relative locations in all $m$ sequences
           Add a 3-hyperedge that links $g_i, g_j, g_k$ in $\mathcal{G}$
       End of if
     End of for
3. While there exist three genes that do not share a 3-hyperedge
     Calculate the degree for each gene in $\mathcal{G}$
     Delete the gene with the smallest degree and 3-hyperedges that links this gene
     % If there are multiple genes with the smallest degree, delete one randomly
   End of while
   % After this while loop, any three genes form a common subsequence
   % If Conjecture 2 holds, the remaining genes form a common subsequence
4. Output remaining genes are not transposable, and other genes are transposable
```

Algorithm 6: A heuristic method for detecting transposable genes in Scenario 4.

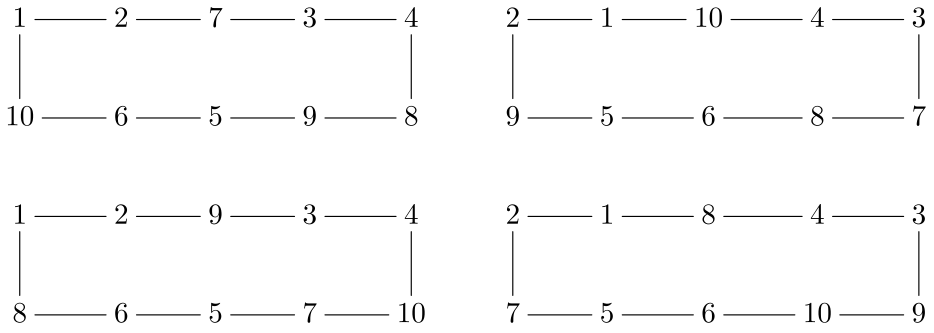

Figure 7: Four circular sequences. The LCS is $(7, 8, 9, 10)$. This counterexample fails Algorithm 6.

the corresponding algorithm for every $m - l$ sequences. Thus the total time complexity will be multiplied by a factor of m^l.

In Scenario 1 and Scenario 2 (linear/circular sequences without duplicated genes), if each sequence has n genes, and the LCS has length $n-k$, then there are at most k proper-transposable genes. About quasi-transposable genes, inspired by Lemma 1, we have the following guess.

Conjecture 3. *Consider m linear/circular sequences with n genes without multiple copies. Assume the length of the LCS is $n - k$, and there are l proper-transposable genes. Then the number of quasi-transposable genes is no larger than $2(k - l)$.*

When $l + 2(k - l) \leq n$, in both linear and circular scenarios, we can find examples with $2(k - l)$ quasi-transposable genes.

9 Conclusion

In this paper, we study the LCS problem and design Algorithms 1–6 for different scenarios. Specifically, we consider the case where the LCS is not unique, and determine whether each number appears in all/some/none of the LCSs. These algorithms are applied to gene sequences to determine the stability of genes. To apply those algorithms, one needs to apply genomic annotation tools to transform raw DNA sequencing data into gene sequences, and replace gene names by numbers. Those algorithms have at most $O(mn^3)$ time complexity, where m is the number of sequences, and n is the number of genes. Thus they can run in a reasonable time for most applications.

We prove that the latter two scenarios are NP-hard (Propositions 1, 2), and propose two unresolved problems (Conjectures 2, 3) in discrete mathematics.

We start with gene sequences and determine translocated genes. Therefore, short transposons (possibly shorter than a gene) cannot be determined. Besides, we do not determine specific genomic rearrangement events. We aim at determining which genes are able to translocate (i.e., less stable). Specifically, we study how many LCSs contain a certain gene, as a measure for its "stability". This mesoscopic viewpoint can be intriguing for understanding changes in genome.

The results in this paper are not limited to Scenarios 1–4. They can be applied to other bioinformatics situations, or even other fields that need discrete mathematics tools, such as text processing, compiler optimization, data analysis, image analysis [22]. Besides, algorithms in this paper might be able to detect non-syntenic regions [36].

There are some possible future directions: (1) prove Conjectures 2, 3; (2) extend Proposition 3 to find more efficient solutions to Scenario 4; (3) determine whether genes appear in all LCSs in other similar scenarios.

Acknowledgments

The author would like to thank Zhongkai Zhao for helping with designing Algorithm 1. The author would like to thank Lucas Böttcher and anonymous reviewers for providing helpful comments.

References

[1] Adi, S. S., Braga, M. D., Fernandes, C. G., Ferreira, C. E., Martinez, F. V., Sagot, M.-F., Stefanes, M. A., Tjandraatmadja, C., and Wakabayashi, Y. Repetition-free longest common subsequence. *Discrete Applied Mathematics 158*, 12 (2010), 1315–1324.

[2] Angelini, E., Wang, Y., Zhou, J. X., Qian, H., and Huang, S. A model for the intrinsic limit of cancer therapy: Duality of treatment-induced cell death and treatment-induced stemness. *PLOS Computational Biology 18*, 7 (2022), e1010319.

[3] Babakhani, S., and Oloomi, M. Transposons: the agents of antibiotic resistance in bacteria. *Journal of Basic Microbiology 58*, 11 (2018), 905–917.

[4] Backurs, A., and Indyk, P. Edit distance cannot be computed in strongly subquadratic time (unless SETH is false). In *Proceedings of the Forty-Seventh Annual ACM Symposium on Theory of Computing* (2015), pp. 51–58.

[5] Bergroth, L., Hakonen, H., and Raita, T. A survey of longest common subsequence algorithms. In *Proceedings Seventh International Symposium on String Processing and Information Retrieval. SPIRE 2000* (2000), IEEE, pp. 39–48.

[6] Blanchette, M., Kunisawa, T., and Sankoff, D. Parametric genome rearrangement. *Gene 172*, 1 (1996), GC11–GC17.

[7] Blum, C., Djukanovic, M., Santini, A., Jiang, H., Li, C.-M., Manyà, F., and Raidl, G. R. Solving longest common subsequence problems via a transformation to the maximum clique problem. *Computers & Operations Research 125* (2021), 105089.

[8] Bohnenkämper, L., Braga, M. D., Doerr, D., and Stoye, J. Computing the rearrangement distance of natural genomes. *Journal of Computational Biology 28*, 4 (2021), 410–431.

[9] Brůna, T., Hoff, K. J., Lomsadze, A., Stanke, M., and Borodovsky, M. Braker2: automatic eukaryotic genome annotation with genemark-ep+ and augustus supported by a protein database. *NAR genomics and bioinformatics 3*, 1 (2021), lqaa108.

[10] Chen, X., and Li, D. ERVcaller: identifying polymorphic endogenous retrovirus and other transposable element insertions using whole-genome sequencing data. *Bioinformatics 35*, 20 (2019), 3913–3922.

[11] Chen, X., Wang, Y., Feng, T., Yi, M., Zhang, X., and Zhou, D. The overshoot and phenotypic equilibrium in characterizing cancer dynamics of reversible phenotypic plasticity. *Journal of Theoretical Biology 390* (2016), 40–49.

[12] Chen, Z.-Z., Gao, Y., Lin, G., Niewiadomski, R., Wang, Y., and Wu, J. A space-efficient algorithm for sequence alignment with inversions and reversals. *Theoretical Computer Science 325*, 3 (2004), 361–372.

[13] DeNicola, G. M., Karreth, F. A., Adams, D. J., and Wong, C. C. The utility of transposon mutagenesis for cancer studies in the era of genome editing. *Genome Biology 16*, 1 (2015), 1–15.

[14] Dessalles, R., Pan, Y., Xia, M., Maestrini, D., D'Orsogna, M. R., and Chou, T. How naive t-cell clone counts are shaped by heterogeneous thymic output and homeostatic proliferation. *Frontiers in immunology 12* (2022), 5529.

[15] Diestel, R. *Graph Theory*, 5 ed. Springer, Berlin, 2017.

[16] Diop, S. I., Subotic, O., Giraldo-Fonseca, A., Waller, M., Kirbis, A., Neubauer, A., Potente, G., Murray-Watson, R., Boskovic, F., Bont, Z., et al. A pseudomolecule-scale genome assembly of the liverwort marchantia polymorpha. *The Plant Journal 101*, 6 (2020), 1378–1396.

[17] Evrony, G. D., Hinch, A. G., and Luo, C. Applications of single-cell dna sequencing. *Annual review of genomics and human genetics 22* (2021), 171–197.

[18] Gardner, E. J., Lam, V. K., Harris, D. N., Chuang, N. T., Scott, E. C., Pittard, W. S., Mills, R. E., Devine, S. E., Consortium, . G. P., et al. The Mobile Element Locator Tool (MELT): population-scale mobile element discovery and biology. *Genome Research 27*, 11 (2017), 1916–1929.

[19] Goel, M., Sun, H., Jiao, W.-B., and Schneeberger, K. Syri: finding genomic rearrangements and local sequence differences from whole-genome assemblies. *Genome biology 20*, 1 (2019), 1–13.

[20] Goubert, C., Craig, R. J., Bilat, A. F., Peona, V., Vogan, A. A., and Protasio, A. V. A beginner's guide to manual curation of transposable elements. *Mobile DNA 13*, 1 (2022), 1–19.

[21] Gu, W., Zhang, F., and Lupski, J. R. Mechanisms for human genomic rearrangements. *Pathogenetics 1* (2008), 1–17.

[22] Hajiaghayi, M., Seddighin, S., and Sun, X. Massively parallel approximation algorithms for edit distance and longest common subsequence. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms* (2019), SIAM, pp. 1654–1672.

[23] Hirschberg, D. S. A linear space algorithm for computing maximal common subsequences. *Communications of the ACM 18*, 6 (1975), 341–343.

[24] Huang, K., Yang, C.-B., Tseng, K.-T., et al. Fast algorithms for finding the common subsequences of multiple sequences. In *Proceedings of the International Computer Symposium* (2004), Citeseer, pp. 1006–1011.

[25] Ibal, J. C., Pham, H. Q., Park, C. E., and Shin, J.-H. Information about variations in multiple copies of bacterial 16s rRNA genes may aid in species identification. *PLOS ONE 14*, 2 (2019), e0212090.

[26] Imbeault, M., Helleboid, P.-Y., and Trono, D. Krab zinc-finger proteins contribute to the evolution of gene regulatory networks. *Nature 543*, 7646 (2017), 550–554.

[27] Islam, M. R., Saifullah, C. K., Asha, Z. T., and Ahamed, R. Chemical reaction optimization for solving longest common subsequence problem for multiple string. *Soft Computing 23* (2019), 5485–5509.

[28] Ivics, Z., and Izsvák, Z. The expanding universe of transposon technologies for gene and cell engineering. *Mobile DNA 1*, 1 (2010), 1–15.

[29] Jiang, D.-Q., Wang, Y., and Zhou, D. Phenotypic equilibrium as probabilistic convergence in multi-phenotype cell population dynamics. *PLOS ONE 12*, 2 (2017), e0170916.

[30] Jiang, H., Li, C.-M., and Manya, F. Combining efficient preprocessing and incremental MaxSAT reasoning for maxclique in large graphs. In *Proceedings of the twenty-second European conference on artificial intelligence* (2016), pp. 939–947.

[31] Jiang, T., Lin, G.-H., Ma, B., and Zhang, K. The longest common subsequence problem for arc-annotated sequences. In *CPM* (2000), vol. 1848, Springer, pp. 154–165.

[32] Kang, Y., Gu, C., Yuan, L., Wang, Y., Zhu, Y., Li, X., Luo, Q., Xiao, J., Jiang, D., Qian, M., et al. Flexibility and symmetry of prokaryotic genome rearrangement reveal lineage-associated core-gene-defined genome organizational frameworks. *mBio 5* (2014), e01867.

[33] Kazazian, H. H., Wong, C., Youssoufian, H., Scott, A. F., Phillips, D. G., and Antonarakis, S. E. Haemophilia A resulting from de novo insertion of L1 sequences represents a novel mechanism for mutation in man. *Nature 332*, 6160 (1988), 164–166.

[34] Kececioglu, J., and Sankoff, D. Exact and approximation algorithms for sorting by reversals, with application to genome rearrangement. *Algorithmica 13*, 1 (1995), 180–210.

[35] Kiyomi, M., Horiyama, T., and Otachi, Y. Longest common subsequence in sublinear space. *Information Processing Letters 168* (2021), 106084.

[36] Lee, K.-C., and Kim, S.-S. Non-synteny regions in the human genome. *Genomics & Informatics 8*, 2 (2010), 86–89.

[37] Li, C.-M., Jiang, H., and Manyà, F. On minimization of the number of branches in branch-and-bound algorithms for the maximum clique problem. *Computers & Operations Research 84* (2017), 1–15.

[38] Lin, C.-H., Lian, C.-Y., Hsiung, C. A., and Chen, F.-C. Changes in transcriptional orientation are associated with increases in evolutionary rates of enterobacterial genes. In *BMC bioinformatics* (2011), vol. 12, BioMed Central, pp. 1–8.

[39] Lin, C.-Y., and Och, F. J. Automatic evaluation of machine translation quality using longest common subsequence and skip-bigram statistics. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL-04)* (2004), pp. 605–612.

[40] Maier, D. The complexity of some problems on subsequences and supersequences. *Journal of the ACM (JACM) 25*, 2 (1978), 322–336.

[41] Makałowski, W., Gotea, V., Pande, A., and Makałowska, I. Transposable elements: Classification, identification, and their use as a tool for comparative genomics. In *Evolutionary Genomics*. Springer, 2019, pp. 177–207.

[42] McClintock, B. The origin and behavior of mutable loci in maize. *Proceedings of the National Academy of Sciences 36*, 6 (1950), 344–355.

[43] Mills, R. E., Bennett, E. A., Iskow, R. C., and Devine, S. E. Which transposable elements are active in the human genome? *Trend in Genetics 23*, 4 (2007), 183–191.

[44] Mitsuhashi, S., Ohori, S., Katoh, K., Frith, M. C., and Matsumoto, N. A pipeline for complete characterization of complex germline rearrangements from long dna reads. *Genome medicine 12*, 1 (2020), 1–17.

[45] Mousavi, S. R., and Tabataba, F. An improved algorithm for the longest common subsequence problem. *Computers & Operations Research 39*, 3 (2012), 512–520.

[46] Mustajoki, S., Ahola, H., Mustajoki, P., and Kauppinen, R. Insertion of Alu element responsible for acute intermittent porphyria. *Human Mutation 13*, 6 (1999), 431–438.

[47] Ngomade, A. N., Myoupo, J. F., and Tchendji, V. K. A dominant point-based parallel algorithm that finds all longest common subsequences for a constrained-mlcs problem. *Journal of computational science 40* (2020), 101070.

[48] Niu, X.-M., Xu, Y.-C., Li, Z.-W., Bian, Y.-T., Hou, X.-H., Chen, J.-F., Zou, Y.-P., Jiang, J., Wu, Q., Ge, S., et al. Transposable elements drive rapid phenotypic variation in capsella rubella. *Proceedings of the National Academy of Sciences 116*, 14 (2019), 6908–6913.

[49] Niu, Y., Wang, Y., and Zhou, D. The phenotypic equilibrium of cancer cells: From average-level stability to path-wise convergence. *Journal of Theoretical Biology 386* (2015), 7–17.

[50] Noorani, I., Bradley, A., and de la Rosa, J. Crispr and transposon in vivo screens for cancer drivers and therapeutic targets. *Genome biology 21*, 1 (2020), 1–22.

[51] Nowacki, M., Higgins, B. P., Maquilan, G. M., Swart, E. C., Doak, T. G., and Landweber, L. F. A functional role for transposases in a large eukaryotic genome. *Science 324*, 5929 (2009), 935–938.

[52] Orozco-Arias, S., Piña, J. S., Tabares-Soto, R., Castillo-Ossa, L. F., Guyot, R., and Isaza, G. Measuring performance metrics of machine learning algorithms for detecting and classifying transposable elements. *Processes 8*, 6 (2020), 638.

[53] Payer, L. M., and Burns, K. H. Transposable elements in human genetic disease. *Nature Reviews Genetics 20*, 12 (2019), 760–772.

[54] Rahrmann, E. P., Collier, L. S., Knutson, T. P., Doyal, M. E., Kuslak, S. L., Green, L. E., Malinowski, R. L., Roethe, L.,

Akagi, K., Waknitz, M., et al. Identification of pde4d as a proliferation promoting factor in prostate cancer using a sleeping beauty transposon-based somatic mutagenesis screen. *Cancer research 69*, 10 (2009), 4388–4397.

[55] Reneker, J., Lyons, E., Conant, G. C., Pires, J. C., Freeling, M., Shyu, C.-R., and Korkin, D. Long identical multispecies elements in plant and animal genomes. *Proceedings of the National Academy of Sciences of the United States of America 109*, 19 (2012), E1183–E1191.

[56] Rota Bulò, S., and Pelillo, M. A continuous characterization of maximal cliques in k-uniform hypergraphs. In *International conference on learning and intelligent optimization* (2007), Springer, pp. 220–233.

[57] Rowley, M. J., and Corces, V. G. Organizational principles of 3D genome architecture. *Nature Reviews Genetics 19*, 12 (2018), 789–800.

[58] Salzberg, S. L. Next-generation genome annotation: we still struggle to get it right, 2019.

[59] Sha, Y., Wang, S., Zhou, P., and Nie, Q. Inference and multiscale model of epithelial-to-mesenchymal transition via single-cell transcriptomic data. *Nucleic acids research 48*, 17 (2020), 9505–9520.

[60] Silfverberg, M., Liu, L., and Hulden, M. A computational model for the linguistic notion of morphological paradigm. In *Proceedings of the 27th International Conference on Computational Linguistics* (2018), pp. 1615–1626.

[61] Sorokin, A. Using longest common subsequence and character models to predict word forms. In *Proceedings of the 14th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology* (2016), pp. 54–61.

[62] Terauds, V., and Sumner, J. Maximum likelihood estimates of rearrangement distance: implementing a representation-theoretic approach. *Bulletin of mathematical biology 81* (2019), 535–567.

[63] Tesler, G. Efficient algorithms for multichromosomal genome rearrangements. *Journal of Computer and System Sciences 65*, 3 (2002), 587–609.

[64] Torres-Jimenez, J., Perez-Torres, J. C., and Maldonado-Martinez, G. hclique: An exact algorithm for maximum clique problem in uniform hypergraphs. *Discrete Mathematics, Algorithms and Applications 9*, 06 (2017), 1750078.

[65] Valiente, G. *Algorithms on Trees and Graphs.* Springer, Berlin, 2002.

[66] Verma, S. C., Qian, Z., and Adhya, S. L. Architecture of the Escherichia coli nucleoid. *PLOS Genetics 15*, 12 (2019), e1008456.

[67] Wang, Q., Korkin, D., and Shang, Y. A fast multiple longest common subsequence (mlcs) algorithm. *IEEE Transactions on Knowledge and Data Engineering 23*, 3 (2010), 321–334.

[68] Wang, T., Weiss, A., Aqeel, A., Wu, F., Lopatkin, A. J., David, L. A., and You, L. Horizontal gene transfer enables programmable gene stability in synthetic microbiota. *Nature Chemical Biology 18*, 11 (2022), 1245–1252.

[69] Wang, X., Wang, L., and Zhu, D. Efficient computation of longest common subsequences with multiple substring inclusive constraints. *Journal of Computational Biology 26*, 9 (2019), 938–947.

[70] Wang, Y. *Some Problems in Stochastic Dynamics and Statistical Analysis of Single-Cell Biology of Cancer.* Ph.D. thesis, University of Washington, 2018.

[71] Wang, Y. Two metrics on rooted unordered trees with labels. *Algorithms for Molecular Biology 17*, 1 (2022), 1–17.

[72] Wang, Y., Cai, S., and Yin, M. Two efficient local search algorithms for maximum weight clique problem. In *Thirtieth AAAI Conference on Artificial Intelligence* (2016), pp. 805–811.

[73] Wang, Y., Kropp, J., and Morozova, N. Biological notion of positional information/value in morphogenesis theory. *International Journal of Developmental Biology 64*, 10-11-12 (2020), 453–463.

[74] Wang, Y., and Wang, L. Causal inference in degenerate systems: An impossibility result. In *International Conference on Artificial Intelligence and Statistics* (2020), PMLR, pp. 3383–3392.

[75] Wang, Y., and Wang, Z. Inference on the structure of gene regulatory networks. *Journal of Theoretical Biology 539* (2022), 111055.

[76] Wang, Y., Zhang, B., Kropp, J., and Morozova, N. Inference on tissue transplantation experiments. *Journal of Theoretical Biology 520* (2021), 110645.

[77] Wei, S., Wang, Y., Yang, Y., and Liu, S. A path recorder algorithm for Multiple Longest Common Subsequences (MLCS) problems. *Bioinformatics 36*, 10 (2020), 3035–3042.

[78] Wu, Q., and Hao, J.-K. A review on algorithms for maximum clique problems. *European Journal of Operational Research 242*, 3 (2015), 693–709.

[79] Xia, M., Greenman, C. D., and Chou, T. Pde models of adder mechanisms in cellular proliferation. *SIAM journal on applied mathematics 80*, 3 (2020), 1307–1335.

[80] Yu, T., Huang, X., Dou, S., Tang, X., Luo, S., Theurkauf, W. E., Lu, J., and Weng, Z. A benchmark and an algorithm for detecting germline transposon insertions and measuring de novo transposon insertion frequencies. *Nucleic Acids Research 49*, 8 (2021), e44–e44.

[81] Zhou, D., Wang, Y., and Wu, B. A multi-phenotypic cancer model with cell plasticity. *Journal of Theoretical Biology 357* (2014), 35–45.

[82] Zhou, P., Wang, S., Li, T., and Nie, Q. Dissecting transition cells from single-cell transcriptome data through multiscale stochastic dynamics. *Nature communications 12*, 1 (2021), 5609.

[83] Zhou, W., Liang, G., Molloy, P. L., and Jones, P. A. Dna methylation enables transposable element-driven genome expansion. *Proceedings of the National Academy of Sciences of the United States of America 117*, 32 (2020), 19359–19366.

[84] Zimin, A. V., Puiu, D., Luo, M.-C., Zhu, T., Koren, S., Marçais, G., Yorke, J. A., Dvořák, J., and Salzberg, S. L. Hybrid assembly of the large and highly repetitive genome of aegilops tauschii, a progenitor of bread wheat, with the masurca mega-reads algorithm. *Genome Research 27*, 5 (2017), 787–792.