New Genome Sequence Detection via Natural Vector Convex Hull Method

Ruzhang Zhao, Shaojun Pei, and Stephen Shing-Toung Yau

Abstract—It remains challenging how to find existing but undiscovered genome sequence mutations or predict potential genome sequence mutations based on real sequence data. Motivated by this, we develop approaches to detect new, undiscovered genome sequences. Because discovering new genome sequences through biological experiments is resource-intensive, we want to achieve the new genome sequence detection task mathematically. However, little literature tells us how to detect new, undiscovered genome sequence mutations mathematically. We form a new framework based on natural vector convex hull method that conducts alignment-free sequence analysis. Our newly developed two approaches, Random-permutation Algorithm with Penalty (RAP) and Random-permutation Algorithm with Penalty and COstrained Search (RAPCOS), use the geometry properties captured by natural vectors. In our experiment, we discover a mathematically new human immunodeficiency virus (HIV) genome sequence using some real HIV genome sequences. Significantly, the proposed methods are applicable to solve the new genome sequence detection challenge and have many good properties, such as robustness, rapid convergence, and fast computation.

Index Terms—Natural vector, convex hull, new genome sequence, random permutation, RAP method

1 Introduction

With the rapid development of biological technologies, more and more genome sequences are measured. Genome sequences store the information of many important physiological processes of life such as reproduction, cell division, and protein synthesis. It is also an internal factor that determines the adaption, evolution, phenotypic variance of life. For example, the genome sequences of human immunodeficiency virus (HIV) help to understand why HIV leads to acquired immunodeficiency syndrome (AIDS). Meanwhile, genome sequences can serve as identification for different species. The genome sequences of HIV assist scientists to distinguish HIV from other species. To distinguish genome sequences of different species, comparative sequence analyses mentioned in [1], [2], [3], [4], [5], are applied to measure the similarity among different genome sequences in the past decades. Some alignment-free sequence comparison methods are also developed [6]. The natural vector method proposed by Deng *et al.* [7] associates a vector with a genome sequence to describe the distribution (i.e., the locations and number of occurrences) for each nucleotide in the genome sequence.

Although many genome sequences have been measured, for many species, only part of genome sequences are known. For example, as an RNA virus, HIV mutates very quickly, so many genome sequence mutations are not learned. If we can find some existing, undiscovered genome sequence mutations or potential future sequence mutations, the scientists can conduct drug research on the virus more comprehensively. Due to the limits of resources and technologies, we propose to detect new, undiscovered genome sequences mathematically. We call the challenge new genome sequence detection. The mathematically detected new genome sequences can also support further biological experiments. For example, newly detected HIV genome sequence may help to understand unknown properties of HIV. Generating new genome sequence is a popular topic. For example, real sequence data is used to simulates new offspring genome [8]; people study genetic variation by simulating genome sequences [9], [10]. But it is not well studied how to use real sequence data to find existing but undiscovered sequence mutations or predict potential sequence mutations. Hence, we are inspired to develop new method.

The motivations of our work are from two aspects. First, we can discover existing but unlearned genome sequence mutations. Many genome sequences have been well studied. However, some existing mutations are still not learned by biologists. We propose to detect the existing, undiscovered genome sequence mutations mathematically. This detection is based on real genome sequences and uses the genome sequence space scheme [11], [12], which is widely used for protein, gene or genome, like Koonin. *et al.* [13]. Also, we can predict potential genome sequence mutations. For example, the mutation of HIV is extremely fast [14]. If we can predict the possible sequence mutations, it may help the drug research. Second, in metagenomics [15], the genomic analysis uses microbial DNA extracted directly from communities in environmental samples. Some existing genome sequences in the communities may not be fully extracted. Our proposed mathematical genome sequence detection can be a good supplement for current metagenomic analysis.

In this work, we solve the new genome sequence detection problem by exploring the properties shown in natural vector method [7]. It is proven that the natural vector holds

• *The authors are with the Department of Mathematical Science, Tsinghua University, Beijing 100084, China. E-mail: rzhao@jhu.edu, psj17@mails.tsinghua.edu.cn, yau@uic.edu.*

Manuscript received 23 Nov. 2019; revised 22 May 2020; accepted 22 Nov. 2020. Date of publication 25 Nov. 2020; date of current version 3 June 2022.
(Corresponding author: Stephen Shing-Toung Yau.)
Digital Object Identifier no. 10.1109/TCBB.2020.3040706

one-to-one relationship with genome sequence [7], which can be represented in a finite dimensional space. The natural vector method achieves good performance in sequence comparison, and displays the natural relationships among genome sequence of different species [16], which is called natural vector convex hull method. Convex hull is a widely used approach in computational biology [17], [18], [19]. Actually, convex hull is the smallest convex polygon formed by a set of points, where the convex polygon encloses all of the points in the set. Natural vector method is extended applying the geometry properties of natural vectors [20], [16]. Experiments demonstrate that the natural vector convex hull method separates natural vectors whose corresponding genome sequences come from different species.

We know the genome sequences with natural vectors in the same convex hull share similar genetic properties and are highly likely in the same genome group. Accordingly, we seek new natural vectors in convex hull to detect new genome sequence. If new genome sequences whose natural vectors fall in the convex hull formed by known ones, the new ones are likely from the same species as the known ones. Our goal is to use natural vector convex hull method to find new genome sequences based on several known, real genome sequences.

Here, we propose our heuristic algorithms, Random-permutation Algorithm with Penalty (RAP, henceforth) and Random-permutation Algorithm with Penalty and COstrained Search (RAPCOS, for short). RAP and RAPCOS enjoy great properties like robustness of convergence, rapid convergence and fast computation.

The rest of paper are organized as follows. Section 2 reviews the related work, defines the new genome sequence detection problem and states the problem formulation. In section 3, we show the new genome sequence detection problem is NP-hard. Section 4 states the newly proposed RAP and RAPCOS methods. In Section 5, we explore the detailed process of solving the detection problem. Section 6 includes the experiments to show the effectiveness and robustness of our proposed methods. The methods are concluded and discussed in Section 7.

2 Related Work

In this part, we review some related work including natural vector method and natural vector convex hull method. Our work is inspired by these two methods.

2.1 Natural Vector Method

We first introduce some denotations applied throughout the paper: bold letters as vectors; $\mathbb{K} = \{\mathrm{A}, \mathrm{C}, \mathrm{G}, \mathrm{T}\}$ as the set of four nucleotides; $\mathbf{S} = (s_1, s_2, \ldots, s_n)$ as a genome sequence of length n ($s_i \in \mathbb{K}, i = 1, 2, \ldots, n$); n_k ($k \in \mathbb{K}$) as the number of nucleotide k; $s_{[k][i]}$ as the distance from the first nucleotide (regarded as origin) to the ith nucleotide k; μ_k and D_2^k (Equation (1)) as the mean position of nucleotide k and the second-order normalized central moment of nucleotide k, respectively.

$$\mu_k = \sum_{i=1}^{n_k} \frac{s_{[k][i]}}{n_k}, \; D_2^k = \sum_{i=1}^{n_k} \frac{(s_{[k][i]} - \mu_k)^2}{n_k n}. \tag{1}$$

Then, the natural vector of genome sequence $\mathbf{S}$ is defined:

$$\mathbf{v} = (n_{\mathrm{A}}, n_{\mathrm{C}}, n_{\mathrm{G}}, n_{\mathrm{T}}, \mu_{\mathrm{A}}, \mu_{\mathrm{C}}, \mu_{\mathrm{G}}, \mu_{\mathrm{T}}, D_2^{\mathrm{A}}, D_2^{\mathrm{C}}, D_2^{\mathrm{G}}, D_2^{\mathrm{T}})'. \tag{2}$$

An example for natural vector: ACACGTGT is computed as $(2, 2, 2, 2, 2, 3, 6, 7, \frac{1}{8}, \frac{1}{8}, \frac{1}{8}, \frac{1}{8})'$. There are two important properties of natural vectors:

$$\sum_{k \in \mathbb{K}} n_k = n, \; \sum_{k \in \mathbb{K}} n_k \mu_k = \frac{1}{2}(n^2 + n).$$

Actually, natural vector can include higher-order normalized central moments like

$$D_j^k = \sum_{i=1}^{n_k} \frac{(s_{[k][i]} - \mu_k)^j}{(n_k n)^{j-1}}, k \in \mathbb{K}, \; j = 2, 3, \ldots, n_k. \tag{3}$$

But natural vector in Equation (2) is the most widely used form, which is also applied in this paper. After displaying genome sequences as natural vectors in a finite dimensional space, we measure the distance between two natural vectors $\mathbf{v}^{(1)}$, $\mathbf{v}^{(2)}$ to demonstrate the difference between genome sequences $\mathbf{S}^{(1)}$, $\mathbf{S}^{(2)}$. In particular, the euclidean distance is used, where $\| \cdot \|$ is the euclidean norm for vector.

$$d(\mathbf{S}^{(1)}, \mathbf{S}^{(2)}) = d(\mathbf{v}^{(1)}, \mathbf{v}^{(2)}) = \|\mathbf{v}^{(1)} - \mathbf{v}^{(2)}\|. \tag{4}$$

2.2 Natural Vector Convex Hull Method

Convex hull is a widely discussed topic in computational biological [17], [18], [19], which is defined as the smallest convex set where all the points are inside. Many algorithms are designed to find convex hull, like gift wrapping algorithm [21], incremental insertion algorithm [22], randomized incremental algorithm [23].

Tian *et al.* [16] and Zhao *et al.* [20] show the natural vector convex hull of one species is disjoint from the convex hull of the others. Applying natural vector to distinguish genome sequences of different species is referred as natural vector convex hull method. With fixed number of natural vectors, natural vector convex hulls are actually convex polyhedrons.

2.3 Problem Formulation

In this section, we define the concept of new genome sequence detection challenge and the importance to solve this problem.

Biological experiments for sequencing requires massive resources and advanced technologies to discover new genome sequence, in particular, new genome sequence mutations. New genome sequence detection is a mathematical problem that detects undiscovered genome sequences mathematically based on known, real genome sequence data. For example, part of HIV genome sequences are known. Mathematically, we can use real HIV genome sequences to find undiscovered genome sequence mutations or predict new ones. We also need to show the obtained new genome sequences share similar properties with known ones. In particular, we use the mathematical properties of genome sequences shown by natural vector method [7] for new genome sequence detection problem.

We define new genome sequence detection problem as finding a new genome sequence based on known, real genome sequences using natural vector method. In detail, in new genome sequence detection problem, first, real genome sequence data from one species are given. The corresponding natural vectors of these sequences form a convex hull. Second, find a new genome sequence whose natural vector falls into the convex hull. Here, we formulate the new genome sequence detection problem. There are N genome sequences and N corresponding natural vectors (Equation (2)): $\mathbf{S}^{(i)}, \mathbf{v}^{(i)}, i = 1, 2, \ldots, N$, where the superscript (i) denotes the ith sequence or vector. N given natural vectors form a convex hull $\mathrm{Conv}(\{\mathbf{v}^{(i)}\}_{i=1}^{N})$. The goal of new genome sequence detection is to find a new genome sequence $\mathbf{S}^{\text{new}}$, whose corresponding natural vector $\mathbf{v}^{\text{new}}$ falls into the convex hull $\mathrm{Conv}(\{\mathbf{v}^{(i)}\}_{i=1}^{N})$, Equation (5),

$$\mathbf{v}^{\text{new}} = \sum_{i=1}^{N} \alpha_i \mathbf{v}^{(i)},\ \alpha_i \geq 0\,,\ \sum_{i=1}^{N} \alpha_i = 1, \tag{5}$$

where α_i is the coefficients of convex combination.

3 New Genome Sequence Detection is NP-Hard

In this section, we illustrate why the new genome sequence detection challenge is NP-hard (non-deterministic polynomial-time hardness). We first give Lemma 1.

Lemma 1. *The new genome detection problem is an integer programming problem.*

The proof of Lemma 1 is in supplementary material, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TCBB.2020.3040706.

Theorem 1. *It is NP-hard to solve the new genome detection problem as an optimization problem.*

Proof. It is proven that integer programming is NP-hard [24] and only linear integer programming has general algorithms to solve. For example, zero-one integer linear programming is widely explored in [25], [26]. Without the linear constrains, the zero-one integer programming can only be solved by heuristic methods. There are no general algorithms to solve such kind of NP-hard problems.

With Lemma 1, the new genome sequence detection is integer programming, so it is NP-hard. □

Stemming from Theorem 1, we design the two algorithms, RAP and RAPCOS in the view of heuristic, randomized algorithm.

4 Methods

Theorem 1 reveals that the new genome sequence detection problem is a challenge to solve an integer programming problem, which is NP-hard. In this section, we propose algorithms to deal with the challenge efficiently.

Natural vector (Equation (2)) uses the information about counts of four nucleotides and their positions in the genome sequence. In particular, the mean position and the second-order normalized central moment in Equation (1) uncover the distributions of the nucleotides. In Equation (5), not all convex combinations can result in new natural vectors. There are some requirements to meet for a natural vector. For example, the first four terms in natural vector are $n_{\text{A}}, n_{\text{C}}, n_{\text{G}}, n_{\text{T}}$, which must be positive integers. In geometry, a point with integer coordinates in the convex hull is called a lattice point. Equation (5) tells us all the possible new natural vectors $\mathbf{v}^{\text{new}}$ should have lattice points as their first four terms. To determine the solution to Equation (5), the subproblem (Equation (6)) should be solved first.

$$n_k^{\text{new}} = \sum_{i=1}^{N} \alpha_i n_k^{(i)},\ k \in \mathbb{K},\ \alpha_i \geq 0,\ \sum_{i=1}^{N} \alpha_i = 1, \tag{6}$$

where n_k^{new} and $n_k^{(i)}$ are from $\mathbf{v}^{\text{new}}$ and $\mathbf{v}$, separately. Beginning with this point, we first solve subproblem (Equation (6)). With the integer points fixed for $\mathbf{v}^{\text{new}}$, the numbers of $\{\mathrm{A}, \mathrm{C}, \mathrm{G}, \mathrm{T}\}$ are constant. Then, the detection problem is simplified as a permutation problem. With a proper permutation approach, we can explore when the new natural vector falls into the convex hull $\mathrm{Conv}(\{\mathbf{v}^{(i)}\}_{i=1}^{N})$.

The first method gives a general solution to address optimization problem in sequence space. Different choices of loss function can deal with different problem settings. The second method focuses on how to formulate a mapping from a natural vector to a genome sequence more accurately compared with the first method. To heuristically solve the new genome sequence detection, we propose Random-permutation Algorithm with Penalty (RAP, henceforth). The algorithm efficiently mapping natural vector to genome sequence is called Random-permutation Algorithm with Penalty and COnstrained Search (RAPCOS, for short). In particular, the method RAPCOS is an improved version of RAP with respect to problems with known natural vector.

The following two definitions are used in the process of method development.

Definition 1 (Target natural vector). *Target natural vector is the target vector we approximate, where the vector is natural vector.*

Definition 2 (Fuzzy target natural vector). *Fuzzy target natural vector is the target vector we approximate, where the vector is not necessarily natural vector.*

4.1 Random-Permutation With Penalty

The proposed approach, Random-permutation Algorithm with Penalty which is called RAP for short, is a heuristic method for solving computationally hard optimization detection problem in sequence space. RAP belongs to the family of random local search [27], where new value of loss function is compared with the previous one. Because the local search is random, the results cannot be guaranteed. In particular, in our detection problem, the new sequence is recorded only when the loss decreases. That is to say, with the goal of minimizing loss function, each local search is adopted only if the loss function drops, which is simple but efficient for the detection problem. Besides the random local search, we use penalty to accelerate the random choice in each step of local search.

After solving subproblem (Equation (6)) and fixing the counts of nucleotides, we need to solve a permutation problem. In each local search, we aim to find a new sequence permuted based on the result of previous step. Herzog *et al.* [28] propose that any permutation can be represented by 2 cycles. Accordingly, our detection problem is simplified as a 2 cycle permutation problem at each step. In each local search of 2 cycle permutation, with the loss function going down, the new sequence replaces the previous one to be the current sequence. We conduct the process of random 2 cycle permutation until the loss value is smaller than a preset limit.

Since the first four terms in natural vector are fixed after solving problem, the initialization of genome sequence can be randomly given. Then, in each step of random 2 cycle permutation, two steps are adopted. First, two nucleotides are randomly selected from the nucleotides $\mathbb{K} = \{\mathrm{A}, \mathrm{C}, \mathrm{G}, \mathrm{T}\}$ and denoted as $k, q \in \mathbb{K}, k \neq q$. Second, two positions are selected from the positions of k, q, respectively. Then, the 2 cycle permutation happens when the loss function drops after the permutation between the two selected positions.

To select two nucleotides more efficiently, the penalty probability is considered to offer different probabilities to different nucleotides. With respect to a certain nucleotide, the larger the difference between current natural vector and target natural vector is, the more likely the nucleotide will be chosen for permutation. Since the counts of with respect to a certain nucleotide are fixed, the mean position and the second-order central normalized moment should be measured simultaneously.

The penalty probability is defined in Equation (7). The probability of $k \in \mathbb{K} = \{\mathrm{A}, \mathrm{C}, \mathrm{G}, \mathrm{T}\}$ is

$$\mathrm{P}_k(\mathbf{S}^{(1)}, \mathbf{S}^{(2)}) = \frac{(|\mu_k^{\mathbf{S}^{(1)}} - \mu_k^{\mathbf{S}^{(2)}}| + |D_2^{k,\mathbf{S}^{(1)}} - D_2^{k,\mathbf{S}^{(2)}}|)}{\sum_{j \in \mathbb{K}} (|\mu_j^{\mathbf{S}^{(1)}} - \mu_j^{\mathbf{S}^{(2)}}| + |D_2^{j,\mathbf{S}^{(1)}} - D_2^{j,\mathbf{S}^{(2)}}|)}, \tag{7}$$

where $\mathbf{S}^{(1)}$ and $\mathbf{S}^{(2)}$ are the two genome sequences, and scalars with superscript $\mathbf{S}^{(i)}$ are the μ_k, D_2^k from $\mathbf{S}^{(i)}$.

The **R**andom-permutation **A**lgorithm with **P**enalty(**RAP**, henceforth) is displayed in Algorithm 1. In particular, the loss function is denoted as "loss", where one choice of the loss function is $\mathrm{loss}(\mathbf{S}^{\mathrm{seq}}) = d(\mathbf{v}^{\mathrm{seq}}, \mathbf{v}^{\mathrm{tg}})$, where $\mathbf{v}^{\mathrm{tg}}$ is the target natural vector, $\mathbf{S}^{\mathrm{seq}}$ and $\mathbf{v}^{\mathrm{seq}}$ are the sequence and natural vector as variable. The distance function d is shown in Equation (4). Denote M_{Iter} as the maximal iteration number; ϵ as a preset limit.

RAP can also be applied to fuzzy target natural vector (Definition 2). Because fuzzy target natural vector is not required to be natural vector, the approximation ends up with a sequence whose natural vector is sufficiently close to the fuzzy target one.

4.2 Random-Permutation With Penalty and Constrained Search

A particular kind of loss function we need to minimize is the distance between the natural vector of our searching sequence and the target natural vector. In each step of 2 cycle permutation, two nucleotides $k, q \in \mathbb{K}, k \neq q$ are selected and μ_k, μ_q, D_2^k and D_2^q change after the permutation. If the four values get closer to the target natural vector than the previous one simultaneously, the new values will be adopted and the algorithm will move to the next round of local search. So, we analyze how the natural vector changes after the permutation.

Algorithm 1. Random-Permutation Algorithm With Penalty (RAP)

```
M_Iter: maximal iteration number; ε: preset limit.
Initialization Step : Randomly generate a sequence $\mathbf{S}^{\mathrm{seq}}$ with $n_k, k \in \mathbb{K}$ requirement. And let Iter = 0.
Iteration Step
 1: Get the positions of [A, C, G, T] based on $\mathbf{S}^{\mathrm{seq}}$.
 2: while $\mathrm{loss}(\mathbf{S}^{\mathrm{seq}}) > \epsilon$ and iter $< M_{\mathrm{Iter}}$ do
 3:   Iter = Iter + 1.
 4:   Get $[\mathrm{P}_{\mathrm{A}}, \mathrm{P}_{\mathrm{C}}, \mathrm{P}_{\mathrm{G}}, \mathrm{P}_{\mathrm{T}}](\mathbf{S}^{\mathrm{seq}}, \mathbf{S}^{\mathrm{tg}})$ based on Equation (7).
 5:   Let $\mathbf{S}^{\mathrm{new}}$ = $\mathbf{S}^{\mathrm{seq}}$.
 6:   while $\mathrm{loss}(\mathbf{S}^{\mathrm{new}}) \geq \mathrm{loss}(\mathbf{S}^{\mathrm{seq}})$ do
 7:     Randomly Select two nucleotides, $k, q \in \mathbb{K}, k \neq q$ based on Probabilities $[\mathrm{P}_{\mathrm{A}}, \mathrm{P}_{\mathrm{C}}, \mathrm{P}_{\mathrm{G}}, \mathrm{P}_{\mathrm{T}}]$.
 8:     Randomly Select a position from the positions of $k$ and $q$, separately: $\mathrm{pos}_k$ and $\mathrm{pos}_q$.
 9:     Get $\mathbf{S}^{\mathrm{new}}$ from $\mathbf{S}^{\mathrm{seq}}$ (Do a permutation between $\mathrm{pos}_k$ and $\mathrm{pos}_q$).
10:   end while
11:   Remove $\mathrm{pos}_k$ from positions of $k$ and remove $\mathrm{pos}_q$ from positions of $q$.
12:   Add $\mathrm{pos}_q$ to positions of $k$ and add $\mathrm{pos}_k$ to positions of $q$.
13:   Let $\mathbf{S}^{\mathrm{seq}} = \mathbf{S}^{\mathrm{new}}$.
14: end while
15: The $\mathbf{S}^{\mathrm{seq}}$ is the sequence which minimizes our loss function with the preset limit.
```

Lemma 2 has restrained the relation between $\mu_{\mathrm{A}}, \mu_{\mathrm{C}}, \mu_{\mathrm{G}}, \mu_{\mathrm{T}}$.

Lemma 2. *For a genome sequence* $\mathbf{S}$, *the corresponding natural vector* $\mathbf{v}$ *is in Equation (2).*

If for $k \in \mathbb{K}$, $\mu_k < \mu_k^{\mathrm{tg}}$, *there must exist* $q \in \mathbb{K}, q \neq k$, *where* $\mu_q > \mu_q^{\mathrm{tg}}$ *and vice versa.*

In the following theorems, we discuss how permutation affects the change of loss, where the loss is the distance between the current sequence and the target sequence. In Theorem 2, four cases are discussed. And we use the following case as an example for denotations. For a genome sequence $\mathbf{S}$ and a target sequence $\mathbf{S}^{\mathrm{tg}}$, the corresponding natural vectors $\mathbf{v}$ and $\mathbf{v}^{\mathrm{tg}}$ are in Equation (2). We show a case LG for comparing $\mathbf{v}$ and $\mathbf{v}^{\mathrm{tg}}$. For a randomly selected nucleotide $k \in \mathbb{K}$.

Case LG: $\mu_k - \mu_k^{\mathrm{tg}} < 0$, $D_2^k - D_2^{k,\mathrm{tg}} > 0$, where we use G and L to represent "Greater than 0" and "Less than 0", respectively. For example, case LG represents the first inequality($\mu_k - \mu_k^{\mathrm{tg}}$) is less than 0 and the second inequality ($D_2^k - D_2^{k,\mathrm{tg}}$) is greater than 0.

In total, we have four cases denoted as LL, LG, GL, GG.

Theorem 2. *For a genome sequence* $\mathbf{S}$ *and a target sequence* $\mathbf{S}^{\mathrm{tg}}$, *the corresponding natural vectors* $\mathbf{v}$ *and* $\mathbf{v}^{\mathrm{tg}}$ *are in Equation (2). We show four cases* LL, LG, GL, GG *for comparing* $\mathbf{v}$ *and* $\mathbf{v}^{\mathrm{tg}}$. *The updated natural vector* $\mathbf{v}^{\mathrm{new}}$ *should meet the following requirement to make* $|\mu_k^{\mathrm{new}} - \mu_k^{\mathrm{tg}}| < |\mu_k - \mu_k^{\mathrm{tg}}|$, $|D_2^{k,\mathrm{new}} - D_2^{k,\mathrm{tg}}| < |D_2^k - D_2^{k,\mathrm{tg}}|$. *Denote* $s_{[k][j]}$ *to be the selected position and* $s_{[k][j]}^{\mathrm{new}}$ *to be the new position. Denote* $\tau_{k,j} = n_k(n_k - 1)^{-1}(2\mu_k - 2s_{[k][j]}n_k^{-1}) - s_{[k][j]}$.

LL: $s^{\text{new}}_{[k][j]} > \max\{\tau_{k,j}, s_{[k][j]}\}$.
LG: *If* $s_{[k][j]} < \mu_k$, *then*, $s_{[k][j]} < s^{\text{new}}_{[k][j]} < \tau_{k,j}$.
GL: $s^{\text{new}}_{[k][j]} < \min\{\tau_{k,j}, s_{[k][j]}\}$.
GG: *If* $s_{[k][j]} > \mu_k$, , $\tau_{k,j} < s^{\text{new}}_{[k][j]} < s_{[k][j]}$.

Theorem 2 provides better a local search strategy compared with random search. However, Theorem 2 only considers the selected one nucleotide $k \in \mathbb{K}$, it is not guaranteed that $\|\mathbf{v}^{\text{new}} - \mathbf{v}^{\text{tg}}\| < \|\mathbf{v} - \mathbf{v}^{\text{tg}}\|$. Hence, we show more strict constraints in Theorem 3. Under the same settings for $\mathbf{S}, \mathbf{v}$ and $\mathbf{S}^{\text{tg}}, \mathbf{v}^{\text{tg}}$, we consider a case LG&GL for two different nucleotides $k, q \in \mathbb{K}$. And the k, q are guaranteed to be selected following Lemma 2.

Case LG&GL: $\mu_k - \mu_k^{\text{tg}} < 0$, $D_2^k - D_2^{k,\text{tg}} > 0$, $\mu_q - \mu_q^{\text{tg}} > 0$, $D_2^q - D_2^{q,\text{tg}} < 0$, where we also use G and L to represent "Greater than 0" and "Less than 0", respectively. For example, case LG&GL represents the first inequality ($\mu_k - \mu_k^{\text{tg}}$) is less than 0, the second inequality ($D_2^k - D_2^{k,\text{tg}}$) is greater than 0, the third inequality ($\mu_q - \mu_q^{\text{tg}}$) is greater than 0 and the fourth inequality ($D_2^q - D_2^{q,\text{tg}}$) is less than 0.

Therefore, we have eight cases denoted as LL&GL, LL&GG, LG&GL, LG&GG, GL&LL, GL&LG, GG&LL, GG&LG.

In Theorem 3, eight detailed cases are discussed for two selected nucleotides k, q.

For a genome sequence $\mathbf{S}$, the corresponding natural vector is $\mathbf{v}$ in Equation (2). For example, when $\mu_k < \mu_k^{\text{tg}}$, since a position of A is replaced with a position of C, if $\mu_q > \mu_q^{\text{tg}}$, after the permutation, μ_k^{new} and μ_q^{new} are closer to μ_k^{tg} and μ_q^{tg} than before, respectively. Strict conditions are considered in Theorem 3 based on Theorem 2. Denote $s_{[k][j]}$ to be the selected position and $s^{\text{new}}_{[k][j]}$ to be the new position. Since k and q are two different nucleotides to be permuted, we have $s^{\text{new}}_{[k][j]} = s_{[q][j]}$, $s^{\text{new}}_{[q][j]} = s_{[k][j]}$.

Theorem 3. *For a genome sequence* $\mathbf{S}$ *and a target sequence* $\mathbf{S}^{\text{tg}}$, *the corresponding natural vectors* $\mathbf{v}$ *and* $\mathbf{v}^{\text{tg}}$ *are in Equation (2). We show eight cases* LL&GL, LL&GG, LG&GL, LG&GG, GL&LL, GL&LG, GG&LL, GG&LG *for comparing* $\mathbf{v}$ *and* $\mathbf{v}^{\text{tg}}$ *with respect to two selected different nucleotides* k, q. *The updated natural vector* $\mathbf{v}^{\text{new}}$ *should meet the following requirement to make* $\|\mathbf{v}^{\text{new}} - \mathbf{v}^{\text{tg}}\| < \|\mathbf{v} - \mathbf{v}^{\text{tg}}\|$. *Denote* $h_k(x) = (n_k - 1)^{-1}[2n_k\mu_k - (n_k + 1)x]$, $g_k(x) = (n_k + 1)^{-1}[2n_k\mu_k - (n_k - 1)x]$, *where* $k \in \mathbb{K}$ *can be replaced by* $q \in \mathbb{K}$. *And denote* $\rho_1 = (n_k + n_q)^{-1}\left[(n_q + 1)n_k\mu_k - (n_k - 1)n_q\mu_q\right]$.

LL&GL: *If* $\mu_k < \mu_q$ *&* $\rho_1 < s_{[k][j]} < \mu_q$, *then* $\max\{s_{[k][j]}, h_k(s_{[k][j]})\} < s^{\text{new}}_{[k][j]} < g_q(s_{[k][j]})$.
LL&GG: $s^{\text{new}}_{[k][j]} > \max\{s_{[k][j]}, h_k(s_{[k][j]}), g_q(s_{[k][j]}), \mu_q\}$.
LG&GL: *If* $s_{[k][j]} < \min\{\mu_k, \mu_q\}$, $s_{[k][j]} < s^{\text{new}}_{[k][j]} < \min\{h_k(s_{[k][j]}), g_q(s_{[k][j]})\}$.
LG&GG: *If* $s_{[k][j]} < \min\{\mu_k, \rho_1, g_k(\mu_q)\}$, *then* $\max\{s_{[k][j]}, \mu_q, g_q(s_{[k][j]})\} < s^{\text{new}}_{[k][j]} < h_k(s_{[k][j]})$.
GL&LL: *If* $\mu_k > \mu_q$ *&* $\mu_q < s_{[k][j]} < \rho_1$, *then* $g_q(s_{[k][j]}) < s^{\text{new}}_{[k][j]} < \min\{s_{[k][j]}, h_k(s_{[k][j]})\}$.
GL&LG: $s^{\text{new}}_{[k][j]} < \min\{s_{[k][j]}, h_k(s_{[k][j]}), g_q(s_{[k][j]}), \mu_q\}$.
GG&LL: *If* $s_{[k][j]} > \max\{\mu_k, \mu_q\}$, *then* $\max\{h_k(s_{[k][j]}), g_q(s_{[k][j]})\} < s^{\text{new}}_{[k][j]} < s_{[k][j]}$.
GG&LG: *If* $s_{[k][j]} > \max\{\mu_k, \rho_1, g_k(\mu_q)\}$ *then* $h_k(s_{[k][j]}) < s^{\text{new}}_{[k][j]} < \min\{s_{[k][j]}, \mu_q, g_q(s_{[k][j]})\}$.

We also extend Theorem 2 to higher-order natural vector (larger than 2) cases in the supplementary material, available online. The proofs of theorems are also supplemented, available online.

RAP (Algorithm 1) can be applied to approximate target natural vector via minimizing the distance between the natural vector of searching sequence and target natural vector. According to Theorem 3, RAP method can be accelerated by the constrained search. Review the eight cases mentioned in Theorem 3. In each step of 2 cycle permutation, with two selected nucleotides to be k and q, based on Theorem 3, it is guaranteed that $|\mu_k^{\text{new}} - \mu_k^{\text{tg}}| < |\mu_k - \mu_k^{\text{tg}}|$, $|\mu_q^{\text{new}} - \mu_q^{\text{tg}}| < |\mu_q - \mu_q^{\text{tg}}|$, $|D_2^{k,\text{new}} - D_2^{k,\text{tg}}| < |D_2^k - D_2^{k,\text{tg}}|$, $|D_2^{q,\text{new}} - D_2^{q,\text{tg}}| < |D_2^q - D_2^{q,\text{tg}}|$. The loss decreases after each step of permutation, $\text{loss}(\mathbf{S}^{\text{new}}) = \|\mathbf{v}^{\text{new}} - \mathbf{v}^{\text{tg}}\| < \text{loss}(\mathbf{S}) = \|\mathbf{v} - \mathbf{v}^{\text{tg}}\|$.

The penalty in Equation (7) is also applied here to accelerate the choice of two nucleotides in each step of 2 cycle permutation.

According to Lemma 2, there exist $k, q \in \mathbb{K}, k \neq q$, where $(\mu_k - \mu_k^{\text{tg}})(\mu_q - \mu_q^{\text{tg}}) < 0$. We can first search the constraints in Theorem 3 based on the mean position condition just like Theorem 2.

The constrained search in Theorem 3 is strict to some extent. In particular, when the loss is really close to the minimum, there might be no positions that meet the requirement. Hence, we make use of the original strategy of RAP method for random permutation when the constrained search fails to get results.

The **R**andom-permutation **A**lgorithm with **P**enalty and **CO**nstrained **S**earch (**RAPCOS** for short, henceforth) is displayed in Algorithm 2. The loss function is also $\text{loss}(\mathbf{S}^{\text{new}}) = \|\mathbf{v}^{\text{new}} - \mathbf{v}^{\text{tg}}\|$.

5 Solve the Detection Problem

There are two different solutions to new genome sequence detection problem. Both start from natural vector convex hull and end up finding a new genome sequence. One deal with sequence directly (Sequence-direct Solution), while the other uses natural vector as an intermedium (Vector-mid Solution).

The first path (Sequence-direct Solution) deals with genome sequences directly. When a genome sequence whose corresponding natural vector falls into the given convex hull, we obtain the solution to the detection problem directly, Fig. 1. We optimize over sequence space to find the solution to detection problem. The second solution (Vector-mid Solution) uses the natural vector falling in the natural vector convex hull as an intermedium. We optimize over vector space in the process. After obtaining a natural vector, we map the natural vector back to a genome sequence, Fig. 2. The final process requires optimization over sequence space. Moreover, in both solutions, we need corresponding loss functions to update genome sequence.

5.1 Solve the Detection Problem by the Sequence-Direct Solution

The Sequence-direct Solution (Fig. 1) deals with the problem efficiently by finding genome sequences whose corresponding natural vectors fall into the convex hull

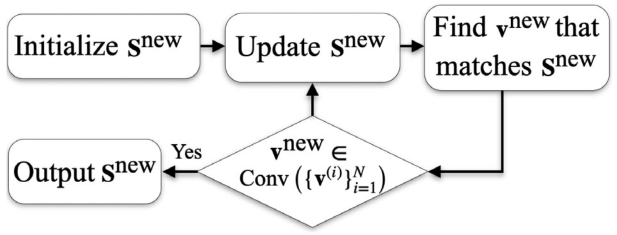
Initialize $\mathbf{S}^{new}$
Update $\mathbf{S}^{new}$
Find $\mathbf{v}^{new}$ that matches $\mathbf{S}^{new}$
$\mathbf{v}^{new} \in \text{Conv}(\{\mathbf{v}^{(i)}\}_{i=1}^N)$
Yes
Output $\mathbf{S}^{new}$

Fig. 1. The workflow of the sequence-direct solution (Deal with sequence directly).

directly. This approach does not require heavy computation in the process of determining the existence of natural vectors compared with the Vector-mid Solution discussed in the next Section 5.2.

Algorithm 2. Random-permutation Algorithm With Penalty and Constrained Search (RAPCOS)

```
$M_{\text{Iter}}$: maximal iteration number; $\epsilon$: preset limit.
**Initialization Step** : Randomly generate a sequence $\mathbf{S}^{\text{seq}}$ with $n_k, k \in \mathbb{K}$ requirement. And let Iter = 0.
**Iteration Step**
 1: **while** loss($\mathbf{S}^{\text{seq}}$) $>\epsilon$ and iter $< M_{\text{Iter}}$ **do**
 2:   Iter = Iter + 1.
 3:   Get the positions of [A, C, G, T] based on $\mathbf{S}^{\text{seq}}$.
 4:   Get $[\text{P}_\text{A}, \text{P}_\text{C}, \text{P}_\text{G}, \text{P}_\text{T}](\mathbf{S}^{\text{seq}}, \mathbf{S}^{\text{tg}})$ based on Equation (7).
 5:   Let $\mathbf{S}^{\text{new}} = \mathbf{S}^{\text{seq}}$.
 6:   **while** loss($\mathbf{S}^{\text{new}}$) $\geq$ loss($\mathbf{S}^{\text{seq}}$) **do**
 7:     Randomly select a nucleotide $k \in \mathbb{K}$ based on probability $[\text{P}_\text{A}, \text{P}_\text{C}, \text{P}_\text{G}, \text{P}_\text{T}]$.
 8:     Randomly select $\text{pos}_k$ from the positions of $k$.
 9:     Denote $\mathbb{K}_c = \mathbb{K} \backslash \{k\}$.
10:     **for** $k_1 \in \mathbb{K}_c$ **do**:
11:       **if** $(\mu_k - \mu_k^{\text{tg}})(\mu_{k_1} - \mu_{k_1}^{\text{tg}}) > 0$ **then**:
12:         $\mathbb{K}_c = \mathbb{K}_c \backslash \{k_1\}$.
13:       **end if**
14:     **end for**
15:     **while** $\mathbb{K} \neq \emptyset$ **do**
16:       Randomly select a nucleotide $q \in \mathbb{K}_c$.
17:       Find the relation of $D_2^k$ and $D_2^q$ according to
          Theorem 3 and constrain the search region.
18:       **if** The requirement for constrained search region is satisfied **then**
19:         Randomly select $\text{pos}_q$ from the constrained search region. Break.
20:       **end if**
21:         $\mathbb{K}_c = \mathbb{K}_c \backslash \{q\}$.
22:       **end while**
23:     **end while**
24:   Remove $\text{pos}_k$ from positions of $k$ and remove $\text{pos}_q$ from positions of $q$.
25:   Add $\text{pos}_q$ to positions of $k$ and add $\text{pos}_k$ to positions of $q$.
26:   Get $\mathbf{S}^{\text{new}}$ from $\mathbf{S}^{\text{seq}}$ (Permute $\text{pos}_k$ and $\text{pos}_q$).
27:   $\mathbf{S}^{\text{seq}} = \mathbf{S}^{\text{new}}$.
28: **end while**
29: The $\mathbf{S}^{\text{seq}}$ is the sequence which minimizes our loss function with preset limit.
```

Because we do not focus on finding target natural vector, the loss function chosen in Algorithms 1 and 2 cannot be used. To obtain the sequence, it is straightforward to measure the distance from natural vector to the convex hull and use it as loss function. However, in high dimensional space, the computation of the distance to convex hull is non-trivial. Moreover, much computation is wasted since the loss function needs to be computed after each time of permutation.

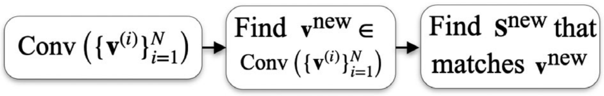
$\text{Conv}(\{\mathbf{v}^{(i)}\}_{i=1}^N)$
Find $\mathbf{v}^{new} \in \text{Conv}(\{\mathbf{v}^{(i)}\}_{i=1}^N)$
Find $\mathbf{S}^{new}$ that matches $\mathbf{v}^{new}$

Fig. 2. The workflow of the Vector-mid Solution (use natural vector as intermedium).

Here, we design a new loss function for more efficient computation. With finite number of points in the set, the convex hull is a convex hyperpolyhedron. The distance from a point to the convex hull is actually the minimal distance among the ones from natural vector to all the surfaces. However, if the number of points are more than the dimension, the computation is arduous. For example, when the dimension is 12 and there are 101 points, there are $\binom{101}{12}$ surfaces which need be searched for the minimal distance. However, when the dimension is 12 and there are 13 points, there are $\binom{13}{12} = 13$ surfaces which should be searched. This motivates us to consider reducing the number of vertices to reduce computation.

Accordingly, the simplest convex hyperpolyhedron, the hyperpyramid is considered to construct loss function. In a n-dimensional space, $n+1$ points can form a hyperpyramid with $n+1$ hyperplanes, where the dimension of the hyperplanes is $n-1$.

5.1.1 The Equation of Hyperplane

We provide the equation to form hyperplane using n vertices in a n-dimensional space. For example, in a 3-dimensional space, $\mathbf{v}_1 = (x_1, y_1, z_1)'$, $\mathbf{v}_2 = (x_2, y_2, z_2)'$, $\mathbf{v}_3 = (x_3, y_3, z_3)'$ are used to determine a 2-dimensional surface $\mathbf{w}'\mathbf{v} + b = 0$.

Combine three equations $\mathbf{w}'\mathbf{v}_i + b_i = 0,\ i = 1, 2, 3,$ the surface is determined by

$$\left| \begin{pmatrix} x - x_1 & y - y_1 & z - z_1 \\ x_2 - x_1 & y_2 - y_1 & z_2 - z_1 \\ x_3 - x_1 & y_3 - y_1 & z_3 - z_1 \end{pmatrix} \right| = 0,$$

where the left part computes the determinant. Similarly, in a n-dimensional space, the corresponding matrix form for the $n-1$-dimensional hyperplane is:

$$\left| \begin{pmatrix} v^1 - v_1^1 & v^2 - v_1^2 & \cdots & v^n - v_1^n \\ v_2^1 - v_1^1 & v_2^2 - v_1^2 & \cdots & v_2^n - v_1^n \\ \vdots & \vdots & \ddots & \vdots \\ v_n^1 - v_1^1 & v_n^2 - v_1^2 & \cdots & v_n^n - v_1^n \end{pmatrix} \right| = 0, \tag{8}$$

where $\mathbf{v} = (v^1, v^2, \ldots, v^n)'$ represents any point in a n-dimensional space and $\mathbf{v}_i = (v_i^1, v_i^2, \ldots, v_i^n)', i = 1, 2, \ldots, n$ represent n given vertices in n-dimensional space.

The distance from a point to the hyperplane, $\mathbf{w}'\mathbf{v} + b = 0$ is $|\mathbf{w}'\mathbf{v} + b|/\|\mathbf{w}\|$. From Equation (8), $\|\mathbf{w}\|^2$ is equal to the quadratic sum of the coefficients of $v^1, v^2, \ldots, v^n$.

5.1.2 Design the Loss Function

In the setting of the Sequence-direct Solution (Fig. 1), the target natural vector is unknown. Here, we design a new loss function to measure the distance from natural vector to hyperpyramid.

The dimension of considered natural vectors is 12. First, if we pick 13 vertices in convex hull and form a hyperpyramid (13 vertices form the simplest convex hull in 12-dimensional space), any natural vector falling into the hyperpyramid is actually in the convex hull. Second, consider 12 vertices out of 13. The hyperplane splits the 12-dimensional space into two subspaces. Compare the relative position of the new natural vector and the remaining vertex. If they are at different subspaces, the new natural vector can never be in the hyperpyramid. After considering all the 13 formed hyperplanes, if natural vector and the remaining vertex are always at the same subspace, we claim the natural vector fall into the hyperpyramid, furthermore, in the convex hull. Hence, the loss function is formed in:

$$\text{loss}(\mathbf{v}) = \sum_{i=1}^{13} \text{ReLU}[-\mathbf{1}' \frac{(\mathbf{w}_i'\mathbf{v} + b_i)'}{\|\mathbf{w}_i\|} \,\text{sgn}(\mathbf{w}_i'\mathbf{v} + b_i)], \tag{9}$$

where $\mathbf{v}$ is the natural vector of new sequence, $\mathbf{w}_i'\mathbf{v} + b_i = 0$ is the equation of the hyperplane Equation (8), ReLU stands for rectified linear unit, $\text{ReLU}(x) = \max(0, x)$ and sgn is sign function. With loss equal to 0, the natural vector falls into the hyperpyramid.

RAP (Algorithm 1) can naturally use loss function (Equation (9)) to perform optimization. But the probabilities for penalty in Equation (7) require target natural vector. The mean of vertices is used here as fuzzy target natural vector (Definition 2). Because fuzzy target natural vector is not necessarily natural vector, RAPCOS cannot be applied in the Sequence-direct Solution (Fig. 1).

The loss function (Equation (9)) has 0 minimum. Because it measures the distance to a convex hull, we only need find the genome sequence which makes the loss function sufficiently small instead of exactly zero to make the natural vector fall into the convex hull. The change from optimization to suboptimization can reduce much computation.

5.1.3 Form the Hyperpyramid in the Convex Hull

The last step to construct the loss function in Equation (9) is to select vertices in convex hull. In particular, the 12-dimensional natural vectors need 13 vertices to form a hyperpyramid. If the vertices are selected out of the convex hull, the algorithm will fail to detect points inside the convex hull. Our strategy applies principal component analysis (PCA) [29] to form new vertices.

Based on the subproblem (Equation (6)), we fix the first four terms to be integers and the new hyperpyramid is formed based on the fixed integers. In details, with $n_{\text{A}}, n_{\text{C}}, n_{\text{G}}, n_{\text{T}}$ fixed, we need to find 13 vertices to form the hyperpyramid in 12-dimensional space. Hence, PCA is applied to reduce the dimension with respect to the sample size in Algorithm 3.

After PCA with respect to the sample size N, principal components (PC_i, $1 \leq i \leq N$) are obtained in our example. However, we cannot guarantee any of the principal components fall into the convex hull. Further detailed strategies are adopted to obtain the final 13 vertices of our hyperpyramid, where the first 12 principal components are used as directions instead of vectors.

First, choose the first 12 principal components (PC_i, $1 \leq i \leq 12$). Second, for each principal component, we use $\text{PC}_i[1:4]$ to denote the first four dimensions. Then, form a line segment in the 4-dimensional space, whose direction is $\text{PC}_i[1:4]$, and $(n_{\text{A}}, n_{\text{C}}, n_{\text{G}}, n_{\text{T}})$ should be on the line segment. Third, pick the head and end of the line segment with certain distance from $(n_{\text{A}}, n_{\text{C}}, n_{\text{G}}, n_{\text{T}})$, where $(n_{\text{A}}, n_{\text{C}}, n_{\text{G}}, n_{\text{T}})$ falls on the connection line between head and end. Fourth, we constrain the generated heads and ends to fall into the convex hull. The `inhull`[1] function implemented in Matlab is easy to test whether a point falls into a convex hull.

Finally, we obtain 24 points, $\{\text{head}_i\}_{i=1}^{12}$ and $\{\text{end}_i\}_{i=1}^{12}$ for the head and end of each linear segment. We choose all 12 heads $\{\text{head}_i\}_{i=1}^{12}$ and end_1 (generated from the first principal component). The choice of the 13 vertices $\{\text{vertex}_i\}_{i=1}^{13}$ can guarantee the point $(n_{\text{A}}, n_{\text{C}}, n_{\text{G}}, n_{\text{T}})$ falls into the convex hull of the first four dimensions of vertices $\{\text{vertex}_i[1:4]\}_{i=1}^{13}$.

The process of producing the new vertices for hyperpyramid is described in Algorithm 3.

Algorithm 3. Process of Producing Vertices for Hyperpyramid

```
N: the number of sequences. The first four elements of natural
vectors $n_A, n_C, n_G, n_T$ are fixed.
Initialization Step : Compute the natural vectors of $N$ sequences: $\{\mathbf{v}^{(i)}\}_{i=1}^{N}$.
1: Apply PCA to $\{\mathbf{v}^{(i)}\}_{i=1}^{N}$ with respect to sample size.
2: Pick $\{\text{PC}_i\}_{i=1}^{12}$ from $\{\text{PC}_i\}_{i=1}^{N}$.
3: For each $1 \leq i \leq 12$, find $\text{head}_i$, $\text{end}_i$ of the $i$th linear segment.
   The direction from $\text{head}_i$ to $\text{end}_i$ is $\text{PC}_i$ and $(n_A, n_C, n_G, n_T)$
   falls on the connection line between $\text{head}_i[1:4]$, $\text{end}_i[1:4]$.
4: Make $\{\text{head}_i\}_{i=1}^{12}$ and $\text{end}_1$ fall into the convex hull.
5: $\{\text{head}_i\}_{i=1}^{12}$ and $\text{end}_1$ are the 13 vertices to form
   hyperpyramid.
```

5.2 Solve the Detection Problem by the Vector-Mid Solution

In this section, we solve the new genome sequence detection problem using the Vector-mid Solution (Fig. 2). Stemming from high dimensional space, Fig. 2 seeks one potential natural vector in the given convex hull. Then mapping function from the natural vector space to the corresponding sequence space is developed. Finally, we obtain a sequence from the found potential natural vector.

It has been widely discussed as a basic algorithm how to find point in a convex hull. However, to find the natural vector with momental conditions in a convex hull is not discussed.

Here, we explore how to test whether a given new vector in space is a natural vector and define the natural vector space. Because the mapping from a genome sequence to the high dimensional space is strict and one-to-one, we can define the natural vector space as a subspace of the high dimensional space.

1. https://www.mathworks.com/matlabcentral/fileexchange/10226-inhull

Inspired by the subproblem (Equation (6)), we construct the natural vector space step by step. First, the first four terms, the counts of nucleotides, are integers. Second, based on the first constraints, the mean positions need to meet requirement in Equation (10). Third, based on the first and second constraints, the second-order normalized central moments need to meet requirement in Equation (10). Then, for higher order central moments (Equation (3)), natural vector needs to meet the requirements in Equation (12).

In details, for genome sequences and corresponding natural vectors $\mathbf{S}^{(i)}, \mathbf{v}^{(i)}, i = 1, 2, \ldots, N$ in Equation (2), we formulate the vectors $\mathbf{n}_k, \boldsymbol{\mu}_k, \mathbf{D}_2^k$. For $k \in \mathbb{K}$,

$$\begin{aligned} \mathbf{n}_k &= (n_k^{(1)}, n_k^{(2)}, \ldots, n_k^{(N)})', \\ \boldsymbol{\mu}_k &= (\mu_k^{(1)}, \mu_k^{(2)}, \ldots, \mu_k^{(N)})', \\ \mathbf{D}_2^k &= (D_2^{k,(1)}, D_2^{k,(2)}, \ldots, D_2^{k,(N)})', \end{aligned}$$

and we want to find $\mathbf{v}^{\mathrm{ch}}$ in the convex hull(ch), i.e.,

$$\mathbf{v}^{\mathrm{ch}} \in \mathrm{Conv}(\{\mathbf{v}^{(i)}\}_{i=1}^N) = \left\{ \sum_{i=1}^N \alpha_i \mathbf{v}^{(i)} | \alpha_i \geq 0, \sum_{i=1}^N \alpha_i = 1 \right\},$$

where $n_k^{\mathrm{ch}} = \boldsymbol{\alpha}' \mathbf{n}_k, \mu_k^{\mathrm{ch}} = \boldsymbol{\alpha}' \boldsymbol{\mu}_k, D_2^{k,\mathrm{ch}} = \boldsymbol{\alpha}' \mathbf{D}_2^k$, $k \in \mathbb{K}$. $\mathbf{v}^{\mathrm{ch}}$ needs to satisfy the constraints in Equation (10). Denote $n^{\mathrm{ch}} = \sum_{k \in \mathbb{K}} n_k^{\mathrm{ch}}$.

$$\sum_{k \in \mathbb{K}} n_k^{\mathrm{ch}} \mu_k^{\mathrm{ch}} = \sum_{i=1}^{n^{\mathrm{ch}}} i, \ \ \sum_{k \in \mathbb{K}} \sum_{i=1}^{n_k^{\mathrm{ch}}} (s_{[k][i]}^{\mathrm{ch}})^2 = \sum_{i=1}^{n^{\mathrm{ch}}} i^2, \tag{10}$$

where the second constraint of Equation (10) is extended as:

$$\sum_{k \in \mathbb{K}} \sum_{i=1}^{n_k^{\mathrm{ch}}} (s_{[k][i]}^{\mathrm{ch}})^2 = \sum_{k \in \mathbb{K}} \Big[n_k^{\mathrm{ch}} n^{\mathrm{ch}} D_2^{k,\mathrm{ch}} + n_k^{\mathrm{ch}} (\mu_k^{\mathrm{ch}})^2 \Big]. \tag{11}$$

Besides the constraints in Equation (10), for higher order central moments (Equation (3)), $\mathbf{v}^{\mathrm{ch}}$ needs to meet the requirements in Equation (12), where we denote $D_j^{k,\mathrm{ch}} = \boldsymbol{\alpha}' \mathbf{D}_j^k$, $k \in \mathbb{K}$.

$$\sum_{k \in \mathbb{K}} \sum_{i=1}^{n_k^{\mathrm{ch}}} (s_{[k][i]}^{\mathrm{ch}})^j = \sum_{i=1}^{n^{\mathrm{ch}}} i^j, j = 2, 3, \ldots, n_k, \tag{12}$$

where the left term in Equation (12) can be expanded using $n_k^{\mathrm{ch}}, \mu_k^{\mathrm{ch}}, D_j^{k,\mathrm{ch}}, j = 2, 3, \ldots, n_k, k \in \mathbb{K}$, where each term is also a function of $\boldsymbol{\alpha}$. Denote $F(\boldsymbol{\alpha}, j), j = 1, 2, \ldots, n_k, k \in \mathbb{K}$. For example, $F(\boldsymbol{\alpha}, 1) = \sum_{k \in \mathbb{K}} n_k^{\mathrm{ch}} \mu_k^{\mathrm{ch}}$ and $F(\boldsymbol{\alpha}, 2) = [\sum_{k \in \mathbb{K}} n_k^{\mathrm{ch}} n^{\mathrm{ch}} D_2^{k,\mathrm{ch}} + n_k^{\mathrm{ch}} (\mu_k^{\mathrm{ch}})^2]$ (right hand side in Equation (11)). The step by step process to explore natural vector space is as follows. Assume we have explored from the counts of nucleotides to jth order central moment, where corresponding equations in Equations (10) and (12) are satisfied. Then, $F(\boldsymbol{\alpha}, j+1)$ varies with $\boldsymbol{\alpha}$ under the constraints. The minimum and maximum of $F(\boldsymbol{\alpha}, j+1)$ can be obtained. If $\sum_{i=1}^{n^{\mathrm{ch}}} i^{j+1}$ falls between the minimum and maximum, we claim there exits natural vector satisfying the constraints of $(j+1)$th order central moment. The details are provided in supplemental material, available online.

In conclusion, the natural vector space should meet the requirements for each central moments. We denote the natural vector space using the counts, the mean positions and the second-order central moments of nucleotides in Equation (13). And leave the natural vector space with higher order central moments in the supplementary material, available online.

We denote the natural vector space as $\mathcal{NV}$. For natural vector $\mathbf{v}$ in Equation (2), denote $n = \sum_{k \in \mathbb{K}} n_k$. Then, $\mathcal{NV}$ is defined in:

$$\begin{aligned} \mathcal{NV} := \{ &\mathbf{v} = (n_{\mathrm{A}}, n_{\mathrm{C}}, n_{\mathrm{G}}, n_{\mathrm{T}}, \mu_{\mathrm{A}}, \mu_{\mathrm{C}}, \mu_{\mathrm{G}}, \mu_{\mathrm{T}}, D_2^{\mathrm{A}}, D_2^{\mathrm{C}}, D_2^{\mathrm{G}}, D_2^{\mathrm{T}})' | \\ &\sum_{k \in \mathbb{K}} n_k \mu_k = \sum_{j=1}^n j, \sum_{k \in \mathbb{K}} \Big[n_k n D_2^k + n_k (\mu_k)^2 \Big] = \sum_{j=1}^n j^2 \}. \end{aligned} \tag{13}$$

The natural vector space $\mathcal{NV}$ constrains the space to be searched, which restrains the space in the convex hull for potential natural vectors.

In the process of searching for potential natural vectors, the constrained conditions are Equations (10) and (12). After obtaining natural vector, the searching of sequence based on known natural vector is quite applicable. In particular, RAP (Algorithm 1) and RAPCOS (Algorithm 2) methods are designed efficiently to solve the problem with target natural vector (Definition 1). And the loss function can be used as the distance between target natural vector and the natural vector of current sequence.

6 Experiments

In our experiments, the given genome sequences are 101 HIV genome sequences, which are collected from `HIV database`.[2]

RAP and RAPCOS (`code`[3]) are both well designed with loss function. However, different loss functions affect the performance of our randomized local search algorithms. As we shown in our experiments, RAP method is suitable for loss function that measures the distance to the convex hull, like loss function (Equation (9)), where RAP performs much better than RAPCOS. In another case, we consider the loss function that measures the distance from the natural vector of searching sequence to target natural vector (Definition 1). RAPCOS performs better than RAP with such kind of loss function, because RAPCOS is an improved version of RAP designed for this loss function.

RAP and RAPCOS show their performance on different tasks. Also, we analyze the convergence of RAP with three different initialization.

In each experiment, the losses and the numbers of permutation are recorded every five times when the losses decrease.

6.1 Results When Loss Function With Target Natural Vector

In this task, we use the loss function that measures the distance from the natural vector of searching sequence to target natural vector. The preset limit for the convergence is set as 10^{-2} and the maximal iteration number is 10^4. We compare the performance of RAP and RAPCOS methods on the task. In particular, we focus on whether the two methods

2. https://www.hiv.lanl.gov/content/index
3. https://github.com/RuzhangZhao/NewGenomeSeqDetection

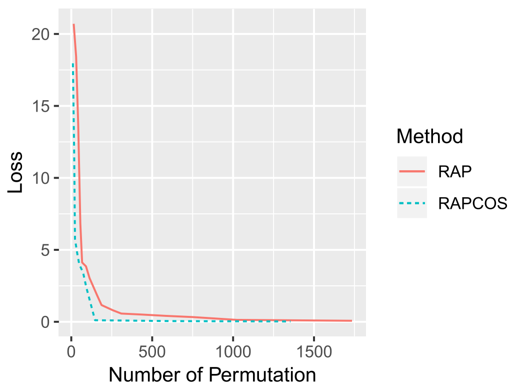

Fig. 3. The convergence of RAP and RAPCOS based on target sequence (target_1), loss function is based on the distance between natural vector and target natural vector.

converge. Also, the rate of convergence is the essential part to show the advantage of one method over the other.

We design three sequences with different lengths. In each sequence, there are four nucleotides $\{A, C, G, T\}$. The numbers of the four nucleotides are $[100, 100, 100, 100]$, separately, for the first sequence. For the second sequence, the numbers are $[1000, 1000, 1000, 1000]$, respectively. We choose the numbers of $\{A, C, G, T\}$ close to real application in the third sequence. The numbers are $[2976, 1386, 1776, 1892]$, separately.

The final sequence we aim to approximate is called target sequence. For each settings, we randomly initialize a sequence with the numbers of nucleotides the same as the target sequence. The target natural vector is computed based on the target sequence. Then, our task is to approximate the target sequence. Our implementation uses another randomly initialized sequence with a certain distance to the target natural vector.

We denote target_i and $\mathrm{start}_i, i = 1, 2, 3$ to be the target sequences and corresponding start sequences. Denote Loss_0 and $\mathrm{Loss}_{\mathrm{final}}$ as initial loss and final loss for each sequence. Denote $\mathrm{Time}_{\mathrm{perm}}$ as the number of permutation of convergence. For the first sequence with length of 400, the distance from the natural vector of starting sequence (start_1) to the target natural vector (target_1) is 25.78.

It is shown that the both method converge more slowly with the reduction of loss. In particular, when the loss is closer to 0, the convergence is much slower. The convergence figure of the comparison between RAP and RAPCOS on the first sequence (target_1, start_1) is displayed in Fig. 3, which shows RAPCOS converges faster than RAP.

For the second sequence with length of 4000, the distance from the natural vector of starting sequence (start_2) to the target natural vector (target_2) is 98.49. The third sequence has the numbers of $\{A, C, G, T\}$ that are closet to the real application of genome sequence. The distance from the natural vector of starting sequence (start_3) to the target natural vector (target_3) is 136.63. The convergence figures of RAP and RAPCOS for the second/ third sequence ($\mathrm{target}_2/\mathrm{target}_3$) are shown in the supplementary material, available online. The details of experiment are shown in Table 1.

In RAPCOS (Algorithm 2), there are some cases when all the eight cases of Algorithm 2 are not satisfied. Then, random selection (RAP) is carried out to continue the random permutation process. Moreover, RAPCOS is an improved version designed for the loss function measuring the distance from natural vector to the target natural vector but the constrains of RAPCOS are more strict than RAP. RAP is used to make RAPCOS more complete when the current case fails to meet any of eight cases in RAPCOS.

TABLE 1
Summary of the Experiments With Different Target Sequences for RAP and RAPCOS

Method	Name of Seq	Length	Loss_0	$\mathrm{Loss}_{\mathrm{final}}$	$\mathrm{Time}_{\mathrm{perm}}$
RAP	target_1, start_1	400	25.78	0.019	4974
RAP	target_2, start_2	4000	98.49	0.003	8745
RAP	target_3, start_3	8030	136.6	0.062	2723
RAPCOS	target_1, start_1	400	25.78	0.023	1355
RAPCOS	target_2, start_2	4000	98.49	0.038	1381
RAPCOS	target_3, start_3	8030	136.6	0.062	1153

1 Loss_0 *is the initial loss for the sequence*
2 $\mathrm{Loss}_{\mathrm{final}}$ *is the final loss when the convergence stops.*
3 $\mathrm{Time}_{\mathrm{perm}}$ *is the permutation count when the convergence stops.*

We record the number when any of RAPCOS eight cases are used for sequence update. In details, for the first sequence (target_1, start_1) with 400 length, there are totally 460 out of 2024 permutations applying the eight cases of RAPCOS, where the proportion is about 22.73 percent. Then, for the second sequence (target_2, start_2) with 4000 length, there are 402 out of 1381 permutations using the eight cases, where there are 29.11 percent permutations. Finally, for the third sequence (target_3, start_3), 301 out of 1153 permutations are carried out based on the eight cases of RAPCOS, where the proportion is about 26.11 percent.

In conclusion, there are about 20 to 30 percent permutations that apply the eight cases of RAPCOS. The remaining permutation uses RAP. However, the true reason for the great performance of RAPCOS is exactly the part applying the eight cases. When permutation is selected from the constrained search region of RAPCOS, the loss certainly decreases. The reduction is guaranteed by Theorem 3. This is also the reason why RAPCOS performs better than RAP with such kind of loss function with target natural vector.

6.2 Results When Loss Function is Distance to Convex Hull

In this part, the task we use to test the performance of RAP is the loss function measuring the distance to the convex hull in Equation (9). The loss function is used in the Sequence-direct Solution (Fig. 1). RAP works well for the task, while for RAPCOS, because target natural vector is required, RAPCOS is not suitable for this task. We compare the performance of RAP and RAPCOS methods on the task. The convergence of RAPCOS is not good, so we just show the reduction of loss for RAPCOS method where the numbers of permutation are limited by the numbers of permutation of RAP. As we shown in the following three cases, the convergence of RAP is great and the rate is fast.

The given 101 sequences are all HIV sequences. We randomly initialize three sequences with the fixed numbers of $n_A, n_C, n_G.n_T$, which are [2976,1386,1776,1892], respectively. The fixed numbers solve the subproblem (Equation (6)). We use three different initialization of sequences, where the three sequences are denoted as $\mathrm{seq}_1, \mathrm{seq}_2, \mathrm{seq}_3$. The preset limit for the convergence of RAP is 10^{-1}.

TABLE 2
Summary of the Experiments With Convex Hull for RAP and RAPCOS

Method	Name of Seq	Loss_0	$\text{Loss}_{\text{final}}$	$\text{Time}_{\text{perm}}$
RAP	seq_1	106.2	0.102	1966
RAP	seq_2	128.3	0.102	4780
RAP	seq_3	108.6	0.104	1724
RAPCOS	seq_1	106.2	7.770	3996
RAPCOS	seq_2	128.3	6.506	2422
RAPCOS	seq_3	108.6	8.489	2182

1 Loss_0 *is the initial loss for the sequence*
2 $\text{Loss}_{\text{final}}$ *is the final loss when the convergence stops.*
3 $\text{Time}_{\text{perm}}$ *is the permutation count when the convergence stops.*

For the first sequence (seq_1), the initial loss is 106.21; the initial loss for the second sequence (seq_2) is 128.34; for the third sequence (seq_3), the initial loss is 108.61.

The comparison between RAP and RAPCOS, including the corresponding final losses and the numbers of permutation, are shown in Table 2.

For the first sequence (seq_1), the comparison of convergences between RAP and RAPCOS is shown in Fig. 4.

For the second/third sequence ($\text{seq}_2/\text{seq}_3$), the comparison between RAP and RAPCOS is supplemented.

The optimization using loss function Equation (9) is hard to reach exact zero, so we stop the convergence using the preset limit (10^{-1}). Then, the results of RAP are tested by inhull function to show whether they fall into the convex hull. Empirically, when the loss is about 0.1, the natural vector of searching sequence falls into the convex hull. The reason is that the hyperpyramid is inside the convex hull. As shown in Fig. 4 and corresponding figures in supplementary material, available online, RAP converges faster than RAPCOS. In fact, RAPCOS cannot converge in this task.

RAPCOS does not performs well for this task because there is no target natural vector to constrained the search region. The target vector we use in the task is the mean of some vertices (fuzzy target natural vector, Definition 2), which may not fall into the natural vector space. Therefore, the constrained search in Theorem 3 does not work.

6.3 Robustness of the Convergence of RAP

The convergence of randomized algorithms is essential for the implementation. Also, the rate of convergence largely affects the effectiveness of the algorithms. And we call an algorithm robust if the convergences are not associated with different initialization. In this section, we use the loss function measuring the distance from the natural vector of searching sequence to convex hull (Equation (9)). The preset limit for the convergence is set as 10^{-2} and the maximal iteration number is 10^4. We consider three aspects to show the effectiveness and robustness RAP. First, the convergence makes the natural vectors of final sequences fall into the convex hull. Second, for different initialization of the starting sequence, the convergences are robust instead of varying a lot with different initialization. Third, the rate of convergence is fast, i.e., the convergence happens in some limited number of permutation.

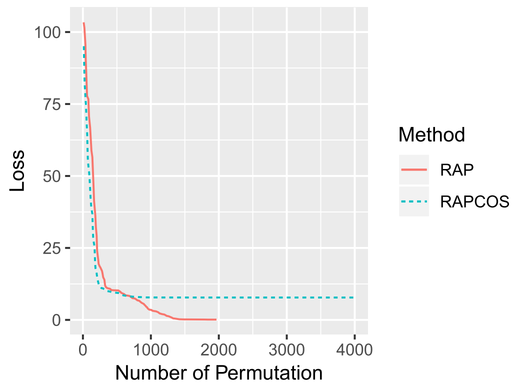

Fig. 4. The convergence of RAP and RAPCOS based on initial sequence (seq_1), loss function is based on the distance from natural vector to convex hull.

TABLE 3
Summary of the Experiments About the Convergence of RAP

Name of Seq	Loss_0	$\text{Loss}_{\text{final}}$	$\text{Time}_{\text{perm}}$
seq^1	9.55	0.100	2827
		0.106	7682
		0.099	6439
		0.105	8259
		0.102	5831
seq^2	11.63	0.098	5207
		0.102	1368
		0.099	3374
		0.108	3596
		0.103	1957
seq^3	8.36	0.1324	3931
		0.104	7533
		0.102	2244
		0.102	2244
		0.103	2895

1 Loss_0 *is the initial loss for the sequence*
2 $\text{Loss}_{\text{final}}$ *is the final loss when the convergence stops.*
3 $\text{Time}_{\text{perm}}$ *is the permutation count when the convergence stops.*

We initialize three sequences $\text{seq}^1, \text{seq}^2, \text{seq}^3$ to test the convergence. In each initialization, the random generation process is repeated for 1,000 times. The sequence with the smallest loss value is chosen to be the initialized sequence for optimization. For each sequence, experiments are repeated for five times. The details of convergence including the initial loss, final loss and the number of convergence are displayed in Table 3.

The initialization process makes the natural vectors of starting sequences close to the convex hull. For the first sequence (seq^1), the initial loss is 9.55; the initial loss for the second sequence (seq^2) is 11.63; for the third sequence (seq^3), the initial loss is 8.36.

The five results for each sequence are denoted as res1, res2, res3, res4, res5 in the following figures.

For the first sequence (seq^1), the convergences of five different initialization are shown in Fig. 5.

For the second sequence (seq^2) and the third sequence (seq^3), the convergences of five different initialization are shown in the supplementary material, available online.

As shown in Fig. 5 and corresponding figures in supplementary material, available online, for different initialization, RAP method converges fast. The convergence happens within about 10,000 times of permutation. And the convergences are

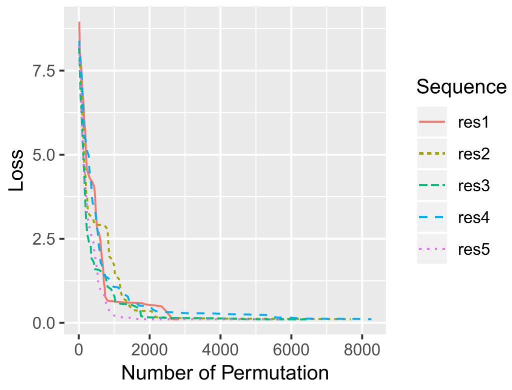

Fig. 5. The convergence of RAP with initial sequence (seq^1) and five repeated results, loss function is based on the distance from natural vector to convex hull.

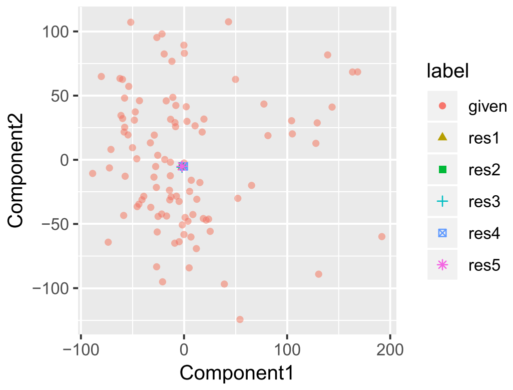

Fig. 6. Visualization based on PCA: The first two principal components (Initial sequence (seq^1) with 5 repeated convergence results).

not associated with different initialization. Thus, RAP approach is robust with respect to different initialization.

To better show the results of convergence, we use principal component analysis (PCA) to visualize the natural vectors of given sequences and the obtained ones. The first and second principal components are used to plot the figures.

For the first sequence (seq^1), the visualization of the convergence is shown in Fig. 6. The five points in Fig. 6 is too close to distinguish each other. One may refer to the supplementary material, available online, for the jittered figure based on Fig. 6.

For the second/third sequence ($\mathrm{seq}^2/\mathrm{seq}^3$), the visualization of the convergence is shown in the supplementary material, available online.

6.4 Computation Time

The computation time is measured using personal computer (PC) since the requirement of memory is small. The experiments are performed on 2.7 GHz Intel Core i5 and 8 GB 1867 MHz DDR3. The time for 1675 times of permutations is 59.89 seconds. In most experiments, the convergence ends within 2 minutes. Due to the high efficiency of permutation, the proposed algorithms are computationally efficient.

7 Conclusion & Discisson

Motivated to find existing undiscovered genome sequence mutations, predict potential genome mutations or provide supplement for metagenomic analysis, we propose the new genome sequence detection problem. We apply the properties of natural vector convex hull method to detect new, undiscovered genome sequences mathematically. Two novel randomized algorithms, RAP and RAPCOS, are proposed to solve the new genome sequence detection problem. In particular, RAPCOS is an improved version of RAP when using distance from natural vector to target natural vector as the loss function. We provide two different solutions to new genome sequence detection problem in Figs. 1 and 2. RAP and RAPCOS are randomized algorithms enjoying many great advantages. First, RAP is simple to implement and require small memory size. Second, the convergence of RAP is robust with respect to different initialization. Third, RAP converges fast. Fourth, for certain loss function, RAPCOS, which is an improved version of RAP, constrains the search region to speed up the convergence. Fifth, the new genome sequence detection problem is NP-hard but our RAP method is applicable to solve the problem.

Our work has some limitations and some challenges remain open to solve in the future. First, this work provides a framework to detect new genome sequence. But the experiments are only based on HIV genome sequences. The further experiments on other viruses, bacteria, eukaryota and more species be carried out under the same framework. For protein, there are twenty different amino acids, when applying the framework to detect new amino acid sequences, the difficulty are still unknown. Second, the choice of vertices in Algorithm 3 is ad hoc. It remains a challenge to design better strategies to assist the convergence of RAP and the detection of new genome sequences. Third, our designed algorithm can provide a genome sequence whose natural vector lies in the given convex hull. It still remains a challenge to detect all the genome sequences whose natural vectors are in the given convex hull. Fourth, we use natural vector with counts, mean positions and the second-order of central moments of each nucleotide. Higher order central moments can be used in further work. Fifth, the convergences of RAP and RAPCOS are analyzed empirically. Some theoretical results about the convergences are open to develop. Sixth, RAP is evaluated in mathematical way, it remains open to design a more biological way to perform evaluation.

Acknowledgments

The authors thank Dr. Changchuan Yin for the suggestions about revision. The authors thank the Department of Mathematical Science at Tsinghua University for library facilities. This work was supported by National Natural Science Foundation of China (NSFC) grant (91746119), Tsinghua University Spring Breeze Fund (2020Z99CFY044), Tsinghua University start-up fund, and Tsinghua University Education Foundation fund (042202008). Professor Stephen Shing-Toung Yau is grateful to the National Center for Theoretical Sciences (NCTS) for providing an excellent research environment while part of this research was done.

References

[1] G. Dey and T. Meyer, "Phylogenetic profiling for probing the modular architecture of the human genome," *Cell Syst.*, vol. 1, no. 2, pp. 106–115, 2015.
[2] M. Elloumi, "Comparison of strings belonging to the same family," *Inf. Sci.*, vol. 111, no. 1–4, pp. 49–63, 1998.

[3] M. R. Kantorovitz, G. E. Robinson, and S. Sinha, “A statistical method for alignment-free comparison of regulatory sequences,” *Bioinformatics*, vol. 23, no. 13, pp. i249–i255, 2007.
[4] R. J. G. B. Campello and E. R. Hruschka, “On comparing two sequences of numbers and its applications to clustering analysis,” *Inf. Sci.*, vol. 179, no. 8, pp. 1025–1039, 2009.
[5] I. S. Povolotskaya and F. A. Kondrashov, “Sequence space and the ongoing expansion of the protein universe,” *Nature*, vol. 465, no. 7300, 2010, Art. no. 922.
[6] S. Vinga and J. Almeida, “Alignment-free sequence comparison—A review,” *Bioinformatics*, vol. 19, no. 4, pp. 513–523, 2003.
[7] M. Deng, C. Yu, Q. Liang, R. L. He, and S. S.-T. Yau, “A novel method of characterizing genetic sequences: Genome space with biological distance and applications,” *PLoS One*, vol. 6, no. 3, 2011, Art. no. e17293.
[8] M. Pérez-Enciso, N. Forneris, G. de los Campos, and A. Legarra, “Evaluating sequence-based genomic prediction with an efficient new simulator,” *Genetics*, vol. 205, no. 2, pp. 939–953, 2017.
[9] S. F. Schaffner, C. Foo, S. Gabriel, D. Reich, M. J. Daly, and D. Altshuler, “Calibrating a coalescent simulation of human genome sequence variation,” *Genome Res.*, vol. 15, no. 11, pp. 1576–1583, 2005.
[10] G. K. Chen, P. Marjoram, and J. D. Wall, “Fast and flexible simulation of dna sequence data,” *Genome Res.*, vol. 19, no. 1, pp. 136–142, 2009.
[11] M. A. DePristo, D. M. Weinreich, and D. L. Hartl, “Missense meanderings in sequence space: A biophysical view of protein evolution,” *Nat. Rev. Genet.*, vol. 6, no. 9, pp. 678–687, 2005.
[12] J. M. Smith, “Natural selection and the concept of a protein space,” *Nature*, vol. 225, no. 5232, pp. 563–564, 1970.
[13] E. V. Koonin, Y. I. Wolf, and G. P. Karev, “The structure of the protein universe and genome evolution,” *Nature*, vol. 420, no. 6912, pp. 218–223, 2002.
[14] J. M. Cuevas, R. Geller, R. Garijo, J. López-Aldeguer, and R. Sanjuán, “Extremely high mutation rate of hiv-1 in vivo,” *PLoS Biol.*, vol. 13, no. 9, 2015, Art. no. e1002251.
[15] C. S. Riesenfeld, P. D. Schloss, and J. Handelsman, “Metagenomics: Genomic analysis of microbial communities,” *Annu. Rev. Genet.*, vol. 38, pp. 525–552, 2004.
[16] K. Tian, X. Zhao, and S. S.-T. Yau, “Convex hull analysis of evolutionary and phylogenetic relationships between biological groups,” *J. Theor. Biol.*, vol. 456, pp. 34–40, 2018.
[17] N. V. Dokholyan, B. Shakhnovich, and E. I. Shakhnovich, “Expanding protein universe and its origin from the biological big bang,” *Proc. Nat. Acad. Sci. USA*, vol. 99, no. 22, pp. 14 132–14 136, 2002.
[18] W. Sellers *et al.*, “Minimum convex hull mass estimations of complete mounted skeletons,” *Biol. Lett.*, vol. 8, no. 5, pp. 842–845, 2012.
[19] S. Nepomnyachiy, N. Ben-Tal, and R. Kolodny, “Global view of the protein universe,” *Proc. Nat. Acad. Sci. USA*, vol. 111, no. 32, pp. 11 691–11 696, 2014.
[20] X. Zhao, K. Tian, R. L. He, and S. S.-T. Yau, “Convex hull principle for classification and phylogeny of eukaryotic proteins,” *Genomics*, vol. 111, pp. 1777–1784, Dec. 2019.
[21] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, “33.3: Finding the convex hull,” *Introduction Algorithms*, pp. 955–956, 1990.
[22] D. Lischinski, “Incremental delaunay triangulation,” *Graph. Gems*, vol. 4, pp. 47–59, 1994.
[23] L. J. Guibas, D. E. Knuth, and M. Sharir, “Randomized incremental construction of delaunay and voronoi diagrams,” *Algorithmica*, vol. 7, no. 1–6, pp. 381–413, 1992.
[24] R. M. Karp, “Reducibility among combinatorial problems,” in *Complexity of Computer Computations*. Berlin, Germany: Springer, 1972, pp. 85–103.
[25] L. A. Wolsey and G. L. Nemhauser, *Integer and Combinatorial Optimization*. Hoboken, NJ, USA: Wiley, 2014.
[26] A. Schrijver, *Theory of Linear and Integer Programming*. Hoboken, NJ, USA: Wiley, 1998.
[27] H. H. Hoos and T. Stützle, *Stochastic Local Search: Foundations and Applications*. Amsterdam, The Netherlands: Elsevier, 2004.
[28] M. Herzog, G. Kaplan, and A. Lev, “Representation of permutations as products of two cycles,” *Discrete Math.*, vol. 285, no. 1–3, pp. 323–327, 2004.
[29] K. Pearson, “Liii. on lines and planes of closest fit to systems of points in space,” *London Edinburgh Dublin Philos. Mag. J. Sci.*, vol. 2, no. 11, pp. 559–572, 1901.

Ruzhang Zhao received the BS degree in mathematics from the Department of Mathematical Sciences, Tsinghua University, Beijing, China, in 2019. He is currently working toward the PhD degree in biostatistics from the Department of Biostatistics, Bloomberg School of Public Health, Johns Hopkins University, Baltimore, MD. His research interests include machine learning, computational genomics, statistical genetics, and high dimensional statistics.

Shaojun Pei received the BS degree from the School of Mathematical Sciences, Beijing Normal University, Beijing, China, in 2017. She is currently working toward the PhD degree in applied mathematics from the Department of Mathematical Sciences, Tsinghua University, Beijing, China. Her research interests include bioinformatics, biostatistics, and computational biology.

Stephen S. T. Yau (Fellow, IEEE) received the PhD degree in mathematics from the State University of New York at Stony Brook, NY, in 1976. He was a member of the Institute of Advanced Study at Princeton from 1976–1977 and 1981–1982, and a Benjamin Pierce assistant professor with Harvard University during 1977–1980. After that, he joined the Department of Mathematics, Statistics and Computer Science (MSCS), University of Illinois at Chicago (UIC), and served for more than 30 years. In 2005, he was named UIC distinguished professor. During 2005–2011, he was a joint professor with the Department of Electrical and Computer Engineering, MSCS, UIC. After his retirement in 2012, he joined Tsinghua University, Beijing, China, where he is a full-time professor with the Department of Mathematical Sciences. His research interests include bioinformatics, computational biology, nonlinear filtering, complex algebraic geometry, CR geometry, and sigularities theory. He is founder and managing editor since 1991 of the *Journal of Algebraic Geometry*. He is also founder and editor-in-chief since 2000 of *Communications in Information and Systems*. He is the coorganizer of Computational and Mathematical Approaches for Bioinformatics and Biophysics Workshop held in Sanya, China. He was awarded the Sloan Fellowship in 1980, the Guggenheim Fellowship in 2000, and the AMS Fellow Award in 2013.