The Effect of Genome Graph Expressiveness on the Discrepancy Between Genome Graph Distance and String Set Distance

Yutong Qiu[1] and Carl Kingsford[1*]

[1]Compuational Biology Department, Carnegie Mellon University,

Pittsburgh, PA, United States 15232

Abstract

Motivation: Intra-sample heterogeneity describes the phenomenon where a genomic sample contains a diverse set of genomic sequences. In practice, the true string sets in a sample are often unknown due to limitations in sequencing technology. In order to compare heterogeneous samples, genome graphs can be used to represent such sets of strings. However, a genome graph is generally able to represent a string set universe that contains multiple sets of strings in addition to the true string set. This difference between genome graphs and string sets is not well characterized. As a result, a distance metric between genome graphs may not match the distance between true string sets.

Results: We extend a genome graph distance metric, Graph Traversal Edit Distance (GTED) proposed by Ebrahimpour Boroojeny et al., to FGTED to model the distance between heterogeneous string sets and show that GTED and FGTED always underestimate the Earth Mover's Edit Distance (EMED) between string sets. We introduce the notion of string set universe diameter of a genome graph. Using the diameter, we are able to upper-bound the deviation of FGTED from EMED and to improve FGTED so that it reduces the average error in empirically estimating the similarity between true string sets. On simulated TCR sequences and Hepatitis B virus genomes, we show that the diameter-corrected FGTED reduces the average deviation of the estimated distance from the true string set distances by more than 250%.

Availability: Data and source code for reproducing the experiments are available at:
https://github.com/Kingsford-Group/gtedemedtest/

Contact: carlk@cs.cmu.edu

1 Introduction

Intra-sample heterogeneity describes the phenomenon where a genomic sample contains a diverse set of genomic sequences. A heterogeneous string set is a set of strings where each string is assigned a weight representing its abundance in the set. Computing the distance between heterogeneous string sets is essentially computing the distance between two distributions of strings. We formulate the problem of heterogeneous sample comparison as the heterogeneous string set comparison problem.

This problem can be used to compare samples where differences can be traced to the differences between sets of genomic sequences. For example, cancer samples are clustered based on differences in their genomic

*Corresponding Author. Email: carlk@cs.cmu.edu

and transcriptomic features [1, 2] into cancer subtypes that correlate with patient survival rates. The dissimilarities between T-cell receptor (TCR) sequences are computed between individuals to study immune responses [3]. Different compositions of these sequences result in different clinical outcomes such as response to treatment.

We point out that the Earth Mover's Distance (EMD) [4], or the Wasserstein distance [5], with edit distance as the ground metric is an elegant metric to compare a pair of heterogeneous string sets. Given two distributions of items and a cost to transform one item into another, EMD computes the total cost of transforming one distribution into another. The EMD was initially used in computer vision to compare distributions of pixel values in images [6] and later adapted to natural language processing [7]. It has also been used to approximate the distance between two genomes [8] by computing the distance between two distributions of k-mers. To compare heterogeneous string sets, when the strings and their distributions are known, we use edit distance as the cost to transform one string to another. We refer to this as the Earth Mover's Edit Distance (EMED).

In practice, the complete strings of interest and their abundances are often unknown, and these strings are only observed as fragmented sequencing reads. It is impossible to exactly compute EMED between the true sets of complete strings from the sequencing reads only.

The challenges posed by incomplete observed sequences can be alleviated by representing the string set using a graph structure. Multiple types of genome graphs have been introduced [9, 10, 11, 12, 13, 14, 15, 16, 17, 18]. For our purposes, a genome graph is a directed multigraph with labeled nodes and weighted edges, along with a source and a sink node. A string is spelled by a source-to-sink path, or $s-t$ path, if it is equal to the concatenation of node labels on the path. We say that a genome graph represents a string set if the union of paths that spells each string in the set is equal to the graph. In other words, a string set can be spelled by a decomposition of the genome graph.

There are several methods that compute the distance between genome graphs [19, 20, 21]. Among those, Graph Traversal Edit Distance (GTED) [21] is a general measure that can be applied to genome graphs and does not rely on the type of genome graphs nor the knowledge of the true string sets. Given two genome graphs, GTED finds an Eulerian cycle in each graph that minimizes the edit distance between the strings spelled by each cycle.

However, applying GTED on genome graphs representing heterogeneous string sets may overestimate the similarity between these string sets for two reasons. First, since GTED computes the distance between Eulerian cycles in genome graphs, it may align the prefix of a string to the suffix of another string with no additional penalties. We address this challenge by proposing an extension of GTED, called FGTED, that penalizes direct alignment of prefixes of a string with suffixes of other strings.

Second, and more significantly, both FGTED and GTED compute the edit distance between the two string sets represented by each genome graph that are most similar to each other. However, a genome graph that is constructed from sequencing fragments typically is able to represent more than one set of strings [22, 23]. As a genome graph merges shared sequences into the same node, it creates chains of bubble structures [24] that result in exponential number of possible paths, and these paths spell a much more diverse collection of strings than the original set. We call the degree to which a genome graph encodes a larger set of strings than the true underlying set the "expressiveness" of a genome graph. Due to the expressiveness of a genome graph, the Eulerian cycles found by GTED may not spell the true set of strings and the computed

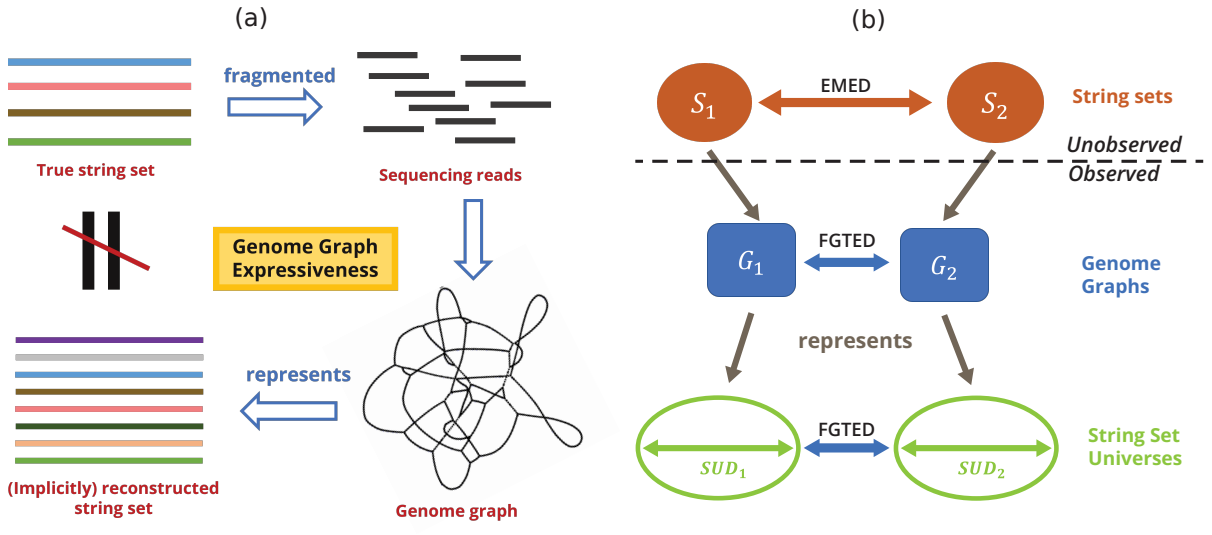

Figure 1: (a) Genome graph expressiveness results in inexact representations of true string sets. (b) Overview of part of theoretical contributions.

distance may be far from the true distance between string sets used to construct the graphs (Figure 1(a)).

We prove both that FGTED always produces a distance that is larger than or equal to GTED, and that FGTED computes a metric that is always less than or equal to the EMED between true sets of strings.

However, FGTED and GTED can be quite far from the EMED. To resolve this discrepancy between FGTED and EMED, we define the collection of strings that can be represented by the genome graph as its string set universe, and genome graph expressiveness as the diameter of its string set universe (SUD), which is the maximum EMED between two string sets that can be represented by the graph (Figure 1(b)).

Using diameters, we are able to upper-bound the deviation of FGTED from EMED. Additionally, we are able to correct FGTED and more accurately estimate the true string set distance empirically. On simulated TCR sequences, we reduce the average deviation of FGTED from EMED by more than 300%, and increase the correlation between the true and estimated string set distances by 20%. On Hepatitis B virus genomes, we reduce the average deviation by more than 250%.

These results provide the first connection between comparisons of genome graphs that encode multiple sequences and a natural string distance and provide the first formalization of the expressiveness of genome graphs. Additionally, they provide a practical method to estimate and reduce discrepancy between genome graph distances and string set distances.

2 Preliminary Concepts

2.1 Strings

Definition 1 (Heterogeneous string set)**.** *A heterogeneous string set* $\mathcal{S} = \{(w_1, s_1), \ldots, (w_n, s_n)\}$ *contains a set of strings, where each string* s_i *is assigned a weight* $w_i \in [0, 1]$ *that indicates the abundance of* s_i *in* $\mathcal{S}$*. We say that the total weight of* $\mathcal{S}$ *is* $\sum_{i \in [1,n]} w_i = 1$*.*

$ED(s_1, s_2)$ is the minimum cost to transform s_1 into s_2 under edit distance [25]. The set of operations that transforms s_1 to s_2 can be written as an alignment between s_1 and s_2, or $A = align(s_1, s_2)$. The i-th position in A is denoted by $A[i] = \begin{bmatrix} c_{(1,i)} \\ c_{(2,i)} \end{bmatrix}$, where $c_{(a,i)}$ is either a gap character "-" or a character in s_a.

2.2 Earth Mover's Edit Distance

To find a distance between two heterogeneous string sets, we need to take into account not only the distance between pairs of strings, but also the weight, or abundance of each string in the set. When we are comparing two heterogeneous string sets, we are essentially comparing two distributions of strings. Therefore, we propose using the Earth Mover's Distance (EMD) as a natural distance measure.

Given two distributions of items (here, strings) and a cost function that quantifies the cost of transforming one item into another, the EMD between the two distributions is the minimum cost to transform one distribution into another. Computing EMD can be viewed as a transportation problem that finds a many-to-many mapping between two sets of items and minimizes the total cost of the mapping [4, 5].

Given two heterogeneous string sets $\mathcal{S}_1 = \{(w_1, s_1), \ldots, (w_n, s_n)\}$ and $\mathcal{S}_2 = \{(w_{n+1}, s_{n+1}), \ldots, (w_m, s_m)\}$, to compute the Earth Mover's Edit Distance (EMED), we use the edit distance between s_i and s_j as the cost of transforming one string to another. Following procedures to compute EMD [4] as a min-cost max-flow problem, we find a mapping M, where $M(s_i, s_j)$ is the amount of $s_i \in \mathcal{S}_1$ to be transformed into $s_j \in \mathcal{S}_2$, that minimizes $cost(M) = \sum_{\substack{s_i \in \mathcal{S}_1 \\ s_j \in \mathcal{S}_2}} M(s_i, s_j) \cdot ED(s_i, s_j)$. We define that $EMED(\mathcal{S}_1, \mathcal{S}_2) = \min_M cost(M)$.

2.3 Flow Networks

Definition 2 (Valid flow network)**.** *A directed graph $\mathcal{G} = (V, E, w)$, where $w(e)$ is the weight of each edge, is a valid flow network if there exists a source s and sink node t such that:*

$$\textit{(Flow conservation)} \quad \sum_{(u,v)\in E} w(u, v) = \sum_{(v,w)\in E} w(v, w) \qquad \forall v \in V, v \neq s, v \neq t$$

$$\textit{(Total capacity)} \quad \sum_{(s,u)\in E} w(s, u) = \sum_{(v,t)\in E} w(v, t).$$

Definition 3 (Flow decomposition)**.** *A flow decomposition of a valid flow graph $\mathcal{G}$, denoted as $D(\mathcal{G})$, is a collection of paths and their weights $\mathcal{P} = \{(w_1, p_1), \ldots, (w_n, p_n)\}$, where $p_i = \big((s, u_1), \ldots, (u_m, t)\big)$ is an ordered sequence of edges in $\mathcal{G}$, such that:*

$$\textit{(Flow coverage)} \quad \sum_{p_i \in \mathcal{P}} O(e, i) \cdot w_i = w(e) \quad \forall e \in \mathcal{G},$$

where $O(e, i)$ is equal to the number of occurrences of edge e in path p_i.

A valid flow network typically has more than one flow decomposition. Let the set of possible flow decompositions of $\mathcal{G}$ be $\mathcal{D}_\mathcal{G}$.

2.4 Genome Graphs

There are many variants of genome graphs used for various purposes and in various settings. Here, we introduce the definition of genome graphs we will use.

Definition 4 (Genome graph)**.** *A genome graph $\mathcal{G} = (V, E, l, w)$ is a valid flow network with node set V, edge set E, node labels $l(u)$ for each $u \in V$ and edge weights $w(e)$ for each $e \in E$. A genome graph contains a source node s and a sink node t, and $l(s) =$ "\$", $l(t) =$ "#", where \$ and # are special characters that do not appear in any string set considered in the scope of this manuscript.*

Define operator $S(\cdot)$ that transforms a set of paths in a genome graph $\mathcal{G}$ to a set of strings by concatenating the node labels on each path. $S(\mathcal{P}) = \{\big(concat(p), w(p)\big) \mid p \in \mathcal{P}\}$ is a heterogeneous string set where the weight of each string is equal to the weight of the path that spells the string.

Definition 5 (String set represented by a genome graph)**.** *A genome graph $\mathcal{G}$ represents a string set $\mathcal{S}$ if there exists a decomposition $D(\mathcal{G}) \in \mathcal{D}_{\mathcal{G}}$, such that $S(D(\mathcal{G})) = \mathcal{S}$.*

We use $\mathcal{G} = G(\mathcal{S})$ to denote when $\mathcal{G}$ represents $\mathcal{S}$.

Definition 6 (String set universe represented by a genome graph)**.** *The string set universe $SU(\mathcal{G})$ of a genome graph $\mathcal{G}$ is the collection of heterogeneous string sets that can be represented by $\mathcal{G}$. Formally, $SU(\mathcal{G}) = \{S(D) \mid D \in \mathcal{D}_{\mathcal{G}}\}$.*

2.5 Alignment Graph

An alignment graph is used to align two genome graphs [21] and can be viewed as a graph product between two genome graphs. A special case of the alignment graph [26] is used to align a string to a graph where the string is represented as a graph with only one path. We assume that the genome graphs to be aligned are transformed so that the label of each node contains only one character.

Definition 7 (Alignment graph)**.** *Given genome graphs $\mathcal{G}_1 = (V_1, E_1, l_1, w_1)$ and $\mathcal{G}_2 = (V_2, E_2, l_2, w_2)$, an alignment graph $AG(\mathcal{G}_1, \mathcal{G}_2) = (V_A, E_A, cost, w)$ is a directed graph with node set V_A, edge set E_A, edge cost $cost(e)$ and edge weight $w(e)$ for each edge $e \in E_A$. The alignment graph is defined following the steps:*

- *V_A is constructed by adding pairings of nodes in V_1 and V_2; that is $V_A = \{(u_1, u_2) \mid u_1 \in V_1, u_2 \in V_2\}$.*
- *For each edge $(u_1, v_1) \in E_1$ and $(u_2, v_2) \in E_2$, create three types of edges:*
  1. *A match/mismatch edge $e = ((u_1, u_2), (v_1, v_2))$ with $w(e) = \min\{w_1(u_1, v_1), w_2(u_2, v_2)\}$.*
  2. *An insertion (in) edge $e = ((u_1, u_2), (u_1, v_2))$ with $w(e) = w_2(u_2, v_2)$.*
  3. *A deletion (del) edge $e = ((u_1, u_2), (v_1, u_2))$ with $w(e) = w_1(u_1, v_1)$.*

  The cost of an in/del edge and a mismatch edge is equal to a customized penalty. The cost of a match edge is equal to zero. A match/mismatch edge should be distinguished with an in/del edge if the corresponding edge in one of the input graphs is a self-loop.

Each edge $e = ((u_1, u_2), (v_1, v_2))$ in an alignment graph can be projected onto one edge in each of the input graphs. An edge in each of the input graphs can also be projected onto a set of edges in AG.

Definition 8 (Projection function)**.** *Define the projection function as $P_{(\mathcal{G},\mathcal{H})}(e) = E'$ that maps an edge e from graph $\mathcal{G}$ to a set of edges E' in graph $\mathcal{H}$. The projection function maps an edge in the alignment graph to the edges in the input graphs that are matched together by that edge. It also maps an edge in one of the input graphs to a set of edges in the alignment graph where it is matched with other edges in another input graph. Specifically:*
Projection from alignment graph to one of the input graphs is defined by

$$P_{(AG,\mathcal{G}_i)}((u_1,u_2),(v_1,v_2)) = \{(u_i,v_i)\},\ i \in \{1,2\}.$$

Projection from one of the input graphs to alignment graph is defined by

$$P_{(\mathcal{G}_1,AG)}((u_1,v_1)) = \{e \mid e = ((u_1,u_2),(v_1,v_2)) \in E_{AG}\},$$
$$P_{(\mathcal{G}_2,AG)}((u_2,v_2)) = \{e \mid e = ((u_1,u_2),(v_1,v_2)) \in E_{AG}\}.$$

Given a set of paths $\mathcal{P}$ in AG, we use $P_{(AG,\mathcal{G}_i)}(\mathcal{P})$ to denote the projection of $\mathcal{P}$ onto $\mathcal{G}_i$, where $P_{(AG,\mathcal{G}_i)}(\mathcal{P}) = \{\big(P_{(AG,\mathcal{G}_i)}(e) \mid e \in p\big) \mid p \in \mathcal{P}\}$.

For convenience, we define that $f_i(D(AG)) = S(P_{(AG,\mathcal{G}_i)}(D(AG)))$, which is the set of strings spelled by a path decomposition in AG that is projected onto $\mathcal{G}_i$.

2.6 Graph Traversal Edit Distance (GTED)

Graph Traversal Edit Distance (GTED), proposed by Ebrahimpour Boroojeny et al. [21], is a distance between two labeled graphs which are assumed to be Eulerian graphs. Given a genome graph in our definition, we add an edge directing from sink to source with weight equal the sum of edge weights that are directing from the source node in order to make an Eulerian graph.

Let the language of $\mathcal{G}$, $L(\mathcal{G})$, be the set of strings spelled by Eulerian cycles in $\mathcal{G}$. Formally, $L(\mathcal{G}) = \{S(c) \mid c \text{ is an Eulerian cycle in } \mathcal{G}\}$.

Definition 9 (Graph Traversal Edit Distance [21])**.** *Let $\mathcal{G}_1$ and $\mathcal{G}_2$ be two Eulerian graphs, where the weights on the edges are seen as the number of times an edge must be visited in each Eulerian cycle. Then,*

$$GTED(\mathcal{G}_1,\mathcal{G}_2) = \min_{\substack{s_1 \in L(\mathcal{G}_1)\\ s_2 \in L(\mathcal{G}_2)}} ED(s_1,s_2).$$

GTED finds one Eulerian cycle in each genome graph such that the edit distance between the strings spelled by the Eulerian cycles is minimized. GTED is computed by solving a linear programming (LP)

formulation (Equations (1)-(4)) on the alignment graph $AG(\mathcal{G}_1, \mathcal{G}_2)$. The LP formulation is as follows:

$$\min_{x \in \mathbb{R}^{|E_A|}} \sum_{e \in E_A} cost(e) \cdot x_e \tag{1}$$

$$s.t. \quad \sum_{j,l} x_{((i,j),(k,l))} = w_1(i,k) \qquad \forall (i,k) \in E_1 \tag{2}$$

$$\sum_{i,k} x_{((i,j),(k,l))} = w_2(j,l) \qquad \forall (j,l) \in E_2 \tag{3}$$

$$\sum_{(u,v) \in E_A} x_{(u,v)} = \sum_{(v,w) \in E_A} x((v,w)) \qquad \forall v \in V_A \tag{4}$$

Ebrahimpour Boroojeny et al. [21] prove that GTED is equal to the optimal solution of this LP formulation.

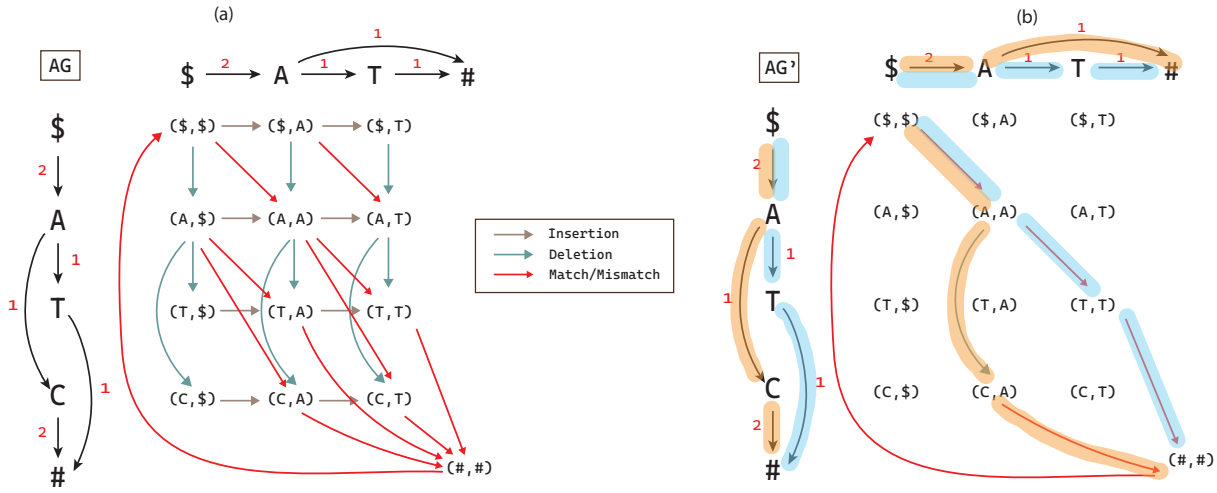

Figure 2: (a) An alignment graph AG between $\mathcal{G}_1$ (vertical) and $\mathcal{G}_2$ (horizontal). Insertion, deletion and match/mismatch edges are labeled with different colors. (b) AG^* after removing all the edges with zero flow in a solution to $FGTED(\mathcal{G}_1, \mathcal{G}_2)$. Edges in $\mathcal{G}_1$ and $\mathcal{G}_2$ that are highlighted with matching colors are projections from edges in AG^* to $\mathcal{G}_1$ and $\mathcal{G}_2$, respectively. Path $(\$, A, T, \#) \in \mathcal{G}_1$ is aligned to $(\$, A, T, \#) \in \mathcal{G}_2$ and path $(\$, A, C, \#) \in \mathcal{G}_1$ is aligned to $(\$, A, \#) \in \mathcal{G}_2$. The weights on AG and AG^* edges are omitted for simplicity.

3 An Extension of GTED

GTED was originally used to compare genome graphs that are assumed to contain single genomes. It is therefore intuitive that each string represented by the genome graph is spelled with an Eulerian cycle. This property follows the property of assembly graphs [27]. When the genome graph represents more than one string, finding a string spelled by an Eulerian cycle c in the graph is equivalent to finding a concatenation of a permutation of strings in a string set. When aligning two Eulerian cycles, c_1 and c_2, from input graphs, the boundaries between strings are ignored and the prefix of one string may be aligned to the suffix of another string with no cost. However, such alignment is not allowed when we align sets of strings using EMED.

We propose an extension of GTED with a modified cost function in edit distance computation so that the cost of aligning the sink character # with any other character is infinity.

Figure 2(a) shows an example of the alignment graph built from two input graphs using the proposed cost function. Let the sink nodes in $\mathcal{G}_1$ and $\mathcal{G}_2$ be t_1 and t_2, and the source nodes be s_1 and s_2, respectively. After removing all the alignment edges with infinite costs, there is an edge to the alignment node (t_1, t_2) in AG if and only if there exists an edge (u_1, t_1) in $\mathcal{G}_1$ and an edge (u_2, t_2) in $\mathcal{G}_2$. The only incoming edge to (s_1, s_2) is $((t_1, t_2), (s_1, s_2))$. We refer to the edge $((t_1, t_2), (s_1, s_2))$ as the sink-to-source edge, or $t - s$ edge in alignment graph in the rest of the manuscript.

We let Flow-GTED, or FGTED, denote the distance computed using the alignment graph after removing all infinity cost edges that forbid aligning the sink with any nodes other than the source node.

Theorem 1. *$GTED(\mathcal{G}_1, \mathcal{G}_2) \leq FGTED(\mathcal{G}_1, \mathcal{G}_2)$ for any pair of genome graphs $\mathcal{G}_1, \mathcal{G}_2$.*

Proof. Since FGTED is computed on a smaller alignment graph that contains fewer edges than that for computing GTED, FGTED explores a smaller solution space than GTED in solving the LP formulation. Therefore, any feasible solution to the LP formulation for $FGTED(\mathcal{G}_1, \mathcal{G}_2)$ is a feasible solution to the LP formulation for $GTED(\mathcal{G}_1, \mathcal{G}_2)$. Since $GTED(\mathcal{G}_1, \mathcal{G}_2)$ minimizes the objective, the theorem is true. □

4 The Relationship Between GTED, FGTED and EMED

Let AG^* be the alignment graph after removing the $t-s$ edge and all the edges from $\{e \mid x_e = 0, e \in E_A\}$ from the LP solution to Equations (1)–(4). We say that AG^* is a solution of FGTED. Due to constraints (2)–(4), AG^* is a valid flow network. Let $D(AG^*)$ be a flow decomposition in AG^*. Similar to the Eulerian cycles found during the GTED computation, each path in $D(AG^*)$ can be projected to a path in $\mathcal{G}_1$ and a path in $\mathcal{G}_2$.

Denote $\mathcal{S}_1^* = f_1(D(AG^*)) = S(P_{(AG^*, \mathcal{G}_1)}(D(AG^*)))$ as the set of strings spelled by the set of projected paths from a decomposition $D(AG^*)$ to $\mathcal{G}_1$. Similarly, $\mathcal{S}_2^* = f_2(D(AG^*)) = S(P_{(AG^*, \mathcal{G}_2)}(D(AG^*)))$. We show an example of path projections in Figure 2(b).

Observing that we can do flow decomposition in both the FGTED solution and input genome graphs, we will show in this section that FGTED can be bounded by EMED between decompositions in input genome graphs and in the alignment graph solutions.

Theorem 2. *Given two sets of strings $\mathcal{S}_1$ and $\mathcal{S}_2$, and genome graphs representing these string sets, $\mathcal{G}_1 = G(\mathcal{S}_1)$ and $\mathcal{G}_2 = G(\mathcal{S}_2)$,*

$$\begin{aligned} 0 &\leq EMED(\mathcal{S}_1, \mathcal{S}_2) - FGTED(\mathcal{G}_1, \mathcal{G}_2) \\ &\leq \min_{\substack{D(AG^*) \in \mathcal{D}_{AG^*} \\ \mathcal{S}_1^* = f_1(D(AG^*)) \\ \mathcal{S}_2^* = f_2(D(AG^*))}} \left(EMED(\mathcal{S}_1, \mathcal{S}_1^*) + EMED(\mathcal{S}_2, \mathcal{S}_2^*) \right) \end{aligned}$$

where AG^* is the solution obtained from $FGTED(\mathcal{G}_1, \mathcal{G}_2)$.

The proof of this theorem is completed in two parts. The first inequality is proven in Section 4.1 and the second is proven in Section 4.2. Since FGTED computes a distance that is larger than GTED between the

same pair of genome graphs (Theorem 1), Theorem 2 also shows that FGTED always estimates the distance between true string sets more accurately than GTED.

4.1 FGTED is Always Less Than or Equal to EMED

We show in this section that FGTED can be expressed in terms of EMED between string sets constructed from decomposing AG^*. In other words, similar to GTED, FGTED finds a decomposition in $\mathcal{G}_1$ and $\mathcal{G}_2$, respectively, that minimizes the EMED between them. Analogous to Definition 9, we have:

Theorem 3. *Given two genome graphs $\mathcal{G}_1$ and $\mathcal{G}_2$,*

$$FGTED(\mathcal{G}_1, \mathcal{G}_2) = \min_{\substack{D(\mathcal{G}_1)\in\mathcal{D}_{\mathcal{G}_2},\\ D(\mathcal{G}_2)\in\mathcal{D}_{\mathcal{G}_2}}} EMED(S(D(\mathcal{G}_1)), S(D(\mathcal{G}_2)))$$

To prove Theorem 3, we first explore the relationship between an $s-t$ path in AG^* and the strings spelled by the projections of this path onto $\mathcal{G}_1$ and $\mathcal{G}_2$.

Lemma 1. *Given an s-t path $p \in D(AG^*)$, let $s_1 = S(P_{(AG^*,\mathcal{G}_1)}(p))$ be the string spelled by projecting p onto $\mathcal{G}_1$, and $s_2 = S(P_{(AG^*,\mathcal{G}_2)}(p))$. Then for any $p \in D(AG^*)$,*

$$\sum_{e\in p} cost(e) = ED(s_1, s_2).$$

Proof. We prove in two directions.

($\geq$ direction) We construct $A = align(s_1, s_2)$ from p. For each $e = ((u_1, u_2), (v_1, v_2)) \in p$:
(1) if $u_1 = v_1$, add $c = \left[\begin{smallmatrix}-\\ l(v_2)\end{smallmatrix}\right]$ to A, (2) if $u_2 = v_2$, add $c = \left[\begin{smallmatrix}l(v_1)\\ -\end{smallmatrix}\right]$ to A, (3) else, add $c = \left[\begin{smallmatrix}l(v_1)\\ l(v_2)\end{smallmatrix}\right]$ to A.
By definition of an alignment graph, $cost(e) = cost(c)$ in for all e, and therefore $cost(A) = \sum_{c\in A} cost(c) = \sum_{e\in p} cost(e)$. Since edit distance minimizes the cost of edit operations, $cost(A) = cost(p) \geq ED(s_1, s_2)$.

($\leq$ direction) We construct p' from $A^* = align(s_1, s_2)$ such that $cost(A^*) = ED(s_1, s_2)$. The procedure is similar as above — for each pair of adjacent entries in A^*, add corresponding edge to p'. Then $cost(p') = cost(A^*) = ED(s_1, s_2)$.

Let $AG' = AG^* \setminus p \cup p'$. Both p and p' can be found in AG, and both p and p' can be constructed by the alignment of the same pair of strings. Therefore, AG' is also a valid flow network and a feasible solution to FGTED. Since AG^* is the optimal solution to $FGTED$, $cost(AG^*) \leq cost(AG')$, and

$$\begin{aligned}
&cost(AG^*) - cost(AG') \leq 0\\
&\Rightarrow w(p)\cdot(cost(p) - cost(p')) \leq 0\\
&\Rightarrow w(p)\cdot(cost(p) - ED(s_1, s_2)) \leq 0\\
&\Rightarrow cost(p) \leq ED(s_1, s_2).
\end{aligned}$$

□

We have shown that the cost of an $s-t$ path in AG^* is equal to the edit distance between its projections onto input graphs. Using this lemma, we can transform an optimal FGTED solution into an EMED solution.

Given an optimal FGTED solution, AG^*, let the set of possible flow decompositions of AG^* be $\mathcal{D}_{AG^*}$. Let $D(AG^*)$ be one of the flow decompositions that is a set of weighted $s-t$ paths. We can construct heterogeneous string sets $\mathcal{S}_1^*$ and $\mathcal{S}_2^*$ by projecting paths in $D(AG^*)$ to $\mathcal{G}_1$ and $\mathcal{G}_2$. Formally, $\mathcal{S}_1^* = \left\{S(P_{(AG^*,\mathcal{G}_1)}(p)) \mid p \in D(AG^*)\right\}$ and $\mathcal{S}_2^* = \left\{S(P_{(AG^*,\mathcal{G}_2)}(p)) \mid p \in D(AG^*)\right\}$.

Lemma 2. *Given $\mathcal{S}_1^*$ and $\mathcal{S}_2^*$ obtained from any decomposition $D(AG^*) \in \mathcal{D}_{AG^*}$,*

$$EMED(\mathcal{S}_1^*, \mathcal{S}_2^*) = cost(AG^*),$$

where $cost(AG^*)$ is sum of edge costs in the solution alignment graph to FGTED.

Proof. We prove in two directions.

($\leq$ direction) We construct a mapping M between strings in $\mathcal{S}_1^*$ and $\mathcal{S}_2^*$ from the decomposition $D(AG^*)$, where $M(s_i, s_j)$ is the portion of $s_i \in \mathcal{S}_1$ and $s_j \in \mathcal{S}_2$ that are aligned. For each $p \in D(AG^*)$, we obtain s_1 and s_2 as strings constructed from projections of p onto $\mathcal{G}_1$ and $\mathcal{G}_2$ and increment the weight of mapping $M(s_1, s_2)$ by $w(p)$. After iterating through all paths in $D(AG^*)$, the cost of M is

$$\begin{aligned} cost(M) &= \sum_{(s_1,s_2)\in M} M(s_1, s_2) \cdot ED(s_1, s_2) \\ &= \sum_{p \in D(AG^*)} w(p) \cdot cost(p) = cost(AG^*). \end{aligned}$$

M is also a feasible solution to the LP formulation of EMED. Since $EMED$ minimizes the cost of mapping between $\mathcal{S}_1^*$ and $\mathcal{S}_2^*$, $EMED(\mathcal{S}_1^*, \mathcal{S}_2^*) \leq cost(AG^*)$.

($\geq$ direction) We construct a valid flow network, AG' using an optimal solution to $EMED(\mathcal{S}_1^*, \mathcal{S}_2^*)$. For each pairing (s_i, s_j) for $s_i \in \mathcal{S}_1$ and $s_j \in \mathcal{S}_2$, we obtain its weight w and cost c from the EMED solution. Let $A = align(s_i, s_j)$ be an optimal alignment under edit distance, and $cost(A) = c$. We then add a path corresponding to A with weight w in AG'. This follows the same procedure in the proof of Lemma 1. After adding all paths, we obtain AG' with $cost(AG') = EMED(\mathcal{S}_1^*, \mathcal{S}_2^*)$. Since $cost(AG^*)$ is minimized by FGTED, $cost(AG') \geq cost(AG^*) \Rightarrow EMED(\mathcal{S}_1^*, \mathcal{S}_2^*) \geq cost(AG^*)$. □

Lemma 2 provides a transformation algorithm between optimal solutions to EMED and solutions to FGTED. Using Lemma 2, we can show that the EMED between $\mathcal{S}_1^*$ and $\mathcal{S}_2^*$ constructed from any decomposition in AG^* is equal to the decompositions of $\mathcal{G}_1$ and $\mathcal{G}_2$ that are closest in terms of EMED.

Lemma 3. *Given $\mathcal{S}_1^*$ and $\mathcal{S}_2^*$ obtained from any decomposition $D(AG^*) \in \mathcal{D}_{AG^*}$,*

$$EMED(\mathcal{S}_1^*, \mathcal{S}_2^*) = \min_{\substack{D(\mathcal{G}_1)\in\mathcal{D}_{\mathcal{G}_2}, \\ D(\mathcal{G}_2)\in\mathcal{D}_{\mathcal{G}_2}}} EMED(S(D(\mathcal{G}_1)), S(D(\mathcal{G}_2)))$$

Proof. In Lemma 2, $\mathcal{S}_1^*$ and $\mathcal{S}_2^*$ can be constructed from decomposing $\mathcal{G}_1$ and $\mathcal{G}_2$. Suppose for contradiction that there exists a decomposition that constructs string sets $\mathcal{S}_1'$ and $\mathcal{S}_2'$, such that $EMED(\mathcal{S}_1^*, \mathcal{S}_2^*) > EMED(\mathcal{S}_1', \mathcal{S}_2')$. Following the procedure in the proof of Lemma 2, we can construct a feasible solution to FGTED with cost equal to $EMED(\mathcal{S}_1', \mathcal{S}_2')$, which is less than $cost(AG^*) = EMED(\mathcal{S}_1^*, \mathcal{S}_2^*)$. This contradicts with the assumption that FGTED minimizes $cost(AG^*)$. □

Theorem 3 is therefore true because of Lemma 2 and 3. Using Theorem 3, we are able to prove the first inequality in Theorem 2 with Lemma 4.

Lemma 4. *Given heterogeneous string sets $\mathcal{S}_1$ and $\mathcal{S}_2$ and genome graphs representing these string sets, $\mathcal{G}_1 = G(\mathcal{S}_1)$ and $\mathcal{G}_2 = G(\mathcal{S}_2)$, $FGTED(\mathcal{G}_1, \mathcal{G}_2) \leq EMED(\mathcal{S}_1, \mathcal{S}_2)$.*

Proof. Given Theorem 3, FGTED finds flow decomposition in $\mathcal{D}_{\mathcal{G}_1}$ and $\mathcal{D}_{\mathcal{G}_2}$ that minimizes the EMED between them. Since $\mathcal{S}_1$ and $\mathcal{S}_2$ can be constructed from a flow decomposition in $\mathcal{D}_{\mathcal{G}_1}$ and $\mathcal{D}_{\mathcal{G}_2}$, respectively, this lemma is true. □

4.2 Genome Graph Expressiveness

A genome graph typically can represent more than one set of strings. We name the collection of string sets representable by a genome graph the *string set universe* of that genome graph, or $SU(\mathcal{G})$. Using Theorem 3, we can say that FGTED finds two sets of strings in the string set universe of $\mathcal{G}$ that are closest in the metric space of EMED. We define the expressiveness of a genome graph as the diameter of its string set universe, which is the maximum EMED between the string sets in $SU(\mathcal{G})$.

Definition 10 (String Set Universe Diameter (SUD))**.** *Given a genome graph $\mathcal{G}$,*

$$SUD(\mathcal{G}) = \max_{\mathcal{S}_a, \mathcal{S}_b \in SU(\mathcal{G})} EMED(\mathcal{S}_a, \mathcal{S}_b)$$

4.2.1 String Set Universe Diameter as an Upper Bound on Deviation of FGTED from EMED

The string set universe diameter gives one measure of the size of $SU(G)$, and it can also be used to characterize the deviation of GTED from EMED.

Recall that $\mathcal{S}_1^*$ and $\mathcal{S}_2^*$ are string sets obtained from a decomposition $D(AG^*)$, and that $EMED(\mathcal{S}_1^*, \mathcal{S}_2^*) = FGTED(G(\mathcal{S}_1), G(\mathcal{S}_2))$, where $\mathcal{S}_1$ and $\mathcal{S}_2$ are true string sets. From Theorem 2, we have that $EMED(\mathcal{S}_1, \mathcal{S}_2) \geq EMED(\mathcal{S}_1^*, \mathcal{S}_2^*)$. We can bound the deviation of $EMED(\mathcal{S}_1^*, \mathcal{S}_2^*)$ from $EMED(\mathcal{S}_1, \mathcal{S}_2)$ using triangle inequalities.

Lemma 5. *Given string sets $\mathcal{S}_1$ and $\mathcal{S}_2$ and genome graphs $\mathcal{G}_1 = G(\mathcal{S}_1)$ and $\mathcal{G}_2 = G(\mathcal{S}_2)$,*

$$EMED(\mathcal{S}_1, \mathcal{S}_2) - FGTED(\mathcal{G}_1, \mathcal{G}_2) \leq \min_{\substack{D(AG^*) \in \mathcal{D}_{AG^*} \\ \mathcal{S}_1^* = f_1(D(AG^*)) \\ \mathcal{S}_2^* = f_2(D(AG^*))}} \big(EMED(\mathcal{S}_1, \mathcal{S}_1^*) + EMED(\mathcal{S}_2, \mathcal{S}_2^*)\big), \tag{5}$$

where AG^* is the solution obtained from $FGTED(\mathcal{G}_1, \mathcal{G}_2)$.

Proof. Both edit distance and EMD are metrics [4, 25], which means that triangle inequality holds for EMED between strings. Therefore, for any string sets $\mathcal{S}_1^*$ and $\mathcal{S}_2^*$,

$$EMED(\mathcal{S}_1, \mathcal{S}_1^*) + EMED(\mathcal{S}_1^*, \mathcal{S}_2) \geq EMED(\mathcal{S}_1, \mathcal{S}_2)$$
$$EMED(\mathcal{S}_2, \mathcal{S}_2^*) + EMED(\mathcal{S}_1^*, \mathcal{S}_2^*) \geq EMED(\mathcal{S}_1^*, \mathcal{S}_2)$$

Combining two inequalities, we have

$$\begin{aligned} &EMED(\mathcal{S}_1^*, \mathcal{S}_2^*) = FGTED(\mathcal{G}_1, \mathcal{G}_2) \geq EMED(\mathcal{S}_1, \mathcal{S}_2) - (EMED(\mathcal{S}_1, \mathcal{S}_1^*) + EMED(\mathcal{S}_2, \mathcal{S}_2^*)) \\ \Rightarrow &EMED(\mathcal{S}_1, \mathcal{S}_2) - FGTED(\mathcal{G}_1, \mathcal{G}_2) \leq EMED(\mathcal{S}_1, \mathcal{S}_1^*) + EMED(\mathcal{S}_2, \mathcal{S}_2^*). \end{aligned} \tag{6}$$

The above inequality (6) holds for any string sets $\mathcal{S}_1^*$ and $\mathcal{S}_2^*$. To give a tight upper bound on the deviation, we take the minimum over all possible pairs of string sets constructed from decomposing AG^* that yields inequality (5). □

Lemma 5 proves the second inequality of Theorem 2 thus completing the proof for Theorem 2 with Lemma 4.

The upper-bound found in Lemma 5 can be used as a factor that evaluates the pair-wise expressiveness of two genome graphs. While a genome graph may represent a large universe of string sets, as long as the true string set is close to the "best" string set in the pair-wise comparison, the deviation of FGTED from EMED is small. We define this upper bound as the String Universe Co-Expansion Factor (SUCEF), which can be used to evaluate the discrepancy between FGTED and EMED.

Definition 11 (String Universe Co-Expansion Factor (SUCEF))**.**

$$SUCEF(\mathcal{S}_1, \mathcal{S}_2, \mathcal{G}_1, \mathcal{G}_2) = \min_{\substack{D(AG^*) \in \mathcal{D}_{AG^*} \\ \mathcal{S}_1^* = f_1(D(AG^*)) \\ \mathcal{S}_2^* = f_2(D(AG^*))}} \big(EMED(\mathcal{S}_1, \mathcal{S}_1^*) + EMED(\mathcal{S}_2, \mathcal{S}_2^*)\big),$$

where AG^* is the solution to $FGTED(\mathcal{G}_1, \mathcal{G}_2)$.

On the other hand, finding SUCEF not only requires knowledge of true string sets $\mathcal{S}_1$ and $\mathcal{S}_2$, but SUCEF is also a pair-dependent measure that needs to be calculated for every pair of string sets and corresponding genome graphs. In order to characterize the effect of the expressiveness of individual genome graphs, we derive another upper bound on the deviation of FGTED from EMED using the string set universe diameters.

The sum of string set universe diameters of two genome graphs is an upper bound on SUCEF of these graphs and any two sets of strings they represent.

Lemma 6. *Given two genome graphs $\mathcal{G}_1$ and $\mathcal{G}_2$ and two sets of strings $\mathcal{S}_1$ and $\mathcal{S}_2$ they represent,*

$$\begin{aligned} EMED(\mathcal{S}_1, \mathcal{S}_2) - FGTED(\mathcal{G}_1, \mathcal{G}_2) &\leq SUCEF(\mathcal{S}_1, \mathcal{S}_2, \mathcal{G}_1, \mathcal{G}_2) \\ &\leq SUD(\mathcal{G}_1) + SUD(\mathcal{G}_2). \end{aligned}$$

Proof. Both $\mathcal{S}_1$ and $\mathcal{S}_1^*$ are represented by $\mathcal{G}_1$ and belong to $SU(\mathcal{G}_1)$. Therefore, by definition of string set universe diameter, $EMED(\mathcal{S}_1, \mathcal{S}_1^*) \leq SUD(\mathcal{G}_1)$ as the diameter maximizes the distance between any pair of strings represented by the genome graph. The same holds for $EMED(\mathcal{S}_2, \mathcal{S}_2^*) \leq SUD(\mathcal{G}_2)$. □

Using Lemma 6, we can now bound the deviation of FGTED from EMED using the expressiveness of individual genome graphs. In the following sections, we show that we can empirically estimate the anticipated discrepancy between FGTED and EMED using string set universe diameters, and we can produce improved FGTED which reduces the anticipated deviation from EMED and better correlates with EMED.

5 Practically Correcting the Discrepancy between FGTED and EMED

5.1 Estimating String Set Universe Diameters

The string set universe diameter of a genome graph can be estimated by sampling flow decompositions of the graph. To sample a flow decomposition, we first sample one $s-t$ path. At each node u, we choose the neighbor v with the highest edge weight $w(u,v)$ with probability 0.5 and randomly choose a neighbor otherwise. After sampling a path, we send flow that is equal to the minimum edge weight on that path and produce the residual graph by subtracting the flow from edge weights on that $s-t$ path. We repeat this process on the residual graph until all edge weights are zero. This process assumes that the input genome graphs are acyclic to ensure all edge capacities (weights) are satisfied. If a genome graph is cyclic, string sets from $SU(\mathcal{G}_1)$ can be obtained by sampling Eulerian cycles in the genome graph. After sampling 50 pairs of flow decompositions, we construct string sets from sampled flow decompositions and calculate pairwise EMED. We then obtain the highest pairwise EMED and use it as the estimated diameter.

5.2 Correcting FGTED Using String Set Universe Diameters

Using sum of SUDs, we empirically estimate the deviation of FGTED from EMED with a linear regression model. We denote the deviation of FGTED from EMED by $deviation(\mathcal{S}_1, \mathcal{S}_2, \mathcal{G}_1, \mathcal{G}_2)$, which is computed as $|EMED(\mathcal{S}_1, \mathcal{S}_2) - FGTED(G(\mathcal{S}_1), G(\mathcal{S}_2))|$. The linear regression model, LR, has the following form

$$deviation(\mathcal{S}_1, \mathcal{S}_2, \mathcal{G}_1, \mathcal{G}_2) = a \cdot \big(SUD(\mathcal{G}_1) + SUD(\mathcal{G}_2)\big) + b = LR(SUD(\mathcal{G}_1) + SUD(\mathcal{G}_2)),$$

where a is the coefficient of the model and b is the intercept. The fitted model will minimize the mean squared error between predicted deviation and true deviation in the training set.

The corrected FGTED for each pair of graphs is calculated using the learned linear regression model as follows.

$$correctedFGTED(\mathcal{G}_1, \mathcal{G}_2) = FGTED(\mathcal{G}_1, \mathcal{G}_2) + LR(SUD(\mathcal{G}_1) + SUD(\mathcal{G}_2))$$

The deviation of corrected FGTED from EMED has the same form as the deviation of uncorrected FGTED from EMED.

5.3 Data

We evaluate the use of string set universe diameters on two sequence sets:

1. **Simulated T-Cell Receptor (TCR) Repertoire.** We simulate 50 sets of TCR sequences and assign weights to each sequence using reference gene sequences of V, D and J genes from Immunogenetics (IMGT) V-Quest sequence directory [28]. The number of sequences in each set varies from 2 to 5. We then generate 225 pairs of TCR string sets. Each TCR sequence is about 300 base pairs long. See the Appendix for detailed simulation process.

2. **Hepatitis B Virus (HBV) Genomes.** We collect 9 sets of HBV genomes from three hosts — humans, bats and ducks — from the NCBI virus database [29]. We build 36 pairs of HBV string sets. See the Appendix for detailed string set construction process.

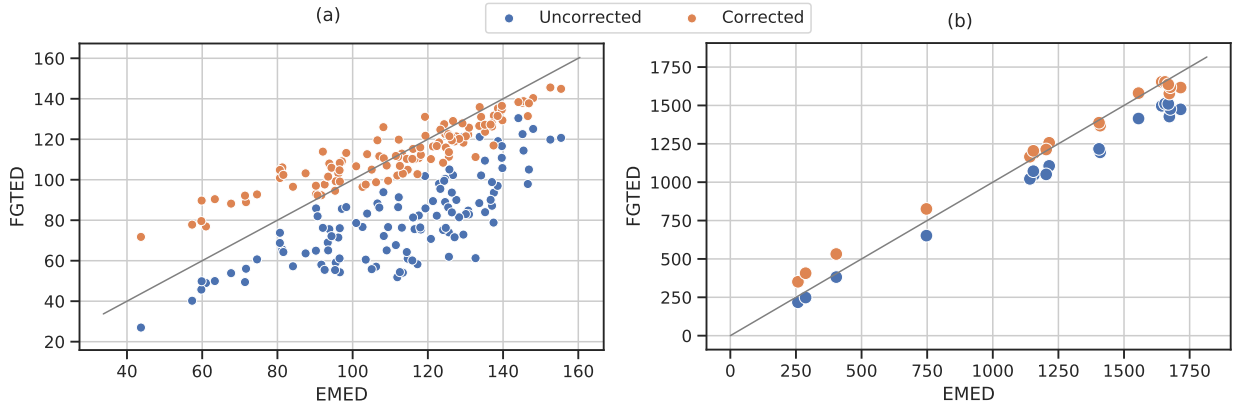

Figure 3: Correlation between EMED and FGTED before and after correction with string set universe diameters on half of (a) simulated TCR sequences and (b) HBV genomes. The relationship between EMED and uncorrected FGTED is shown in blue and corrected FGTED is shown in orange. Grey lines signify equality between EMED and FGTED. The data shown in Figure 3 are held-out sets from TCR and HBV genomes that consist of 50% of the data. The corrected FGTED is computed using the LR models fit on the other half.

We construct a partial order MSA graph on each string set [30]. We first conduct multiple sequence alignment (MSA) for each string set using Clustal Omega [31]. Then for each column of the MSA, we create a node for each unique character and add an edge between two nodes if the characters in node labels are adjacent in the input strings at that column. For each consecutive stretch of gap characters, no nodes are created, but an edge is added between flanking columns of the stretch of gaps. We also create a source node and a sink node that are connected to nodes representing the first and last characters of the input strings. The MSA graphs created in this process are all acyclic. We compute FGTED on MSA graphs by adding sink-to-source edges.

5.4 Corrected FGTED More Accurately Estimates Distance Between Unseen String Sets Encoded With Genome Graphs

We compute EMED and FGTED on string set pairs and genome graph pairs. Computing FGTED takes about 1 hour using 10 threads for each MSA graph of the simulated TCR sequence sets and 4 hours on using 10 threads for each MSA graph on the HBV genomes. The LP optimizations are done using the Gurobi solver [32].

For each pair of string sets, we obtain the deviation of FGTED from EMED and sum of estimated SUDs. We fit two linear regression models, LR_{TCR} and LR_{HBV}, to predict deviation from sum of SUDs on simulated TCR sequences and HBV genomes separately.

We evaluate the corrected and uncorrected FGTED by performing Pearson correlation experiments. We fit LR models on half of the data and compute the corrected FGTED on the other half as the test set. We evaluate the correlation between corrected and uncorrected FGTED and EMED on the test set. Two-tail P-values are calculated for each correlation experiment to test for non-correlation.

The LR models are evaluated with 10-fold cross validation. We randomly permute and split data into

10 equal parts. In each of the 10 iterations, we use one part as the test set and the rest as the training set. An average deviation is calculated across all iterations.

FGTED	Pearson Correlation		Average Deviation	
	Simulated TCR	HBV	Simulated TCR	HBV
Uncorrected	0.74	**0.99**	32.74	140.12
Corrected	**0.90**	**0.99**	**9.13**	**54.87**

Table 1: Pearson correlation and average deviation between EMED and corrected and uncorrected FGTED on simulated TCR and HBV sequences. Pearson correlation is calculated on a held-out set of data for both simulated TCR and HBV that consist of 50% of data, and LR model is fit on the other half. Average deviation is calculated by taking the average of mean deviations across 10-fold cross validation.

In Table 1, we show that using string set universe diameters, we are able to improve the correlation between FGTED and EMED on both the simulated TCR sequences and HBV genomes (Figure 3). Both Pearson correlation experiments are statistically significant with P-values < 0.01. On HBV genomes, since the correlation between uncorrected FGTED and EMED is approaching 1, no significant improvement is observed. On the other hand, significant reduction in average deviation is observed on both types of data. We are able to reduce the average deviation from 32.74 to 9.13 on simulated TCR sequences and from 140.12 to 54.87 on HBV genomes.

One caveat of using SUDs for correcting distances between genome graphs is that this correction is not guaranteed to always improve the distance. Given two string sets, there is always an adversarial worst case where adjusting the distance using this approach reduces the accuracy in estimating string sets distances. As seen from Figure 3, when EMED between true string sets are small, the corrected FGTED may overestimate the EMED and result in a larger deviation. Nevertheless, we show that corrected FGTED reduces the anticipated deviation from EMED.

6 Discussion

A genome graph's string set universe diameter (SUD) provides information on the size and diversity of the represented string sets. We show that we can use SUDs to practically characterize the discrepancy between GTED and EMED and to obtain a more accurate distance between unseen string sets encoded in genome graphs on average. While the results are obtained on short genomic sequences due to the high computational cost of FGTED and GTED, this result is encouraging, and string set universe diameters could be also used on approximate genome graph comparison methods to evaluate and improve their accuracies [19, 20].

SUDs could also be used to characterize the diversity of strings represented by reference genome graphs that are used in sequence-to-graph alignment [33, 34]. In sequence-to-graph alignment, it is often desired that a more diverse set of strings than the original reference string set is represented by the graph. Here, SUDs could be used as a measure to control the right amount of variation in the string set universe of created genome graphs.

Another future direction is to use expressiveness as a regularization term in the objective function to construct better genome graphs. To ensure efficiency of genome graphs in storing sequences, we can construct genome graphs that minimize their sizes [35, 36]. However, only optimizing for the size of a genome graph

may result in graphs that are highly expressive, and the distance between these genome graphs will deviate further from distances between true string sets. Adding a SUD term to the objective may address this problem.

7 Acknowledgements

This work was supported in part by the Gordon and Betty Moore Foundation's Data-Driven Discovery Initiative [GBMF4554 to C.K.], by the US National Institutes of Health [R01GM122935], the US National Science Foundation [DBI-1937540] and the Carnegie Mellon University School of Computer Science Sansom graduate fellowship for computational cancer research to Y.Q.

Conflict of Interest: C.K. is a co-founder of Ocean Genomics, Inc.

8 Appendix: Procedures to generate data sets

	Group 1	Group 2	Group 3	Group 4	Group 5
TRB_V	5	34	63	92	121
TRB_J	5	8	11	14	15
TRB_D	3	3	3	3	3

Table 2: The number of unique V, J and D gene sequences in each reference gene group.

8.1 Synthetic Sets of T-cell Receptor Sequences

We construct five reference gene groups by sampling reference sequences obtained from the IMGT database [28] that represent varied diversities of V, D, J gene repertoires. The number of sequences in each group is shown in Table 2. We construct five TCR sequence groups, and each group of TCR sequences are constructed using genes from one of the five reference gene groups. To generate each TCR sequence, we randomly select a V, D and J gene from corresponding gene group, and randomly introduce $m \in \{1, 3, 5, 8, 10\}$ single-nucleotide mutations to each sequence at random locations. This step is to simulate recombination and occurrences of junction single nucleotide polymorphisms (SNPs). 500 sequences are generated in each TCR sequence group.

We construct 50 immune repertoires in five groups. Each repertoire group are constructed using simulated TCR sequences from corresponding TCR sequence group. Within each group, each sequence set contains 2–10 sequences with randomly assigned weights that sum to 100. 45 string set pairs are generated within each group.

8.2 Heterogeneous sets of Hepatitis B Virus genomes

We downloaded 30 HBV genomes from each of three host species — human, bat, duck — from the NCBI virus database [29]. We construct 3 string sets for each host species. For each string set, we randomly select 5 HBV genomes from one host and randomly assign a weight to each string so that the sum of string weights in each set is equal to 100.

References

[1] Luc GT Morris, Nadeem Riaz, Alexis Desrichard, Yasin Şenbabaoğlu, A Ari Hakimi, Vladimir Makarov, Jorge S Reis-Filho, and Timothy A Chan. Pan-cancer analysis of intratumor heterogeneity as a prognostic determinant of survival. *Oncotarget*, 7(9):10051, 2016.

[2] Lan Zhao, Victor HF Lee, Michael K Ng, Hong Yan, and Maarten F Bijlsma. Molecular subtyping of cancer: current status and moving toward clinical applications. *Briefings in Bioinformatics*, 20(2): 572–584, 2019.

[3] Christopher R Bolen, Florian Rubelt, Jason A Vander Heiden, and Mark M Davis. The repertoire dissimilarity index as a method to compare lymphocyte receptor repertoires. *BMC Bioinformatics*, 18 (1):1–8, 2017.

[4] Yossi Rubner, Carlo Tomasi, and Leonidas J Guibas. The Earth Mover's distance as a metric for image retrieval. *International Journal of Computer Vision*, 40(2):99–121, 2000.

[5] Leonid N Wasserstein et al. Markov processes over denumerable products of spaces describing large systems of automata. *Problems of Information Transmission*, 5(3):47–52, 1969.

[6] Elizaveta Levina and Peter Bickel. The Earth Mover's distance is the mallows distance: Some insights from statistics. In *Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001*, volume 2, pages 251–256. IEEE, 2001.

[7] Matt Kusner, Yu Sun, Nicholas Kolkin, and Kilian Weinberger. From word embeddings to document distances. In *International Conference on Machine Learning*, pages 957–966. PMLR, 2015.

[8] Serghei Mangul and David Koslicki. Reference-free comparison of microbial communities via de Bruijn graphs. In *Proceedings of the 7th ACM International Conference on Bioinformatics, Computational Biology, and Health Informatics*, pages 68–77, 2016.

[9] Guillaume Holley and Páll Melsted. Bifrost: highly parallel construction and indexing of colored and compacted de Bruijn graphs. *Genome Biology*, 21(1):1–20, 2020.

[10] Fatemeh Almodaresi, Hirak Sarkar, Avi Srivastava, and Rob Patro. A space and time-efficient index for the compacted colored de Bruijn graph. *Bioinformatics*, 34(13):i169–i177, 2018.

[11] Heng Li, Xiaowen Feng, and Chong Chu. The design and construction of reference pangenome graphs with minigraph. *Genome Biology*, 21(1):1–19, 2020.

[12] Zamin Iqbal, Mario Caccamo, Isaac Turner, Paul Flicek, and Gil McVean. De novo assembly and genotyping of variants using colored de Bruijn graphs. *Nature Genetics*, 44(2):226–232, 2012.

[13] Alexander Dilthey, Charles Cox, Zamin Iqbal, Matthew R Nelson, and Gil McVean. Improved genome inference in the MHC using a population reference graph. *Nature Genetics*, 47(6):682–688, 2015.

[14] Erik Garrison, Jouni Sirén, Adam M Novak, Glenn Hickey, Jordan M Eizenga, Eric T Dawson, William Jones, Shilpa Garg, Charles Markello, Michael F Lin, et al. Variation graph toolkit improves read mapping by representing genetic variation in the reference. *Nature Biotechnology*, 36(9):875–879, 2018.

[15] Ilia Minkin, Son Pham, and Paul Medvedev. TwoPaCo: An efficient algorithm to build the compacted de Bruijn graph from many complete genomes. *Bioinformatics*, 33(24):4024–4032, 2017.

[16] Benedict Paten, Adam M Novak, Jordan M Eizenga, and Erik Garrison. Genome graphs and the evolution of genome inference. *Genome Research*, 27(5):665–676, 2017.

[17] Benedict Paten, Mark Diekhans, Dent Earl, John St John, Jian Ma, Bernard Suh, and David Haussler. Cactus graphs for genome comparisons. *Journal of Computational Biology*, 18(3):469–481, 2011.

[18] Heewook Lee and Carl Kingsford. Kourami: graph-guided assembly for novel human leukocyte antigen allele discovery. *Genome Biology*, 19(1):1–16, 2018.

[19] Ilia Minkin and Paul Medvedev. Scalable pairwise whole-genome homology mapping of long genomes with BubbZ. *IScience*, 23(6):101224, 2020.

[20] Evgeny Polevikov and Mikhail Kolmogorov. Synteny paths for assembly graphs comparison. In *19th International Workshop on Algorithms in Bioinformatics (WABI 2019)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2019.

[21] Ali Ebrahimpour Boroojeny, Akash Shrestha, Ali Sharifi-Zarchi, Suzanne Renick Gallagher, S Cenk Sahinalp, and Hamidreza Chitsaz. Graph traversal edit distance and extensions. *Journal of Computational Biology*, 27(3):317–329, 2020.

[22] Carl Kingsford, Michael C Schatz, and Mihai Pop. Assembly complexity of prokaryotic genomes using short reads. *BMC Bioinformatics*, 11(1):21, 2010.

[23] Benedict Paten, Jordan M Eizenga, Yohei M Rosen, Adam M Novak, Erik Garrison, and Glenn Hickey. Superbubbles, ultrabubbles, and cacti. *Journal of Computational Biology*, 25(7):649–663, 2018.

[24] Daniel R Zerbino and Ewan Birney. Velvet: algorithms for de novo short read assembly using de bruijn graphs. *Genome Research*, 18(5):821–829, 2008.

[25] Vladimir I Levenshtein et al. Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet Physics Doklady*, volume 10, pages 707–710. Soviet Union, 1966.

[26] Chirag Jain, Haowen Zhang, Yu Gao, and Srinivas Aluru. On the complexity of sequence-to-graph alignment. *Journal of Computational Biology*, 27(4):640–654, 2020.

[27] Pavel A Pevzner, Haixu Tang, and Michael S Waterman. An Eulerian path approach to DNA fragment assembly. *Proceedings of the National Academy of Sciences of USA*, 98(17):9748–9753, 2001.

[28] Marie-Paule Lefranc and Gérard Lefranc. *The immunoglobulin factsbook*. Academic press, 2001.

[29] Eneida L Hatcher, Sergey A Zhdanov, Yiming Bao, Olga Blinkova, Eric P Nawrocki, Yuri Ostapchuck, Alejandro A Schäffer, and J Rodney Brister. Virus variation resource–improved response to emergent viral outbreaks. *Nucleic Acids Research*, 45(D1):D482–D490, 2017.

[30] Christopher Lee, Catherine Grasso, and Mark F Sharlow. Multiple sequence alignment using partial order graphs. *Bioinformatics*, 18(3):452–464, 2002.

[31] Fabian Sievers, Andreas Wilm, David Dineen, Toby J Gibson, Kevin Karplus, Weizhong Li, Rodrigo Lopez, Hamish McWilliam, Michael Remmert, Johannes Söding, et al. Fast, scalable generation of high-quality protein multiple sequence alignments using clustal omega. *Molecular Systems Biology*, 7 (1):539, 2011.

[32] Gurobi Optimization, LLC. Gurobi Optimizer Reference Manual, 2021. URL `https://www.gurobi.com`.

[33] Mikko Rautiainen and Tobias Marschall. GraphAligner: rapid and versatile sequence-to-graph alignment. *Genome Biology*, 21(1):1–28, 2020.

[34] Jouni Sirén, Erik Garrison, Adam M Novak, Benedict Paten, and Richard Durbin. Haplotype-aware graph indexes. *Bioinformatics*, 36(2):400–407, 2020.

[35] Yutong Qiu and Carl Kingsford. Constructing small genome graphs via string compression. *Bioinformatics*, 37(Supplement 1):i205–i213, 2021.

[36] Prashant Pandey, Yinjie Gao, and Carl Kingsford. VariantStore: an index for large-scale genomic variant search. *Genome Biology*, 22(1):1–25, 2021.