View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by IUPUIScholarWorks

The Longest Common Exemplar Subsequence Problem

Shu Zhang[1], Ruizhi Wang[1], Daming Zhu[1,*], Haitao Jiang[1], Haodi Feng[1], Jiong Guo[1] and Xiaowen Liu[2]
[1]School of Computer Science and Technology, Shandong University
[2]Department of BioHealth Informatics, Indiana University-Purdue University Indianapolis
[*]Corresponding author, Email: dmzhu@sdu.edu.cn

***Abstract*—In this paper, we propose to find order conserved subsequences of genomes by finding longest common exemplar subsequences of the genomes. The longest common exemplar subsequence problem is given by two genomes, asks to find a common exemplar subsequence of them, such that the exemplar subsequence length is maximized.**

We focus on genomes whose genes of the same gene family are in at most s spans. We propose a dynamic programming algorithm with time complexity $O(s4^s mn)$ to find a longest common exemplar subsequence of two genomes with one genome admitting s span genes of the same gene family, where m, n stand for the gene numbers of those two given genomes. Our algorithm can be extended to find longest common exemplar subsequences of more than one genomes.

***Index Terms*—dynamic programming, algorithm, exemplar subsequence, genome, s-span.**

I. Introduction

A set of genes that is conserved in the same order during the evolution suggests that it participates to the same biological process [1]. Finding conserved sets of genes in genomes can be done by genome comparison, where a similarity measure of genomes has to be introduced for measuring how good a conserved gene set is. The number of breakpoints as well as adjacencies has usually been used as the similarity measure in genome comparisons [2] [3] [4] [5]. Other kinds of genome similarity measures can be looked up in [6] [7] [8].

The exemplar breakpoint distance problem (abbr. EBD) proposed by Sankoff [2] is representative in approaches for finding conserved sets of genes. This problem is given by two genomes, asks to find two respective exemplar gene sequence of the two given genomes, such that the breakpoint distance between the two exemplar gene sequence is minimized. For finding solutions of EBD, Sankoff proposed a branch and bound based algorithm [2], then later, Thach, Tay, and Zhang proposed a divide and conquer based algorithm [4]. Moreover, Zhu presented the tractability for parameterized computation of universal EBD [9]. For two genomes in which one has no repetition of the same gene family and the other has twice repetition genes of the same gene family, EBD has been shown to be NP-Hard by Bryant [5] and APX-Hard by Angibaud *et al.* [10]. Blin *et al.* also showed that no performance factor can be achieved for approximating this problem [1]. If one of the two given genomes has no gene repetition of the same gene family, the other allows genes of the same gene family to repeat at most q times, Fu and Zhang developed an $O(2^m n^{O(1)})$ time algorithm for EBD, where m is the number of the gene families and n is the number of genes in the genomes [11].

If it asks to find two exemplar gene sequence with zero breakpoint distance, then EBD turns to be called the zero exemplar breakpoint distance problem (abbr. ZEBD). ZEBD is NP-Hard [1], even if those two given genomes all admit at most 2 repetition genes of the same gene family. Blin et al. used the color-coding technique to design an algorithm with $O(n2^n)$ time for ZEBD, where n is the number of gene families in the given genomes [1]. They also gave a parameterized algorithm for ZEBD with $O(m2^{2s}s^3)$ time, where m is the number of genes in that given genome with less genes than the other given genome. Jiang proposed a polynomial algorithm for a special version of ZEBD, where one given genome is asked to have no repetition genes of the same gene family and the other genome can admit q repetition genes of the same gene family [12]. Moreover, Zhu and Wang [13] proposed an exact algorithm for ZEBD, for the situation where each of the two given genomes admits at most two repetition genes of the same gene family, with running time $O(n^2 1.86121^n)$.

In this paper, we focus on how to find a longest subsequence of exemplar genes that is conserved in two genomes. We set up an optimization problem model to describe on finding a longest subsequence of exemplar genes that is conserved in two genomes, and name it as the longest common exemplar subsequence problem (abbr. LCES). LCES can be recognized to be NP-hard because ZEBD is NP-hard even if the two given genome each admits at most twice repetitions of the same gene family. Then, if in one of the two given genomes, any two genes of the same gene family are in at most s spans, we propose a dynamic programming algorithm to solve it with $O(s2^s mn)$ space and $O(s4^s mn)$ time, where m, n stand for the gene numbers of those two given genomes.

II. Preliminaries

Let Σ be an alphabet. Each symbol in Σ represents a *gene family*. A *genome* on Σ refers to a sequence of occurrences of the gene families in Σ. Each occurrence of a gene family in a genome is referred to as a *gene*. There can be more than one genes of the same gene family in a genome. Two genes in a genome are referred to as *identical*, if they are of the

This paper is supported by national natural science foundation of China, No. 61472222, 61732009, 61761136017, 61672325.

This is the author's manuscript of the article published in final edited form as:

Zhang, S., Wang, R., Zhu, D., Jiang, H., Feng, H., Guo, J., & Liu, X. (2018). The Longest Common Exemplar Subsequence Problem. 2018 IEEE International Conference on Bioinformatics and Biomedicine (BIBM), 92–95. https://doi.org/10.1109/BIBM.2018.8621304

same gene family. We always use the same symbol as used to represent a gene's family to represent the gene. Thus two genes, say x and y, are identical, if and only if $x = y$. A sequence of genes is referred to as a *subsequence* of a genome, if it can be obtained by deleting some genes (not necessarily consecutive) from the genome. A subsequence of a genome is referred to as an *exemplar subsequence* of the genome, if every gene family in Σ occurs in the subsequence at most once.

Let $A = A[1]$ .. $A[m]$ be a genome on Σ, $C = A[x[1]]$ ... $A[x[p]]$ an exemplar subsequence of A, $1 \leq x[1] < ... < x[p] \leq m$. Then , $A[x[i]] \neq A[x[j]]$ for $x[i] \neq x[j]$. A *common exemplar subsequence* of more than one genomes (or gene sequences) refers to a sequence of genes, which is an exemplar subsequence of each of the genomes. The *length* of a genome (or a gene sequence) A, which is denoted as $|A|$, is the gene number of A. A common exemplar subsequence of more than one genomes is *longest*, if no other common exemplar subsequence of these genomes can be longer than it. The longest common exemplar subsequence problem (abbr. LCES) of two genomes is motivated by finding a set of genes that is conserved in the same order, and can be formalized as,

Instance: Two genomes A, B on Σ.

Object: Find a longest common exemplar subsequence of both A and B.

For an arbitrary genome $A = A[1]$ ... $A[m]$, let $A[i,\ j]$ denote the consecutive subsequence $A[i]$ $A[i+1]$ ... $A[j]$ of A, $1 \leq i \leq j \leq m$. Let X and Y be two gene sequences on Σ. We denote by $X \parallel Y$ the concatenation of X and Y. For example, $A[i,\ j] = A[i,\ j-1] \parallel A[j]$.

Moreover, we refer to $j - i$ for $i \leq j$ as the *span* of $A[i]$ and $A[j]$ in A. Two genes are *adjacent* in a genome if their span is 1 in the genome. The *span* of a genome refers to the maximum span of two identical genes in that genome. A genome is referred to as *s-span* if the span of the genome is s.

Without loss of generality, every gene family in Σ is assumed to occur in both A and B. Moreover, since for two adjacent genes $A[i]$, $A[i+1]$ in A, if $A[i] = A[i+1]$, the deletion of $A[i+1]$ (or $A[i]$) from A doesn't change the set of A's exemplar subsequences, we assume any two adjacent genes in A as well as B are not identical.

III. An algorithm for LCES with s-span

Researches on comparing human and mouse genomes shows that a large number of micro-rearrangements happen to human or mouse genomic sequences [14]. In the algorithm design literatures of genome comparison, the span of two identical genes in a genome has usually been used as a parameter to design efficient algorithms [1] [15]. Since so, we pay attention to s-span genomes to find their common exemplar subsequences.

Assume $A = A[1]$ ... $A[m]$ and $B = B[1]$ ... $B[n]$ serve as an instance of LCES.

A. The primary dynamic programming

All common exemplar subsequences of A and B can be enumerated by dynamic programming, which goes from a set of all common exemplar subsequences of $A[1,\ i]$, $B[1,\ j-1]$, a set of all common exemplar subsequences of $A[1,\ i-1]$, $B[1,\ j]$ and a set of all common exemplar subsequences of $A[1,\ i-1]$, $B[1,\ j-1]$ to a set of all common exemplar subsequences of $A[1,\ i]$, $B[1,\ j]$.

Let $C(i,\ j)$ denote the set of all common exemplar subsequences of $A[1,\ i]$ and $B[1,\ j]$. Since no gene occurs at position 0 in A as well as B, $C(i,\ j)$ is assigned with an empty set for $i = 0$ or $j = 0$.

A common exemplar subsequence of $A[1, i-1]$, $B[1, j]$or $A[1, i]$, $B[1, j-1]$ or $A[1, i-1]$, $B[1, j-1]$ must be a common exemplar subsequence of $A[1, i]$ and $B[1, j]$. Thus $C(i-1,\ j) \subseteq C(i,\ j)$, $C(i-1,\ j-1) \subseteq C(i,\ j)$ and $C(i,\ j-1) \subseteq C(i,\ j)$. Let C denote an arbitrary common exemplar subsequence of $A[1,\ i]$ and $B[1,\ j]$ ($1 \leq i \leq m$ and $1 \leq j \leq n$). Then C can be arrived at from a member in $C(i-1,\ j)$ or $C(i,\ j-1)$ or $C(i-1,\ j-1)$ under two situations.

(1) If $A[i] \neq B[j]$, then $C \in C(i-1, j)$ or $C \in C(i, j-1)$. It follows $C(i-1,\ j) \subseteq C(i,\ j)$ and $C(i,\ j-1) \subseteq C(i,\ j)$ that $C(i,\ j) = C(i-1,\ j) \cup C(i,\ j-1)$.

(2) If $A[i] = B[j]$, let C' be a member in $C(i-1,\ j-1)$, which does not admit any identical gene to $A[i]$, then $C \in C(i-1,\ j)$ or $C \in C(i,\ j-1)$ or $C' \parallel A[i] \in C(i, j)$. Let $C(i-1,\ j-1,\ \overline{A[i]})$ denote the subset of $C(i-1, j-1)$ with all those members in $C(i-1, j-1)$ which admit no identical genes to $A[i]$, $C(i,\ j,\ A[i]) = \{C' \parallel A[i] \mid C' \in C(i-1,\ j-1,\ \overline{A[i]})\}$. Then $C \in C(i,\ j,\ A[i])$. It follows $C(i-1,\ j) \subseteq C(i,\ j)$, $C(i,\ j-1) \subseteq C(i,\ j)$ and $C(i,\ j,\ A[i]) \subseteq C(i,\ j)$ that $C(i,\ j) = C(i-1,\ j) \cup C(i,\ j-1) \cup C(i,\ j,\ A[i])$.

Then $C(i,\ j)$ can be arrived at recursively as in Formula (1). This leads to a dynamic programming based algorithm to find a longest common exemplar subsequence of A, B.

$$C(i,j) = \begin{cases} \emptyset & i{=}0 \text{ or } j{=}0 \\ C(i-1,j) \cup C(i,j-1) & A[i] \neq B[j],\ i \neq 0 \text{ and } j \neq 0 \\ C(i-1,j) \cup C(i,j-1) \cup C(i,j,A[i]) & A[i] = B[j],\ i \neq 0 \text{ and } j \neq 0 \end{cases} \quad (1)$$

A member in $C(i,\ j)$ has at most $|\Sigma|$ genes, each of which can occur in a member in $C(i,\ j)$ or not. It follows that $|C(i,\ j)| \leq 2^{|\Sigma|}$. It takes $O(|\Sigma| 2^{|\Sigma|})$ time to get $C(i,\ j,\ A[i])$ if $A[i] = B[j]$. It also takes $O(|\Sigma| 2^{|\Sigma|})$ time to get $C(i-1,\ j) \cup C(i,\ j-1) \cup C(i,\ j,\ A[i])$ if $A[i] = B[j]$, or $C(i-1,\ j) \cup C(i,\ j-1)$ if $A[i] \neq B[j]$. Thus the time complexity of getting $C(i,\ j)$ from $C(i-1,\ j) \cup C(i,\ j-1) \cup C(i-1,\ j-1)$ is $O(|\Sigma| 2^{|\Sigma|})$. Since $1 \leq i \leq m$, $1 \leq j \leq n$, the time complexity of getting a longest common exemplar subsequence of A and B is $O(nm|\Sigma| 2^{|\Sigma|})$.

B. Gene family set based dynamic programming

To improve the time complexity, an intuitive way is to store less common exemplar subsequences of $A[1, i]$ and $B[1, j]$ than those in $C(i, j)$ in the dynamic programming. To achieve this point, we pay attention to those sets of gene families which

occur both in the members in $C(i, j)$ and $A[i+1, m]$ or $B[j+1, n]$. Let C be a common exemplar subsequence of $A[1, i]$ and $B[1, j]$. A gene family which occurs in C is referred to as *confused*, if it also occurs in $A[i+1, m]$ as well as $B[j+1, n]$. The *confused gene family set* of C refers to the set of all confused gene families which occur in C, and will be denoted as $F(i, j, C)$. Every common exemplar subsequence in $C(i,j)$ is accompanied with a confused gene family set. In fact, if two common exemplar subsequences in $C(i, j)$ have the same confused gene family sets, then we can show it sufficient to give up that one with less genes than the other. That is,

Lemma III.1. *Let for $i \leq m$ and $j \leq n$, C_1 and C_2 be two common exemplar subsequences of $A[1, i]$ and $B[1, j]$ with $F(i, j, C_1) = F(i, j, C_2)$. If $|C_1| \geq |C_2|$, the longest extension of C_1 in $C(m, n)$, must have no less genes than any extension of C_2 in $C(m, n)$.*

By Lemma III.1, to achieve a longest common exemplar subsequence of A and B by dynamic programming, it suffices to maintain a subset of $C(i, j)$ whose common exemplar subsequences admit mutually distinct confused gene family sets. Let for a common exemplar subsequence $C \in C(i,j)$, $f(i, j, C)$ denote the confused gene family set of C. The *confused gene family set collection* of a subset of $C(i, j)$ refers to the collection of the confused gene family sets of all those common exemplar subsequences in the subset of $C(i, j)$. A subset of $C(i, j)$ is referred to as *representative*, if each member in $C(i, j)$ admits the same confused gene family set with a member in the subset, each member in it is the longest over all common exemplar subsequences in $C(i, j)$ with the same confused gene family sets. A subset of $C(i, j)$ is referred to as *minimum*, if every two members in it admit distinct confused gene family sets. A representative subset of $C(i, j)$ is not guaranteed unique, even if it is minimum. Let $CF(i, j)$ be an arbitrary minimum representative subset of $C(i, j)$. A longest common exemplar subsequence of A and B must occur in a minimum representative subset of $C(m,n)$. Instead of getting a minimum representative subset of $C(i, j)$ from $C(i, j)$, we pay attention to getting a minimum representative subset of $C(i, j)$ from $CF(i-1, j)$, $CF(i, j-1)$ and $CF(i-1, j-1)$.

It is trivial for $CF(i, j)$ to be initialized as $\emptyset$, if $i = 0$ or $j = 0$. Then there are two cases to get a minimum representative subset of $C(i, j)$ from $CF(i-1, j)$ and $CF(i, j-1)$.

(1) If $A[i] \neq B[j]$, since $C(i, j) = C(i, j-1) \cup C(i-1, j)$ by Formula (1), then there exists a minimum representative subset of $C(i, j)$ which is a subset of $CF(i, j-1) \cup CF(i-1, j)$. A common exemplar subsequence in $CF(i, j-1)$ and $CF(i-1, j)$ will turn into a member in $C(i, j)$ with an other confused gene family set than it is in $CF(i, j-1)$ and $CF(i-1, j)$. Let $C \in CF(i, j-1)$ (resp. $CF(i-1, j)$). The confused gene family set of C in $C(i, j)$ can be extracted under two subcases.

(1.1) If no identical gene to $A[i]$ (resp. $B[j]$) occurs in $A[i+1, m]$ as well as $B[j+1, n]$, and $f(i-1, j, C)$ (resp. $f(i, j-1, C)$) contains the gene family of $A[i]$ (resp. $B[j]$), then $f(i, j, C) = f(i-1, j, C)$ (resp. $f(i, j-1, C)) \setminus \{A[i]\}$ (resp. $\setminus B[j]$), where $A[i]$ (resp. $B[j]$) represents a gene as well as its gene family.

(1.2) Otherwise, $f(i, j, C) = f(i-1, j, C)$ (resp. $f(i, j-1, C)$).

Let $C(i, j, CF(i, j-1))$ (resp. $C(i, j, CF(i-1, j))$) denote the subset of $C(i, j)$ with none other than those member in $CF(i-1, j)$ (resp. $CF(i, j-1)$), $F(i, j, CF(i-1, j))$ (resp. $F(i, j, CF(i, j-1))$) the confused gene family set collection of those members in $C(i, j, CF(i-1, j))$ (resp. $C(i, j, CF(i, j-1))$). We set a subroutine named as SM($F(x, y)$, G) to remove those gene families in a gene family set G from every set in $F(x, y)$, and output the sets in $F(x, y)$ in exclusion of the gene families in G. Thus for $x = i-1$, $y = j$ or $x = i$, $y = j-1$, $F(i, j, CF(x, y))$ can be expressed in Formula (2).

A minimum representative subset of $C(i, j)$, say $CF(i, j)$, can be got by checking every two members in $C(i, j, CF(i, j-1)) \cup C(i, j, CF(i-1, j))$ for if they admit the same confused gene family sets, and if yes, removing the shorter one of them.

(2) If $A[i] = B[j]$, then by Formula (1), $C(i, j) = C(i, j-1) \cup C(i-1, j) \cup C(i, j, A[i])$. Let $CF(i, j, A[i])$ denote a minimum representative subset of $C(i, j, A[i])$. There exists a minimum representative subset of $C(i, j)$ which is a subset of $CF(i, j-1) \cup CF(i-1, j) \cup CF(i, j, A[i])$. A minimum representative subset of $C(i, j, A[i])$ can be identified as,

Lemma III.2. *The set $\{C' \parallel A[i] \mid C' \in CF(i-1, j-1\ \overline{A[i]})\}$ is a minimum representative subset of $C(i, j, A[i])$.*

Let $C' \in CF(i-1, j-1, \overline{A[i]})$. In consideration of getting $f(i, j, C' \parallel A[i])$, it suffices to check for $f(i, j, C' \parallel A[i]) = f(i-1, j-1, C')$ or $f(i-1, j-1, C') \cup \{A[i]\}$. If no identical gene to $A[i]$ occurs in $A[i+1, m]$ or $B[j+1, n]$, then $f(i, j, C' \parallel A[i]) = f(i-1, j-1, C')$. If an identical gene to $A[i]$ occurs in both $A[i+1, m]$ and $B[j+1, n]$, then $f(i, j, C' \parallel A[i]) = f(i-1, j-1, C') \cup \{A[i]\}$.

Then, the confused gene family set of a member $C = C' \parallel A[i] \in CF(i, j, A[i])$, can be expressed as,

$$f(i, j, C) = \begin{cases} f(i, j, C') & A[i] \notin A[i+1, m] \vee B[j+1, n]; \\ f(i, j, C') \cup \{A[i]\} & A[i] \in A[i+1, m] \wedge B[j+1, n]. \end{cases} \quad (3)$$

Let $F(i, j, A[i])$ denote the confused gene family set collection of $CF(i, j, A[i])$. Then,

$$F(i, j, A[i]) = \{f(i, j, C) | C \in CF(i, j, A[i])\}. \quad (4)$$

A minimum representative subset of $C(i, j)$, say $CF(i, j)$, can be got by checking every two members in $CF(i, j-1) \cup CF(i-1, j) \cup CF(i, j, A[i])$ for if they admit the same confused gene family sets in $C(i, j)$, and if yes, removing the shorter one of them.

Aiming to use less storage space in the dynamic programming, we select to maintain the confused gene family sets and lengths of those common exemplar subsequences in $CF(i, j)$

$$F(i,j,CF(x,y)) = \begin{cases} F(x,y) & A[i] \in A[i+1,m] \wedge B[j+1,n], B[j] \in A[i+1,m] \wedge B[j+1,n]; \\ SM(F(x,y),\{A[i]\}) & A[i] \notin A[i+1,m] \wedge B[j+1,n], B[j] \in A[i+1,m] \wedge B[j+1,n]; \\ SM(F(x,y),\{B[j]\}) & A[i] \in A[i+1,m] \wedge B[j+1,n], B[j] \notin A[i+1,m] \wedge B[j+1,n]; \\ SM(F(x,y),\{A[i],B[j]\}) & A[i] \notin A[i+1,m] \wedge B[j+1,n], B[j] \notin A[i+1,m] \wedge B[j+1,n]; \end{cases} \quad (2)$$

instead of $CF(i,j)$ itself. Let $f = f(i,j,C)$ be the confused gene family set of $C \in C(i,j)$. Then the length of C is referred to as the *CES length* of f and denoted as $L(f)$. The CES length of an arbitrary confused gene family set f in $F(i,\ j,\ CF(i-1,j))$, $F(i,\ j,\ CF(i,j-1))$, or $F(i,\ j,\ A[i])$, can be computed recursively as in Formula (5).

$$L(f) = \begin{cases} L(f(i-1,j,C)) & f = f(i,j,C) \in F(i,j,CF(i-1,j)); \\ L(f(i,j-1,C)) & f = f(i,j,C) \in F(i,j,CF(i,j-1)); \\ L(f(i-1,j-1,C')+1 & f = f(i,j,C) \in F(i,j,A[i]), C = C' \parallel A[i]. \end{cases} \quad (5)$$

Let $F(i,\ j)$ denote the confused gene family set collection of $CF(i,\ j)$. To extract $F(i,j)$ from $F(i,\ j,\ CF(i-1,j)) \cup F(i,\ j,\ CF(i,j-1))$, or $F(i,\ j,\ CF(i-1,j)) \cup F(i,\ j,\ CF(i,j-1)) \cup F(i,j,A[i])$, it suffices to check every two members in $F(i,\ j,\ CF(i-1,j)) \cup F(i,\ j,\ CF(i,j-1))$ or $F(i,j,CF(i-1,j)) \cup F(i,j,CF(i,j-1)) \cup F(i,j,A[i])$ for if they are equal to each other, and if yes, removing from them that one with small CES length than the other. Let F denote a confused gene family set collection. We use a subroutine named as U(F) to find a minimum representative subset of F. Finally, $F(i,\ j)$ can be expressed recursively in Formula (6).

$$F(i,j) = \begin{cases} U(F(i,j,CF(i-1,j)) \cup F(i,j,CF(i,j-1))) & A[i] \neq B[j]; \\ U(F(i,j,CF(i-1,j)) \cup F(i,j,CF(i,j-1)) \cup F(i,j,A[i])) & A[i] = B[j] \end{cases} \quad (6)$$

The confused gene family set collections of $A[1,i]$ and $B[1,j]$, $F(i,j)$ for $0 \leq i \leq m$, $0 \leq j \leq n$, can be got by Algorithm 1.

Algorithm 1 The dynamic programming for LCES

```
F(i, 0) = F(0, j) ← {∅}, 0 ≤ i ≤ m, 0 ≤ j ≤ n;
for i from 1 to m do
    for j from 1 to n do
        Get F(i, j) by (2), (3), (4), (5), (6);
    end for
end for
```

Then by tracing back from $F(m,n)$ to $F(i,\ j)$ where $i = 0$ or $j = 0$, a common exemplar subsequence of A and B can be got.

If at least one of the two given genomes is s-span, then Algorithm 1 can get a longest common exemplar subsequence of A and B in $O(s4^s mn)$ time with $O(s2^s mn)$ space.

IV. Conclusion

In this paper, we proposed a dynamic programming algorithm for LCES with space complexity $O(s2^s mn)$ and time complexity $O(s4^s mn)$ if one of the two given genomes is s-span, where m, n stand for the gene numbers of those two given genomes. Our algorithm can be extended to find longest common exemplar subsequences of more than one genomes.

References

[1] G. Blin, G. Fertin, F. Sikora, and S. Vialette, "The exemplarbreakpoint-distance for non-trivial genomes cannot be approximated," *Journal of Thermal Analysis & Calorimetry*, vol. 36, no. 1, pp. 1–7, 2009.

[2] D. Sankoff, "Genome rearrangement with gene families," *Bioinformatics*, vol. 15, no. 11, pp. 909–917, 1999.

[3] Z. Chen, B. Fu, and B. Zhu, *The Approximability of the Exemplar Breakpoint Distance Problem*. Springer Berlin Heidelberg, 2006.

[4] C. T. Nguyen, Y. C. Tay, and L. Zhang, "Divide-and-conquer approach for the exemplar breakpoint distance." *Bioinformatics*, vol. 21, no. 10, pp. 2171–2176, 2005.

[5] D. Bryant, "The complexity of calculating exemplar distances," *Computational Biology*, vol. 1, pp. 207–211, 2000.

[6] G. Blin and R. Rizzi, *Conserved Interval Distance Computation Between Non-trivial Genomes*. Springer Berlin Heidelberg, 2005.

[7] A. Bergeron and J. Stoye, "On the similarity of sets of permutations and its applications to genome comparison," in *International Computing and Combinatorics Conference*, 2003, pp. 68–79.

[8] S. Bérard, A. Bergeron, and C. Chauve, "Conservation of combinatorial structures in evolution scenarios," *Lecture Notes in Computer Science*, vol. 3388, pp. 1–14, 2004.

[9] B. Zhu, "Approximability and fixed-parameter tractability for the exemplar genomic distance problems," *Lecture Notes in Computer Science*, vol. 5532, pp. 71–80, 2009.

[10] S. Angibaud, G. Fertin, and I. Rusu, "On the approximability of comparing genomes with duplicates," in *International Workshop on Algorithms and Computation*. Springer, 2008, pp. 34–45.

[11] B. Fu and L. Zhang, "A polynomial algebra method for computing exemplar breakpoint distance," in *International Symposium on Bioinformatics Research and Applications*. Springer, 2011, pp. 297–305.

[12] M. Jiang, "The zero exemplar distance problem," *Journal of Computational Biology A Journal of Computational Molecular Cell Biology*, vol. 18, no. 9, p. 1077, 2011.

[13] D. Zhu and L. Wang, "An exact algorithm for the zero exemplar breakpoint distance problem," *IEEE/ACM Transactions on Computational Biology & Bioinformatics*, vol. 10, no. 6, pp. 1469–1477, 2014.

[14] P. Pevzner and G. Tesler, "Genome rearrangements in mammalian evolution: lessons from human and mouse genomes." *Genome Research*, vol. 13, no. 1, pp. 37–45, 2003.

[15] Z. Wei and D. Zhu, "A dynamic programming algorithm for unsigned (1,2)-exemplar breakpoint distance problem with span constraint," in *Sixth International Conference on Business Intelligence and Financial Engineering*, 2014, pp. 39–43.