HONORS THESIS

Numerical Encoding of Symbolic Data: Standard, State of the Art, and New Techniques

Perrin Ruth

Department of Applied Mathematics

University of Colorado - Boulder

Defended on Wednesday, March 24 2021

Committee Members

Thesis Advisor: Manuel E. Lladser, Department of Applied Mathematics (APPM)

Anne Dougherty, College of Arts and Sciences Honors Program & APPM

Youjian Liu, Department of Electrical, Computer, and Energy Engineering (ECEE)

Juan G. Restrepo, Department of Applied Mathematics (APPM)

Abstract

With an increasing prevalence of symbolic data comes a growing need for tools to study it. Nevertheless, most data analysis techniques have been developed for numerical datasets, and the natural numerical representation methods of symbolic data are highly dimensional. This motivates the problem of representing symbolic data as low-dimensional numerical vectors.

An ordered subset of nodes in a graph is called resolving when each vertex is uniquely determined by its distances to the set nodes. Multilateration is the process of representing any node by its distances to the elements of a resolving set. The metric dimension of a graph is the cardinality of the smallest resolving set; hence, the smallest dimension by which nodes in a graph may be represented as numerical vectors via multilateration.

Multilateration of Hamming graphs has shown to be a promising method for representing symbolic data numerically as low-dimensional vectors; however, this approach restricts the data to be words of the same length. To overcome this limitation, we commit to a novel notion of Levenshtein graphs: the vertices of these graphs are possibly words of varying size, and two vertices are connected by an edge only when their edits distance is one.

In this thesis, we uncover various properties of Levenshtein graphs. We find necessary and sufficient conditions for when the geodesic distance of these graphs is equivalent to the edit distance. Then, utilizing strings with at most two runs, we construct bounds on the metric dimension of Levenshtein graphs and a resolving set. We also characterize the automorphism group and determining sets of Levenshtein graphs. Finally, as a proof of concept, we demonstrate how these tools may be applied to represent DNA sequences numerically.

Table of Contents

1 Introduction

Suppose you were studying a robot moving on a network and you wanted to track its position. If you were to track a point on the plane you could use trilateration, where any such point can be uniquely identified by its distances to three noncollinear points. Similarly, we could place "landmark" devices on our network so that the robot could locate itself through its distances to these devices. It turns out that this problem is guaranteed to have a solution, but the question remains: *where are the best locations to place these landmarks?* This question has served as a classic inspiration for the theoretical study of the so-called resolving sets [16].

For a general unweighted graph $G = (V, E)$, a set $R \subset V$ is called resolving when for every pair of distinct nodes $u, v \in V$ there exists $r \in R$ such that $d(u, r) \neq d(v, r)$, where $d(\cdot, \cdot)$ denotes the geodesic distance between pairs of vertices in G. Note, we write $u \sim v$ if u and v are adjacent in G. The metric dimension of a graph, $\beta(G)$, is defined as the size of a smallest possible resolving set [13, 25]. See Figure 1.

The problem of finding the metric dimension of an arbitrary graph is NP-Complete [8, 12, 16]. Nevertheless, it has been characterized for trees, paths, and complete graphs [7], among many other graph families [28]. It is possible to estimate the metric dimension of a graph using the ICH algorithm [14], which runs in $O(|V|^3)$ time with an approximation ratio of $1 + (1 + o(1)) \ln |V|$. Therefore, in cases when the distance matrix is easily computable, e.g. on the order of thousands of nodes, the ICH algorithm serves as an effective method for generating resolving sets.

In general, resolving sets are useful to embed graphs into Euclidean spaces, and the smaller the cardinality of such a set, the smaller the dimension of the embedding. Indeed, if $R = \{r_1, \ldots, r_n\}$ of cardinality n resolves G then the transformation $\Phi_R : V \to \mathbb{R}^n$ defined as

$$\Phi_R(v) := \big(d(v, r_1), \ldots, d(v, r_n)\big) \tag{1}$$

represents nodes in G as n-dimensional vectors, see Figure 1. The smallest possible dimension in which to embed a graph in this way corresponds to its metric dimension. Resolving set based embeddings have various advantages over other state-of-the-art methods: they are one-to-one, require comparatively low complexity to be computed and, due to the triangular inequality, map nearby nodes in G to tuples with similar numerical coordinates [29].

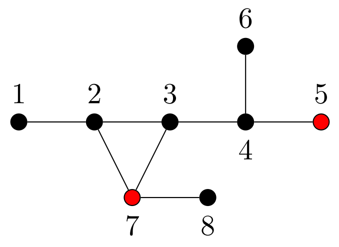


Figure 1: Visualization of a graph with a metric dimension of 2, because no single node multilaterates it (i.e., given any node, there are at least two nodes at the same geodesic distance from it). The set of nodes $R = \{5, 7\}$ (colored red) forms a minimal resolving set because any node can be represented uniquely by its geodesic distances to R. In fact, defining $\Phi_R(v) := \big(d(v, 5), d(v, 7)\big)$, we find that $\Phi_R(1) = (4, 2)$, $\Phi_R(2) = (3, 1)$, $\Phi_R(3) = (2, 1)$, $\Phi_R(4) = (1, 2)$, $\Phi_R(5) = (0, 3)$, $\Phi_R(6) = (2, 3)$, $\Phi_R(7) = (3, 0)$, $\Phi_R(8) = (4, 1)$.

The Cartesian product of two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$, denoted as $G_1 \,\square\, G_2$, has vertex set $V = V_1 \times V_2$, and two nodes $(u_1, u_2), (v_1, v_2) \in V$ are neighbors if either $u_1 \sim v_1$ in G_1 and $u_2 = v_2$, or $u_2 \sim v_2$ in G_2 and $u_1 = v_1$.

Hamming graphs have been of particular interest for the study of metric dimension. The Hamming graph $\mathbb{H}_{k;a}$ may be defined in terms of Cartesian powers of the complete graph, i.e.:

$$\mathbb{H}_{k;a} = K_a^{\square k} = \underbrace{K_a \,\square\, \cdots \,\square\, K_a}_{\text{k terms}}.$$

Nodes of $\mathbb{H}_{k;a}$ can be described by strings of a fixed length k over an alphabet of size a, where the geodesic distance is the Hamming distance (See Sec. 2). For simplicity, unless otherwise specified, we always assume our alphabet is of the form $\{0, 1, \ldots, a-1\}$.

Hamming graphs, while hosting a simple definition, appear in a number of applications. For a binary alphabet, $a = 2$, the metric dimension problem on $\mathbb{H}_{k;a}$ is equivalent to the coin weighing problem first introduced in [26]. Resolving sets of $\mathbb{H}_{k;a}$ can be used as solutions for the game of Mastermind as well [9]. Resolving sets of $\mathbb{H}_{k;a}$ allow strings of fixed length to be represented as numerical vectors. Furthermore, many machine learning classification algorithms are restricted to numerical data and cannot accept symbolic data like text or DNA. For this reason, resolving sets of Hamming graphs are useful for applying machine learning classification problems to symbolic data, specifically genetic data, of fixed length [29]. For more on the applications of Hamming graphs and resolving sets in general, we refer the reader to [9, 28].

Recall, an automorphism of a graph $G = (V, E)$ is a bijection of its nodes onto themselves, $\sigma : V \to V$, that preserves edge structure, namely, $u \sim v$ if and only if $\sigma(u) \sim \sigma(v)$. The set of all possible automorphisms of a graph $G = (V, E)$, called the automorphism group of G, is denoted $\mathbb{A}(G)$.

One key property of Hamming graphs is that they have many symmetries or automorphisms. These automorphisms are related to the convenient distance metric for nodes of Hamming graphs, and they stem from properties of Cartesian products of graphs [6]. Additionally, automorphisms preserve distances between nodes, so if the set R is resolving then so is $\sigma(R)$. Interestingly, the notions of automorphisms and resolving sets may be related further.

Determining sets were introduced independently by Boutin [3] and by Erwin and Harary as fixing sets [11]. Similar to how nodes can be identified by their distances to nodes of a resolving set, an automorphism can be determined by its action on a determining set. This provides a method of studying the automorphism group of a graph while only having to account for a potentially small set of nodes. Specifically, a set of nodes S is said to be determining if for each distinct pair of automorphisms $\sigma_1, \sigma_2 \in \mathbb{A}(G)$ there exists some $v \in S$ such that $\sigma_1(v) \neq \sigma_2(v)$. The determining number of a graph, $\text{Det}(G)$, is the size of a smallest determining set.

It happens that every resolving set is also a determining set, i.e. $\text{Det}(G) \leq \beta(G)$ [3, 11]. However, the difference between the determining number and the metric dimension of a graph can be quite large. In fact, $\beta(G) - \text{Det}(G)$ can grow on the order of $\Omega\left(\frac{2}{5}|V|\right)$ [5]. The determining number has also been studied for a few families of graphs including trees [5, 11] and Cartesian products of graphs [4, 5].

Thesis organization. In Section 2, we discuss some of the prominent methods for constructing resolving sets for Hamming graphs. After this we are motivated by the problem of representing biological strings of varying length as numerical vectors. In Section 3, we expand on the definition of Hamming graphs to construct what we call, Levenshtein graphs. The nodes of Levenshtein graphs are strings of varying length and, except for pathological cases, their geodesic distance is the well-known edit distance. In Section 4, we show a formula to describe the Levenshtein distance of an arbitrary string to a string with at most two runs. The tools in Section 4 are then used to prove results in Sections 5 and 6. In Section 5, we provide bounds on the metric dimension of Levenshtein graphs, including a constructive resolving set. In Section 6, we characterize the automorphism group of Levenshtein graphs. In Section 7, we discuss the determining number of Levenshtein graphs. Finally, in Section 8, we then provide an example of embedding genetic data using the techniques discussed in Section 5.

2 Hamming Graphs

The Hamming distance, $h(u, v)$, between two strings u and v of the same length is the total number of mismatches between them. For example, $h(abc, acc) = 1$. Up to a graph isomorphism, the Hamming graph $\mathbb{H}_{k;a}$, with $k \geq 1$ and $a \geq 2$ integers, has vertex set $\{0, \ldots, a-1\}^k$, and two vertices u and v are neighbors if and only if $h(u, v) = 1$; in particular, the geodesic distance between nodes in $\mathbb{H}_{k;a}$ is precisely their Hamming distance. In this section, we discuss current methods for constructing resolving sets for Hamming graphs and for reducing their size.

2.1 Constructing resolving sets for Hamming graphs

Recent work has shown that [15]:

$$\beta(\mathbb{H}_{k,a}) \sim \frac{2k}{\log_a(k)}, \text{ as } k \to \infty.$$

Since the proof of this result is constructive, for large k, one should be able to expose a resolving set for $\mathbb{H}_{k,a}$ of approximate size $2k/\log_a(k)$. Note the diameter of $\mathbb{H}_{k;a}$ is k, so an embedding using a resolving set R would map this Hamming graph into the set $\{0, 1, \ldots, k\}^{|R|}$. The number of bits required to represent a node using this way is approximately

$$\frac{2k}{\log_a(k)} \lceil \log_2(k+1) \rceil \approx 2k \log_2(a).$$

Therefore, one only needs about twice as many computer bits to store one of the a^k strings than a minimal mapping to binary digits.

For an intermediate string length k, a resolving set may be found recursively using an algorithm in [29]: starting from a resolving set of certain size s in $\mathbb{H}_{k-r,a}$, a resolving set of size $s + r\lfloor a/2 \rfloor$ may be determined for $\mathbb{H}_{k,a}$ in $O(ar^2)$ time. As seen in Figure 2, what distinguishes a large from an intermediate size k depends on the alphabet size a. This can be used in tandem with the ICH algorithm, which can be used to effectively produce a resolving set when k is relatively small.

2.2 Verifying resolving sets of Hamming graphs

Recently, a necessary and sufficient condition for resolvability in $H_{k,a}$ has been given in terms of one-hot-encodings in [17], which may be useful to identify unnecessary nodes in a resolving set and reduce its size. Applied to a resolving set of size 82 of $\mathbb{H}_{8;20}$ from the recursive method described in [29], it was possible to remove 5 nodes creating a resolving set of size 77 [17].

We note that not all of the nodes obtained using the asymptotic method in [15] are necessary either. For instance, the following list of size 26 was generated using this method to generate a resolving set for $\mathbb{H}_{10,5}$;

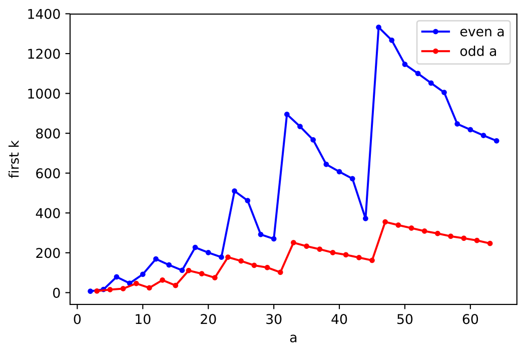


Figure 2: Plot of the smallest k such that a resolving set for $\mathbb{H}_{k,a}$, obtained using the constructive proof in [15], is smaller than the one obtained using the algorithm in [29] with $s = (a-1)$ and $r = (k-1)$.

however, the string 2024004423 (displayed in italics) is produced twice by the method. In particular, $\mathbb{H}_{10,5}$ may be resolved using only 25 nodes.

$$\begin{bmatrix} 0040144441, & 0304034414, & 0324113040, & 0324144443, & 0434300304, & 0434334414, \\ 1110401044, & 1110404414, & 1443101044, & 1443104410, & \mathit{2024004423}, & \mathit{2024004423}, \\ 2024040100, & 2024043340, & 2301004421, & 2344104421, & 3134100024, & 3134104424, \\ 3400404424, & 4003010424, & 4003044011, & 4320010420, & 4320044013, & 4341131424, \\ 4414104134, & 4414131424 & & & & \end{bmatrix}.$$

3 Levenshtein Graphs

Hamming graphs may be limiting because of their restriction on equal length strings, and because the Hamming distance is usually not relevant in biological settings. To overcome this limitation we may consider the Levenshtein distance [18], also called edit distance.

The Levenshtein distance $\ell(u, v)$ between two strings u and v of possibly different lengths is the minimal number of character substitutions, insertions, or deletions required to transform one string into the other. Since the Hamming distance can be thought of as the minimal number of substitutions to transform one string into the other, it follows for strings u and v of the same length, i.e. when $|u| = |v|$, that $\ell(u, v) \leq h(u, v)$.

The Levenshtein distance can also be described as the least possible score (i.e. total number of mismatches, or insertions and deletions aka indels) of an alignment between strings [10]. Indels are denoted with the reserved character $-$. To fix ideas, equations 2-4 display three alignments between the strings 001 and 01. The score of the alignment A in Equation 2 is two because the second 0 in the first row is mismatched with the character 1 in the second row, and the 1 in the first row is aligned against an indel. Similarly, the scores of alignments B and C are one. Since the score of any alignment between different strings must be one or larger, B and C are optimal alignments and $\ell(001,01)=1$.

$$A := \begin{array}{ccc} \mathtt{0} & \mathtt{0} & \mathtt{1} \\ \mathtt{0} & \mathtt{1} & - \end{array} \tag{2}$$

$$B := \begin{array}{ccc} \mathtt{0} & \mathtt{0} & \mathtt{1} \\ \mathtt{0} & - & \mathtt{1} \end{array} \tag{3}$$

$$C := \begin{array}{ccc} \mathtt{0} & \mathtt{0} & \mathtt{1} \\ - & \mathtt{0} & \mathtt{1} \end{array} \tag{4}$$

Optimal alignments can be determined and scored through a well-known dynamic programming approach, which has been invented many times in different contexts [18, 21, 32]. For strings $A=\alpha_1...\alpha_m$ and $B=\beta_1...\beta_n$ of lengths m and n, respectively, where α_i and β_j denote alphabet characters, this algorithm computes the columns (or rows) of the matrix with entries $d_{i,j}:=\ell(\alpha_1...\alpha_i,\beta_1...\beta_j)$ of dimensions $(m\times n)$ via the recursion:

$$d_{i,j}=\min\Big\{d_{i-1,j-1}+[\![\alpha_i\neq\beta_j]\!]\;, d_{i-1,j}+1, d_{i,j-1}+1\Big\},$$

where $[\![\cdot]\!]$ is the indicator function of the proposition within. The time complexity of this algorithm is $O(mn)$, which is expensive for long pairs of strings. However, by focusing on the diagonals of the matrix $(d_{i,j})$, as oppose to its columns or rows, it is possible to speed up the calculations to an $O\big(\ell(A,B)\cdot\min\{m,n\}\big)$ time complexity [30].

In Section 3.1, we use the edit distance to define a new notion of Levenshtein graphs similar to that of Hamming graphs, and we compare it to other notions of Levenshtein graphs in the literature. In Section 3.2 we show Levenshtein graphs are connected and provide necessary and sufficient conditions for the geodesic distance of two nodes to be equal to their edit distance.

3.1 Definition of Levenshtein graphs

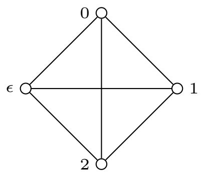
In this manuscript, we adopt the following notion of Levenstein graph.

Definition 3.1. *For integers $0\leq k_1\leq k_2$ and $a\geq 2$, the Levenshtein graph $\mathbb{L}_{k_1,k_2;a}$ has as vertices all strings of a length between k_1 and k_2 (inclusive) formed using the characters in $\{0,\dots,a-1\}$, and two nodes u and v are connected by an edge only when $\ell(u,v)=1$. We denote the vertex and edge set of this graph as $V_{k_1,k_2;a}$ and $E_{k_1,k_2;a}$, respectively. (See figures 3 and 4.)*

This definition could be extended to include $a=1$, in which case the associated graph would be a path. We impose the constraint that $a\geq 2$ for technical reasons, and because paths are well studied graphs [7].

In what follows, we write $\mathbb{L}_{k;a}$ as shorthand for $\mathbb{L}_{0,k;a}$. Accordingly, we denote the vertex and edge set of $\mathbb{L}_{k;a}$ as $V_{k;a}$ and $E_{k;a}$, respectively. The empty string, denoted from now on as ϵ, is the only vertex of length zero in this graph. We define $\mathbb{L}_a$ as the graph with vertex set $\cup_{k\geq 1}V_{k;a}$ where two nodes u and v are neighbors if and only if $\ell(u,v)=1$. Observe that all nodes in $\mathbb{L}_a$ have finite length.

Varshney, Kusuma, and Goyal [31] implicitly use a notion of Levenshtein graph similar to ours, and point out that, unlike Hamming graphs, Levenstein graphs cannot be represented as Cartesian products—which makes their study particularly difficult. Various other notions of Levenshtein graphs have been considered in the literature, usually motivated by specific applications. One common definition is that two nodes are neighbors when their Levenshtein distance is underneath some bound. For instance, Pisanti, Et, and Diderot [23] define Levenshtein graphs over a vertex set of arbitrary genes, and two genes u and v are joined by an edge



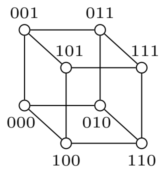


Figure 3: Visual representation of $\mathbb{L}_{0,1;3}$ (left), and $\mathbb{L}_{3,3;2}$ (right). Note that graph $\mathbb{L}_{3,3;2}$ and $\mathbb{H}_{3;2}$ are isomorphic.

when $\ell(u, v) \leq t$; which they use to test random graphs as viable models for genomic data. Instead, Sala et al. [24] define the vertex set of Levenshtein graphs as $\{0, \ldots, a-1\}^k$, and u and v are neighbors only when $\ell(u, v) \leq 2t$; they use this to help expand on information about the number of common subsequences and supersequences a pair of strings have. Zhong, Heinicke, and Rayner [33] define the vertex set of the Levenshtein graph to have nodes corresponding to microRNAs in mice and people, and u and v are connected by an edge only when $\ell(u, v) \leq 3$. Finally, Stahlberg [27] defines the vertex set of Levenshtein graphs from all strings of a given set M as well as all strings that lie on a shortest path between two strings in M, and nodes u and v are then joined by an edge if and only if $\ell(u, v) = 1$.

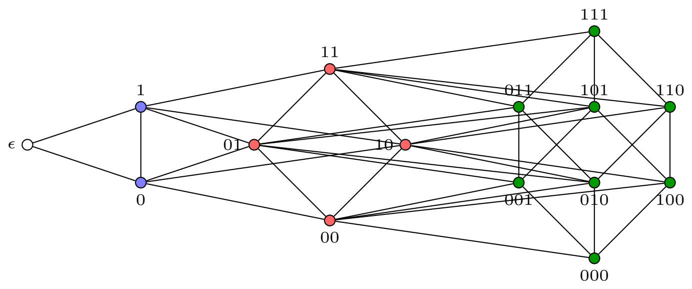


Figure 4: Visual representation of $\mathbb{L}_{3;2}$. The sub-graphs of all strings of fixed length are Hamming graphs: the white, blue, red, and green nodes form $\mathbb{H}_{0,2}$, $\mathbb{H}_{1,2}$, $\mathbb{H}_{2,2}$, and $\mathbb{H}_{3,2}$, respectively. Observe that $\ell(010, 101) = 2 < 3 = h(010, 101)$.

3.2 Geodesic versus edit distance & Connectivity

The geodesic distance between pairs of nodes in a Hamming graph is equal to their Hamming distance, however, the Hamming distance between strings of equal length is not necessarily equal to their Levenshtein distance. Consequently, the geodesic distance between pairs of nodes in an arbitrary Levenshtein graph is not necessarily their edit distance (see Figure 4). In the remainder of this section we characterize the Levenshtein graphs on which the geodesic and edit distance coincide between all pairs of nodes.

Ahead, the length of a path is understood as the number edges that compose it.

Lemma 3.2. *Let $k_1 < k_2$. For all nodes u and v in $\mathbb{L}_{k_1,k_2;a}$, there is a path of length $\ell(u, v)$ that connects u with v. In particular, $\mathbb{L}_{k_1,k_2;a}$ is connected, and for all $u, v \in V_{k_1,k_2;a}$, $d(u, v) \leq \ell(u, v)$.*

Proof. We show something more general, namely, for any alignment between two nodes $\mathbb{L}_{k_1,k_2;a}$, there is a path of the same length as the alignment score that connects them, while visiting only nodes of a length between the shortest and longest of the two.

Consider a fixed alignment A between two nodes u and v. Define $\delta := |u| - |v|$. Since alignment scores are invariant under permutations of their rows, as well as their columns, we may assume without any loss of generality that $|u| \geq |v|$, and that A is of the form:

$$A = \begin{array}{c|c|c|c} u_0 & u_1 & u_2 & -^k \\ v_0 & -^\delta & -^k & v_2 \end{array} ;$$

where the u_i's and v_i's are nodes in $\mathbb{L}_{k_1,k_2;a}$ such that $|u_0| = |v_0| \geq 0$, $|u_1| = \delta$, $|u_2| = |v_2| = k$ for some $k \geq 0$, and $-^n$ denotes n consecutive gaps.

Let s_0 denote the score of the alignment associated with u_0 and v_0 above. Clearly, we can construct a path of length s_0 from $u = u_0u_1u_2$ to $v_0u_1u_2$ substituting, one at a time, the mismatched characters in u_0 by the corresponding characters in v_0. Since substitutions do not alter the length of a node, all nodes in this path have length $|u|$.

Next, we can construct a path of length δ from $v_0u_1u_2$ to v_0u_2 deleting, one at a time, the characters in u_1. In particular, the nodes in this path have a (decreasing) length between $|v_0u_1u_2| = |u|$ and $|v_0u_2| = |v|$, inclusive.

To complete the proof we use the notation $w_{(n)}$ and $w^{(n)}$ to denote the prefix and suffix of length n of a word w, respectively.

Finally, we can construct a path of length $2k$ from v_0u_2 to $v_0v_2 = v$, stitching the following paths of length 2. When $|v| < k_2$, each of these paths is obtained by inserting a character from v_2, and subsequently deleting another in u_2. As a result, all nodes in these paths have a length between $|v|$ and $|v| + 1 \leq k_2$, inclusive. The short paths are:

$$\begin{array}{l} v_0\, u_2^{(k)}\, v_{2(0)},\ v_0\, u_2^{(k-1)}\, v_{2(0)},\ v_0\, u_2^{(k-1)}\, v_{2(1)}; \\ v_0\, u_2^{(k-1)}\, v_{2(1)},\ v_0\, u_2^{(k-2)}\, v_{2(1)},\ v_0\, u_2^{(k-2)}\, v_{2(2)}; \\ \vdots \\ v_0\, u_2^{(1)}\, v_{2(k-1)},\ v_0\, u_2^{(0)}\, v_{2(k-1)},\ v_0\, u_2^{(0)}\, v_{2(k)}. \end{array}$$

Similarly, when $|v| = k_2$, each of these paths is obtained by deleting a character in v_2, and subsequently inserting a character from u_2. All nodes in these paths have a length between $|v|$ and $|v| - 1 \geq k_1$ inclusive.

Appending all the previous paths, we obtain a path from u to v of length $s_0 + \delta + 2k$, which is precisely the score of A. This shows the lemma because each node in this path is contained in $\mathbb{L}_{k_1,k_2;a}$. $\square$

Lemma 3.3. *Let $k_1 < k_2$. For all nodes u and v in $\mathbb{L}_{k_1,k_2;a}$, $d(u, v) \geq \ell(u, v)$.*

Proof. Clearly, $d(u, v) = 0$ if and only if $\ell(u, v) = 0$. Thus, without loss of generality, we may assume that $n := d(u, v) \geq 1$. Due to Lemma 3.2, n is finite; in particular, there is in $\mathbb{L}_{k_1,k_2;a}$ a (simple) path $w_0 = u, \ldots, w_n = v$ of length n that connects u and v. Since $d(w_i, w_{i+1}) = \ell(w_i, w_{i+1}) = 1$, the triangular inequality implies that:

$$d(u, v) = \sum_{i=0}^{n-1} d(w_i, w_{i+1}) = \sum_{i=0}^{n-1} \ell(w_i, w_{i+1}) \geq \ell(u, v),$$

which shows the lemma. $\square$

Lemma 3.4. *For all $k \geq 0$, $\mathbb{L}_{k,k;a} = \mathbb{H}_{k;a}$; in particular, $\mathbb{L}_{k,k;a}$ is connected. Further, the geodesic distance between every pair of nodes on $\mathbb{L}_{k,k;a}$ is equal to their Levenshtein distance if and only if $k \leq 2$.*

Proof. To show the first claim, it suffices to show that $\mathbb{L}_{k,k;a}$ and $\mathbb{H}_{k,a}$ have the same edges. Indeed, if $h(u, v) = 1$ then u and v can be aligned perfectly except for one mismatch. In particular, $\ell(u, v) \leq 1$. But, since $u \neq v$, $\ell(u, v) > 0$, hence $\ell(u, v) = 1$. Conversely, if $\ell(u, v) = 1$ then an optimal alignment between u and v consists of a single mismatch, or a single indel. Since the latter is not possible because $|u| = |v|$, $h(u, v) = 1$, which shows the claim.

Due to the first claim, $d(u, v) = h(u, v)$ for all pairs of nodes u, v in $\mathbb{L}_{k,k;a}$. We use this to show the second claim, assuming, without loss of generality, that $u \neq v$.

The second claim is trivial when $k = 0$. If $k = 1$ then, as we argued before, $\ell(u, v) = 1 = h(u, v) = d(u, v)$. Instead, if $k = 2$ and $h(u, v) = 1$ then, as we just argued, $\ell(u, v) = 1 = h(u, v) = d(u, v)$. Otherwise, if $k = 2$ but $h(u, v) = 2$ then Lemma 3.3 implies that $0 < \ell(u, v) \leq 2$; however, $\ell(u, v) = 1$ is not possible because the optimal alignment between u and v would then have to use a single indel, which in turn is not possible because u and v are of the same length. Hence, $\ell(u, v) = 2$ and again $\ell(u, v) = h(u, v) = d(u, v)$.

Finally, if $k > 2$, and since $a \geq 2$, there is in $\mathbb{L}_{k,k;a}$ a node u of length k formed by alternating 0's and 1's. Let v be the flip of u. Then $h(u, v) = k$ but $\ell(u, v) \leq 2$ because the strings $-u$ and $v-$ align perfectly except for their ends; in particular, $h(u, v) > \ell(u, v)$ i.e. $d(u, v) > \ell(u, v)$. ☐

Together, Lemmas 3.2–3.4 imply the following result.

Corollary 3.4.1. *Levenshtein graphs are connected, and the geodesic distance between every pair of nodes on $\mathbb{L}_{k_1,k_2;a}$ is equal to their Levenshtein distance if and only if $k_1 < k_2$ or $k_1 = k_2 \leq 2$. Otherwise, if $k > 2$ then the geodesic distance in $\mathbb{L}_{k,k;a}$ is the Hamming distance.*

4 Runs and Resolvability of Levenshtein Graphs

A run is a maximal substring of a single repeated character in a string. In this section, we study the Levenshtein distance between an arbitrary string and another with limited runs. This will prove useful for studying the resolvability of Levenshtein graphs and in turn, their automorphism group. In fact, in most cases, it is sufficient to limit one string to either one or two runs.

In what follows, the total number of occurrences of an alphabet character α in a string w is denoted $N_\alpha(w)$, whereas the number of runs in w is denoted $r(w)$. For example, $N_0(01121) = 1$, $N_1(01121) = 3$, $N_2(01121) = 1$, and $r(01121) = 4$.

4.1 Edit distance to a single run string

Lemma 4.1. *For all strings w, if $k \geq 0$ and α is an alphabet character then*

$$\ell(w, \alpha^k) = \max\{|w|, k\} - \min\{N_\alpha(w), k\}.$$

Proof. Assume that $k > 0$, otherwise the statement is trivial. The score of an alignment is its length minus the number of matches in it. But the length of an alignment is at least the length of the longest string, and the number of matches is at most the number of characters shared by the strings. In particular, since the Levenshtein distance between w and α^k is the score of some optimal alignment, we have that: $\ell(w, \alpha^k) \geq \max\{|w|, k\} - \min\{N_\alpha(w), k\}$. To complete the proof, it suffices to expose an alignment with the same score as the right-hand side of this inequality.

Let $n := N_\alpha(\omega)$. Assume first that α^n is a prefix of w. We now consider two cases. If $|w| \leq k$ then $w = \alpha^n u$, with $N_\alpha(u) = 0$, and the following alignment between w and α^k has the desired score:

$$\begin{array}{c|c|c|} \alpha^n & u & -^{k-|w|} \\ \alpha^n & \alpha^{|w|-n} & \alpha^{k-|w|} \end{array}\,.$$

Otherwise, if $|w| \geq k$, let $\delta = \min\{n, k\}$ and write $w = \alpha^\delta uv$, with $|u| = k - \delta$ and $|v| = |w| - k$. Now, the following alignment has the desired score:

$$\begin{array}{c|c|c} \alpha^\delta & u & v \\ \alpha^\delta & \alpha^{k-\delta} & -^{|w|-k} \end{array}\,.$$

The previous argument assumes that α^n is a prefix of w. If this is not the case, we may shuffle the columns of the alignments to reproduce w on the top row but without altering their scores. From this, the lemma follows. $\square$

4.2 Edit distance to a two runs string

Lemma 4.2. *Let $k, l, r \geq 0$ be integers. If $w = w_1 \cdots w_k$ is a string of length k and α, β are different alphabet characters then*

$$\ell(w, \alpha^l \beta^r) = \min_{i_0 \leq i \leq i_1} \left\{ \ell(w_1 \cdots w_i, \alpha^l) + \ell(w_{i+1} \cdots w_k, \beta^r) \right\},$$

where $i_0 := \max\{0, \min\{l, k - r\}\}$ and $i_1 := \min\{k, \max\{l, k - r\}\}$.

Proof. Define $l_i := N_\alpha(w_1 \cdots w_i)$ and $r_i := N_\beta(w_{i+1} \cdots w_k)$, for $0 < i < k$. Further, define $l_i := 0$ and $r_i := N_\beta(w)$ for $i \leq 0$, and $l_i := N_\alpha(w)$ and $r_i := 0$ for $i \geq k$.

Any alignment A between w and $\alpha^l \beta^r$ may be segmented as

$$A = \begin{array}{c|c} u_0 & u_1 \\ v_0 & v_1 \end{array}\,,$$

where u_0 and u_1 correspond to a possibly empty prefix and suffix of w, respectively, and v_0 and v_1 correspond to the strings α^l and β^r, respectively. (u_0, u_1, v_0, v_1 may contain $-$'s.) Since this also applies to an optimal alignment between w and $\alpha^l \beta^r$, it follows that

$$\begin{aligned} \ell(w, \alpha^l \beta^r) &= \min_{0 \leq i \leq k} \quad \ell(w_1 \cdots w_i, \alpha^l) + \ell(w_{i+1} \cdots w_k, \beta^r) \\ &= \min_{0 \leq i \leq k} \quad \max\{l, i\} - \min\{l, l_i\} + \max\{r, k - i\} - \min\{r, r_i\} \\ &= \min_{0 \leq i \leq k} \quad \frac{k + |l - i| + |k - r - i| + |l - l_i| - l_i + |r - r_i| - r_i}{2}, \end{aligned}$$

where for the second identity we have used Lemma 4.1, and for the third one the well-known identities $\max\{a,b\} = (a+b+|a-b|)/2$, and $\min\{a,b\} = (a+b-|a-b|)/2$.

Consider the functions $f_1, f_2 : \mathbb{Z} \to \mathbb{Z}$ defined as

$$f_1(i) := \frac{k-l-r}{2} + \frac{|l-i| + |k-r-i|}{2}$$
$$f_2(i) := \frac{|l-l_i| + l - l_i}{2} + \frac{|r-r_i| + r - r_i}{2}.$$

In particular, $\ell(w, \alpha^l\beta^r) = \min_{0\le i\le k} f_1(i) + f_2(i)$. Next we show that this minimum is achieved at some $i_0 \le i \le i_1$.

Observe that up to a constant summand, $f_1(i)$ is the average of the distance from i to l, and from i to $k-r$. So $f_1(i)$ is strictly decreasing for $i \le \min\{i, k-r\}$, and strictly increasing for $\max\{i, k-r\} \le i$. In particular, when restricted to the domain $\{0,\dots,k\}$, f_1 is monotone decreasing to the left of i_0 and monotone increasing to the right of i_1.

On the other hand, observe that $f_2(i) = g(l-l_i) + g(r-r_i)$, where

$$g(x) := \frac{|x| + x}{2}, \text{ for } x \in \mathbb{Z};$$

satisfies $|g(x) - g(x-1)| \le 1$. In particular, if $w_{i+1} = \alpha$ then $|f_2(i+1) - f_2(i)| \le 1$ because $l_{i+1} = l_i + 1$ and $r_{i+1} = r_i$. Similarly, if $w_{i+1} = \beta$ then $|f_2(i+1) - f_2(i)| \le 1$ because $l_{i+1} = l_i$ and $r_{i+1} = r_i - 1$. Finally, if $w_{i+1} \notin \{\alpha, \beta\}$ then $l_{i+1} = l_i$ and $r_{i+1} = r_i$, hence $f_2(i+1) = f_2(i)$. In either case, we find that $|f_2(i+1) - f_2(i)| \le 1$ for $0 \le i < k$. As a result, since f_1 is integer-valued, $f_1 + f_2$ is decreasing for $i \le i_0$ but increasing for $i_1 \le i$, from which the lemma follows. ☐

Remark 4.3. *The proof of Lemma 4.2 can be adapted into an algorithm that finds the distance between an arbitrary string u to a string of the form $v = \alpha^l\beta^r$ in $O(|u|)$ time. This involves first noting that $f_1(i) = |u| - l - r$ for $i_0 \le i \le i_1$, reducing the problem to minimizing f_2 over the restricted domain. This can be done through a loop where $f_2(i_0)$ can be found directly and the remaining values can be found recursively by finding $f_2(i+1) - f_2(i)$ through cases depending on l_i, r_i, and u_{i+1}. This is faster than standard methods of finding the edit distance between strings which is of complexity $O(|u||v|)$, and beats some more sophisticated methods as well. Indeed, by noting that $\ell(u,v) = O(\max\{|u|,|v|\})$, this is faster than the diagonal method of order $O(\ell(u,v) \cdot \min\{|u|,|v|\})$ [30].*

A number of papers suggest methods for effectively find the distance between run-length encoded strings with the standard dynamic programming approach, including [2, 20].

By conditioning on the length of the input strings in lemmas 4.1-4.2, we obtain the following noteworthy result.

Corollary 4.3.1. *If u and v are strings of the same length, and u or v have at most two runs, then $\ell(u,v) = h(u,v)$.*

Proof. Suppose that $|u| = |v| = k$, and write $u = u_1 \cdots u_k$ with $u_1, \dots, u_k$ alphabet characters. Without any loss of generality assume that $r(v) \le 2$.

If $r(v) = 0$ then $u = v$; in particular, $\ell(u,v) = 0 = h(u,v)$. Instead, if $r(v) = 1$ then $v = \alpha^k$ for some alphabet character α, and Lemma 4.1 implies that

$$\ell(u,v) = k - N_\alpha(u) = \sum_{i=1}^{k} [\![u_i \ne \alpha]\!] = h(u,v).$$

Finally, if $r(v) = 2$ then $v = \alpha^l \beta^{k-l}$ for some integer $1 \le l < k$ and alphabet characters $\alpha \ne \beta$. Hence, from Lemma 4.2, and the previous result for when $r(v) = 1$, we find that

$$\begin{aligned}\ell(u, v) &= \ell(u_1 \cdots u_l, \alpha^l) + \ell(u_{l+1} \cdots u_k, \beta^{k-l})\\ &= h(u_1 \cdots u_l, \alpha^l) + h(u_{l+1} \cdots u_k, \beta^{k-l}) = h(u, v),\end{aligned}$$

as claimed. □

5 Metric Dimension of Levenshtein Graphs

Recall that a subset of nodes R in a graph G is said to *resolve* a pair of nodes u and v if there exists $r \in R$ such that $d(u, r) \ne d(v, r)$. Moreover, R is said to *resolve the graph* when it resolves all pairs of different nodes. Lastly, recall that the metric dimension of a graph, $\beta(G)$, is the size of the smallest resolving set.

In this section we show that

$$O\left(\frac{k_2}{\log_a k_2}\right) \le \beta(\mathbb{L}_{k_1,k_2;a}) \le O\left(ak_2(k_2 - k_1 + 1)\right).$$

Note, if $\Delta = k_2 - k_1 + 1 \approx k_2$, then $R_{k_1,k_2;a}$ has quadratic growth in terms of the maximum string length k_2. However, if Δ remains fixed then $R_{k_1,k_2;a}$ grows linearly with the largest string length.

5.1 Metric dimension lower bound

By definition, if the diameter of a graph is δ then for every pair of nodes u and v, $d(u, v) \in \{0, 1, \ldots, \delta\}$ (where $d(u, v) = 0$ if and only if $u = v$). Thus, embedding a node u through a resolving set R maps it to an element of the set $\{0, 1, \ldots \delta\}^{|R|}$. Since this mapping is one-to-one we arrive at the following bound.

Lemma 5.1. *([16], Theorem 3.6) If $G = (V, E)$ is a graph with metric dimension β and diameter δ then $|V| \le \delta^\beta + \beta$.*

Corollary 5.1.1. *If $k_2 \ge k_1$ and $a \ge 2$ are non-negative integers then $\beta(\mathbb{L}_{k_1,k_2;a}) \ge \frac{k_2}{\log_a(k_2+1)}$.*

Proof. Let $\beta = \beta(\mathbb{L}_{k_1,k_2;a})$. Observe that the diameter of $\mathbb{L}_{k_1,k_2;a}$ is at most k_2 because $\ell(u, v) \le \max\{|u|, |v|\}$ for all pair of strings u and v. As a result, using Lemma 5.1, we obtain that

$$a^{k_2} = |V_{k_2,k_2;a}| \le |V_{k_1,k_2;a}| \le k_2^\beta + \beta \le (k_2 + 1)^\beta.$$

The result follows now taking the logarithm in base a. □

It may seem like Corollary 5.1.1 neglects the parameter k_1. However, using that $a \ge 2$, we find that

$$|V_{k_1,k_2;a}| = \sum_{i=k_1}^{k_2} a^i \le a^{k_2} \sum_{j=0}^{\infty} a^{-j} = \frac{a^{k_2}}{1 - 1/a} \le 2a^{k_2},$$

i.e., our estimate of $|V_{k_1,k_2;a}|$ is within a factor of 2 of its true value.

By setting $k_1 = k_2$, Corollary 5.1.1 may be applied to Hamming graphs as well. In this case, the lower bound of the Corollary is within a factor of 2 of the true asymptotic value (see Section 2.1). Accordingly, we regard the lower bound of the Corollary a benchmark for the metric dimension of Levenshtein graphs.

5.2 Metric dimension upper bound

In this section we construct a resolving set $\mathbb{L}_{k_1,k_2;a}$; in particular, this set cardinality is an upper bound the metric dimension of $\mathbb{L}_{k_1,k_2;a}$. We start by constructing a set nodes that resolves any pair of distinct nodes of equal length.

Lemma 5.2. *In $\mathbb{L}_{k_1,k_2;a}$ any pair of different strings of a same length $k_1 \leq k \leq k_2$ is resolved by the set*

$$R_{k,a} := \bigcup_{n=0}^{\lfloor a/2 \rfloor - 1} \Big\{ (2n)^w (2n+1)^{k-w} : w = 0, \ldots, k \Big\}. \tag{5}$$

Proof. Let $u = u_1 \cdots u_k$ and $v = v_1 \cdots v_k$ be nodes in $\mathbb{L}_{k_1,k_2;a}$ of length k that differ at position j. Define $\alpha := u_j$. Without loss of generality assume that $\alpha \neq (a-1)$ when a is odd.

Due to Corollary 3.4.1, the geodesic distance between pairs of nodes in $\mathbb{L}_{k_1,k_2;a}$ is either their Hamming or Levenshtein distance. But, since nodes in $R_{k,a}$ have at most two runs, Corollary 4.3.1 implies that $\ell(u,r) = h(u,r)$ and $\ell(v,r) = h(v,r)$, for each $r \in R$. In other terms, the geodesic distance between u and v to the nodes of $R_{k,a}$ is always the Hamming distance.

If α is even, we claim that $\{\alpha^{j-1}(\alpha+1)^{k-j+1}, \alpha^j(\alpha+1)^{k-j}\}$ resolves u and v. By contradiction suppose otherwise, i.e. assume that

$$\begin{aligned} d(u, \alpha^{j-1}(\alpha+1)^{k-j+1}) &= d(v, \alpha^{j-1}(\alpha+1)^{k-j+1}), \text{ and} \\ d(u, \alpha^{j}(\alpha+1)^{k-j}) &= d(v, \alpha^{j}(\alpha+1)^{k-j}). \end{aligned}$$

If we define δ as the geodesic distance between u or v and $\alpha^{j-1}(\alpha+1)^{k-j+1}$ then

$$\begin{aligned} d(u, \alpha^j(\alpha+1)^{k-j}) &= h(u, \alpha^j(\alpha+1)^{k-j}) \\ &= \sum_{i=1}^{j} [\![u_i \neq \alpha]\!] + \sum_{i=j+1}^{k} [\![u_i \neq \alpha+1]\!] \pm [\![u_j \neq \alpha+1]\!] \\ &= \delta - 1. \end{aligned}$$

On the other hand, since $v_j \neq \alpha$:

$$\begin{aligned} d(v, \alpha^j(\alpha+1)^{k-j}) &= h(v, \alpha^j(\alpha+1)^{k-j}) \\ &= \sum_{i=1}^{j} [\![v_i \neq \alpha]\!] + \sum_{i=j+1}^{k} [\![v_i \neq \alpha+1]\!] \pm [\![v_j \neq \alpha+1]\!] \\ &= 1 - [\![v_j \neq \alpha+1]\!] + \delta \\ &\geq \delta, \end{aligned}$$

implying that $d(u, \alpha^j(\alpha+1)^{k-j}) \neq d(v, \alpha^j(\alpha+1)^{k-j})$, which is a contradiction. So, $\{\alpha^{j-1}(\alpha+1)^{k-j+1}, \alpha^j(\alpha+1)^{k-j}\}$ resolves u and v.

Likewise, if α is odd, one can show that $\{(\alpha-1)^{i-1}\alpha^{k-i+1}, (\alpha-1)^i\alpha^{k-i}\}$ resolves u and v. In either case, we have found a subset of $R_{k,a}$ that resolves u and v, which shows the lemma. $\square$

Corollary 5.2.1. *$\mathbb{L}_{k_1,k_2;a}$ is resolved by a set of size $O\left(ak_2(k_2 - k_1 + 1)\right)$.*

Proof. Consider the set of nodes

$$R := \begin{cases} \cup_{k=k_1}^{k_2} R_{k,a} \cup \{(a-1)^{k_2}\}, & \text{if } a \text{ is odd;} \\ \cup_{k=k_1}^{k_2} R_{k,a}, & \text{if } a \text{ is even.} \end{cases}$$

We claim that R resolves $\mathbb{L}_{k_1,k_2;a}$. To see this, let u and v be distinct vertices of $\mathbb{L}_{k_1,k_2;a}$. If $|u| = |v| = k$ then u and v are resolved by $R_{k,a} \subseteq R$. Instead, if $|u| \neq |v|$, we claim that $\{0^{k_2}, 1^{k_2}, \ldots, (a-1)^{k_2}\} \subset R$ resolves u and v. Otherwise, since $k_2 \geq |u|, |v|$, Lemma 4.1 would imply that

$$|u| = \sum_{\alpha=0}^{a-1} N_\alpha(u) = \sum_{\alpha=0}^{a-1}(k_2 - d(\alpha^{k_2}, u)) = \sum_{\alpha=0}^{a-1}(k_2 - d(\alpha^{k_2}, v)) = \sum_{\alpha=0}^{a-1} N_\alpha(v) = |v|,$$

which is not possible. In either case, there is a subset of R that resolves u and v; in particular, R resolves $\mathbb{L}_{k_1,k_2;a}$.

Finally, observe that

$$|R_{k,a}| = \begin{cases} 1, & \text{if } k = 0; \\ \lfloor \frac{a}{2} \rfloor (k+1), & \text{if } k > 0. \end{cases}$$

As a result:

$$\begin{aligned} |R| &\leq 1 + \sum_{k=k_1}^{k_2} |R_{k,a}| \\ &\leq 1 + \left\lfloor \frac{a}{2} \right\rfloor \sum_{k=k_1}^{k_2} (k+1) \\ &= 1 + \left\lfloor \frac{a}{2} \right\rfloor \left(\binom{k_2+2}{2} - \binom{k_1+1}{2} \right), \end{aligned}$$

from which the result follows. □

6 Automorphism Group

In this section we characterize the automorphism group of all Levenstein graphs to aid in their future study. In particular, in Section 6.1 we discuss the automorphism group of general Levenshtein graphs, and in Section 6.2 we verify these results for typical Levenshtein graphs.

6.1 Overview of automorphisms on Levenshtein graphs

In what follows, ρ denotes the *string reversal*, i.e. if $u = u_1 \cdots u_k$ is a string of length $k \geq 1$ then $\rho(u) := u_k \cdots u_1$. By definition, $\rho(\varepsilon) := \varepsilon$. On the other hand, given an alphabet bijection $\xi : \{0, \ldots, a-1\} \to \{0, \ldots, a-1\}$, we define $\xi(u) := \xi(u_1) \cdots \xi(u_k)$ and $\xi(\varepsilon) := \varepsilon$. We refer to any such transformation as a *character bijection*. As expected, we have the following result.

Proposition 6.1. *The string reversal and character bijections are automorphisms of* $\mathbb{L}_{k_1,k_2;a}$.

Proof. Let ξ be a character bijection. Since ξ and ρ preserve string lengths, $\xi(V_{k_1,k_2}) \subset V_{k_1,k_2}$ and $\rho(V_{k_1,k_2}) \subset V_{k_1,k_2}$. Furthermore, since the character bijection associated with the alphabet bijection ξ^{-1} is an inverse for ξ, and ρ is an involution, ξ and ρ are bijections from V_{k_1,k_2} onto itself. It is convenient to extend ξ to strings formed from the enlarged alphabet $\{0, \ldots, a-1, -\}$, defining $\xi(-) = -$. Likewise, extend ρ to strings that may include indels besides alphabet characters.

Let $u, v \in V_{k_1,k_2}$ and A an alignment of length $k \geq 1$ between them:

$$A = \begin{matrix} \alpha_1 & \ldots & \alpha_k \\ \beta_1 & \ldots & \beta_k \end{matrix} .$$

Define

$$\xi(A) := \begin{matrix} \xi(\alpha_1) & \ldots & \xi(\alpha_k) \\ \xi(\beta_1) & \ldots & \xi(\beta_k) \end{matrix} ;$$

which correspond to alignments between $\xi(u)$ and $\xi(v)$. Clearly, score$(\xi(A)) =$ score(A), which implies that $\ell(\xi(u), \xi(v)) \leq \ell(u, v)$, for all $u, v \in V_{k_1,k_2}$ and character bijection ξ. In particular, we can also assert that $\ell(\xi^{-1}(\xi(u)), \xi^{-1}(\xi(v))) \leq \ell(\xi(u), \xi(v))$, implying that $\ell(u, v) = \ell(\xi(u), \xi(v))$. A similar argument shows that $\ell(u, v) = \ell(\rho(u), \rho(v))$, which completes the proof. $\square$

Proposition 6.1 suggests that $\mathbb{L}_{k_1,k_2;a}$ has at least $2^{[\![k_2 \geq 2]\!]} \cdot a!$ automorphisms, where $[\![k_2 \geq 2]\!]$ indicates whether there is a string that is not equal to its reversal (any string with two runs satisfies this). When $k_2 \geq 2$, one can easily verify that each of these automorphisms is unique by observing how they act over the strings $0^{k_2}, 1^{k_2}, \ldots, (a-1)^{k_2}$ and $0^{k_2-1}1$. For finite Levenshtein graphs, the automorphism group can be described as follows (which we refer to as Levenshtein graphs of Type 1, 2, and 3, respectively):

1. $\mathbb{L}_{k_1,k_2;a}$, with $k_1 \neq k_2$ and $k_2 \geq 2$: we show in the next section that the automorphism group contains only the string reversal, character bijections, or compositions of both. In particular, Type 1 Levenshtein graphs graphs have exactly $2 \cdot a!$ automorphisms.

2. $\mathbb{L}_{k,k;a} = \mathbb{H}_{k,a}$: the automorphism group is $\mathbb{A}(\mathbb{H}_{k,a}) = (\times_{i=1}^{k} S_a) \rtimes S_k$ [6], where S_x is the permutation group of $\{0, 1, \ldots, x-1\}$. In other terms, the Hamming graph automorphisms are the composition of character permutations with character-wise alphabet bijections. Accordingly, the automorphism group has size $(a!)^k k!$.

3. $\mathbb{L}_{0,1;a} = K_{a+1}$: in this case, bijections of the nodes onto themselves are the only automorphisms; in particular, the automorphism group has size $(a+1)!$.

Compared to Hamming graphs (Type 2), typical Levenshtein graphs (Type 1) have relatively few automorphisms which, surprisingly, do not depend on k_1 or k_2. Levenshtein graphs of Type 3 form a degenerate case, and are the only Levenshtein graphs where automorphisms do not necessarily preserve string lengths.

6.2 Automorphisms of typical Levenshtein graphs

This section is dedicated to show why the automorphism group of Type 1 Levenshtein graphs is so restrictive in comparison to Type 2. We aim to prove that the automorphisms described in Proposition 6.1 are the only automorphisms available to Levenshtein graphs of Type 1. For this purpose we remind that, since automorphisms preserve edges, they also preserve the node degrees, and the distance between pairs of nodes.

Next, we discuss the degree of nodes on the infinite graph $\mathbb{L}_a$. The next result can be generalized easily to arbitrary Levenshtein graphs by restricting the length of the neighbors of a node.

Recall that the number of runs in a node u is denoted $r(u)$.

Proposition 6.2. *A node u on $\mathbb{L}_a$ has $r(u)$ neighbors of length $|u|-1$, $|u|(a-1)$ neighbors of length $|u|$, and $a+|u|(a-1)$ neighbors of length $|u|+1$. In particular, u has degree $a+r(u)+2|u|(a-1)$.*

Proof. Recall that substitutions keep the length of a node, whereas deletions and insertions reduce and increase, respectively, its length by one unit. In particular, u has $|u|(a-1)$ neighbors of length $|u|$, and $r(u)$ neighbors of length $|u|-1$.

Let us now focus on the neighbors of u that can be reached due to a single insertion. An insertion may either keep or increase the number of runs. The former occurs only if a run is enlarged by one character, and there are $r(u)$ ways to do so. The latter occurs only if a run is split by a character into two, or two consecutive runs are separated by a single-character run, which can be done in $(|u|+1)(a-1)-(r(u)-1)=a+|u|(a-1)-r(u)$ ways. In particular, $r(u)+a+|u|(a-1)-r(u)=a+|u|(a-1)$ nodes can be reached from u through a single insertion. From this, the proposition follows. ☐

The number of strings that can be created by a given number of insertions onto a given string, and a bound on the number of strings that can be formed by a given number of deletions from a given string is discussed in [19]. Surprisingly, the number of strings that can be created by inserting characters into a given string does not depend on the number of runs of the string.

Lemma 6.3. *If $k_1+1<k_2$ then any automorphism of $\mathbb{L}_{k_1,k_2;a}$ preserves the length of strings of length k_2.*

Proof. Let σ be an automorphism of $\mathbb{L}_{k_1,k_2;a}$ (recall the implicit assumption that $a\geq 2$). We claim that $\sigma(V_{k_2,k_2;a})\subset V_{k_1,k_2-2;a}\cup V_{k_2,k_2;a}$. By contradiction suppose that there is a node u such $|u|=k_2$ and $|\sigma(u)|=k_2-1$. Then, due to Proposition 6.2:

$$\begin{aligned}\deg(u)&=r(u)+k_2(a-1)\\ \deg(\sigma(u))&=r(\sigma(u))+a+2(k_2-1)(a-1).\end{aligned}$$

As a result, using that $1\leq r(w)\leq |w|$ for any non-empty string w, we obtain that

$$\begin{aligned}deg(\sigma(u))&\geq 1+a+2(k_2-1)(a-1)\\ &\geq 1+a+(k_2-1)(a-1)+(k_2-1)\\ &=k_2+k_2(a-1)+1\\ &>\deg(u),\end{aligned}$$

which is not possible because automorphisms preserve node degrees.

Finally, we show that $\sigma(V_{k_2,k_2;a})=V_{k_2,k_2;a}$. For this note that no vertex in $V_{k_2,k_2-2;a}$ can be a neighbor of a vertex in $V_{k_2,k_2;a}$ because any alignment between a word of length k_2-2 and another of length k_2 must include at least two indels. On the other hand, since $V_{k_2,k_2;a}$ is the vertex set of $\mathbb{H}_{k_2;a}$, which is a connected sub-graph of $\mathbb{L}_{k_1,k_2;a}$, $\sigma(V_{k_2,k_2;a})$ is the vertex set of a connected subgraph of $\mathbb{L}_{k_1,k_2;a}$. As a result, since $\sigma(V_{k_2,k_2;a})\subset V_{k_1,k_2-2;a}\cup V_{k_2,k_2;a}$, either $\sigma(V_{k_2,k_2;a})\subset V_{k_1,k_2-2;a}$ or $\sigma(V_{k_2,k_2;a})\subset V_{k_1,k_2;a}$. Since the former inclusion is not possible because $|V_{k_1,k_2-2;a}|<|V_{k_1,k_2;a}|$, we must have $\sigma(V_{k_2,k_2;a})\subset V_{k_2,k_2;a}$, which shows the proposition. ☐

Lemma 6.4. *Let $k_1\neq k_2$ and $k_2\geq 2$, and define $X:=\{0^{k_2},\ldots,(a-1)^{k_2}\}$. If σ is an automorphism of $\mathbb{L}_{k_1,k_2;a}$ then $\sigma(X)=X$.*

Proof. Let σ be an automorphism of $\mathbb{L}_{k_1,k_2;a}$.

We first show that $\sigma(X) \subset V_{k_2,k_2;a}$. Due to Lemma 6.3, this is direct when $k_1 + 1 < k_2$. Hence assume that $k_1 + 1 = k_2$; in particular, $V_{k_1,k_2;a} = V_{k_1,k_1;a} \cup V_{k_2,k_2;a}$. Suppose that $\sigma(X) \cap V_{k_1,k_1;a} \neq \emptyset$. Then, there would be $x \in X$ such that $|\sigma(x)| = k_1$. In particular, due to Proposition 6.2, it would follow that

$$\begin{aligned}\deg(\sigma(x)) &= a + 2(k_2 - 1)(a - 1)\\ &> a + (k_2 - 1)(a - 1)\\ &= 1 + k_2(a - 1)\\ &= \deg(x),\end{aligned}$$

which it is not possible because automorphisms preserve node degrees. As a result, $\sigma(X) \cap V_{k_1,k_1;a} = \emptyset$, i.e. $\sigma(X) \subset V_{k_2,k_2;a}$, which shows the claim.

Finally, since $\sigma(X) \subset V_{k_2,k_2;a}$, for each $x \in X$, Proposition 6.2 implies that $\deg(x) = 1 + k_2(a - 1)$ and $\deg(\sigma(x)) = r(\sigma(x)) + k_2(a - 1)$. Since $\deg(x) = \deg(\sigma(x))$, we must have $r(\sigma(x)) = 1$, i.e. $\sigma(x) \in X$, which shows the lemma. □

Lemma 6.5. *Let $k_1 \neq k_2$ and $k_2 \geq 2$. If $\sigma(\cdot)$ is an automorphism of $\mathbb{L}_{k_1,k_2;a}$ then the following apply.*

1. *There is a character bijection ξ such that, for every alphabet character α and string $u \in V_{k_1,k_2;a}$, $N_\alpha(u) = N_{\xi(\alpha)}(\sigma(u))$; in particular, $\sigma(\alpha^k) = \xi(\alpha)^k$ for each alphabet character α and $k_1 \leq k \leq k_2$.*
2. *For all $u \in V_{k_1,k_2;a}$, $|\sigma(u)| = |u|$.*
3. *For all $u \in V_{k_1,k_2;a}$ with $|u| = k_2$, $r(\sigma(u)) = r(u)$.*

Proof. Consider an automorphism σ of $\mathbb{L}_{k_1,k_2;a}$, and let X be as in Lemma 6.4. In particular, $\sigma(X) = X$. Since σ is bijective, there exists an alphabet bijection $\xi : \{0, \ldots, a-1\} \to \{0, \ldots, a-1\}$ such that $\sigma(x) = \xi(x)^{k_2}$, for each $x \in X$. As before, we denote the automorphism associated with ξ with the same symbol.

Let α be an alphabet character, and u a node in $\mathbb{L}_{k_1,k_2;a}$. Since $\alpha^{k_2} \in X$, it follows from Lemma 4.1 that

$$\ell(\sigma(u), \sigma(\alpha^{k_2})) = \ell(\sigma(u), \xi(\alpha)^{k_2}) = k_2 - N_{\xi(\alpha)}(\sigma(u)).$$

Since $\ell(u, \alpha^{k_2}) = k_2 - N_\alpha(u)$, and we must have $\ell(u, \alpha^{k_2}) = \ell(\sigma(u), \sigma(\alpha^{k_2}))$, Property 1 follows. From this, Property 2 is immediate because

$$|u| = \sum_{\alpha=0}^{a-1} N_\alpha(u) = \sum_{\alpha=0}^{a-1} N_{\xi(\alpha)}(\sigma(u)) = |\sigma(u)|.$$

Finally, due to Property 2 and Lemma 4.1, if $|u| = k_2$ then $\deg(\sigma(u)) = r(\sigma(u)) + k_2(a - 1)$. Likewise, $\deg(u) = r(u) + k_2(a - 1)$. In particular, $r(u) = r(\sigma(u))$ because $\deg(u) = \deg(\sigma(u))$, which shows Property 3. □

The next result follows from [19, Theorem 4]. We note that the neighboring nodes in this lemma are called *supersequences* of v because each is obtained inserting a character into it.

Lemma 6.6. *(Adjusted from [19].) A node v in $\mathbb{L}_a$ is uniquely determined by three of its different neighbors of length $|v| + 1$.*

Now we have all the ingredients to prove the following.

Theorem 6.7. *Let $k_1 \neq k_2$ and $k_2 \geq 2$. An automorphism of $\mathbb{L}_{k_1,k_2;a}$ is a character bijection, string reversal, or a composition of both. In particular, $|\mathbb{A}(L_{k_1,k_2;a})| = 2\,a!$.*

Proof. Let σ be an automorphism of $\mathbb{L}_{k_1,k_2;a}$, and ξ be the corresponding character bijection described in Lemma 6.5. Observe that $(\xi^{-1} \circ \sigma)$ preserves character counts because, due to property (1) in the lemma, $N_\alpha(u) = N_\alpha((\xi^{-1} \circ \sigma)(u))$ for each character α and $u \in V_{k_1,k_2;a}$.

Next observe the string $0^{k_2-1}1$. From properties (2) and (3) in Lemma 6.5, we find that $(\xi^{-1} \circ \sigma)(0^{k_2-1}1)$ is a string of length k_2 with two runs. In particular, since $(\xi^{-1} \circ \sigma)$ preserves character counts, $(\xi^{-1} \circ \sigma)(0^{k_2-1}1) \in \{0^{k_2-1}1, 10^{k_2-1}\}$. If $(\xi^{-1} \circ \sigma)(0^{k_2-1}1) = 10^{k_2-1}$, define $\psi := \rho$, otherwise define ψ to be the identity. In either case, ψ is its own inverse; in particular, if we define

$$\iota := \psi \circ \xi^{-1} \circ \sigma = \psi^{-1} \circ \xi^{-1} \circ \sigma,$$

then

$$\iota(0^{k_2-1}1) = 0^{k_2-1}1. \tag{6}$$

We aim to show next that ι is the identity, focusing first on strings of length k_2 with two runs. In fact, note that ι preserves character and run counts because ψ and $(\xi^{-1} \circ \sigma)$ do. Hence, if $\alpha \neq \beta$ are characters and $0 < k < k_2$ then

$$\iota(\alpha^{k_2-k}\beta^k) \in \{\alpha^{k_2-k}\beta^k, \beta^k\alpha^{k_2-k}\}. \tag{7}$$

First, let $\alpha = 0$ and $\beta = 1$. Assume that $\iota(0^{k_2-k}1^k) = 1^k0^{k_2-k}$ for some $0 < k < k_2$. Then, using Corollaries 3.4.1 and 4.3.1, and Equation 6, we find the following distances are

$$\begin{aligned} d(0^{k_2-k}1^k, 0^{k_2-1}1) &= h(0^{k_2-k}1^k, 0^{k_2-1}1) = k - 1; \\ d(\iota(0^{k_2-k}1^k), \iota(0^{k_2-1}1)) &= h(1^k0^{k_2-k}, 0^{k_2-1}1) = k + 1; \end{aligned}$$

which is not possible because automorphisms preserve distances. Therefore $\iota(0^{k_2-k}1^k) = 0^{k_2-k}1^k$, for all $0 < k < k_2$.

Second, if $\alpha = 1$, $\beta = 0$, and $\iota(1^{k_2-k}0^k) = 0^k1^{k_2-k}$ for some $0 < k < k_2$, then $\iota(1^{k_2-k}0^k) = 0^k1^{k_2-k} = \iota(0^k1^{k_2-k})$, which is not possible because ι is one-to-one. Therefore $\iota(1^{k_2-k}0^k) = 1^{k_2-k}0^k$, for all $0 < k < k_2$.

Third, let $\alpha \neq 1$ and $\beta = 1$. Assume that $\iota(\alpha^{k_2-k}1^k) \neq \alpha^{k_2-k}1^k$ for some $0 < k < k_2$. Then, due to Equation 7:

$$\begin{aligned} d(\alpha^{k_2-k}1^k, 0^{k_2-k}1^k) &= h(\alpha^{k_2-k}1^k, 0^{k_2-k}1^k) = (k_2 - k)[\![\alpha \neq 0]\!] \ ; \\ d(\iota(\alpha^{k_2-k}1^k), \iota(0^{k_2-k}1^k)) &= h(1^k\alpha^{k_2-k}, 0^{k_2-k}1^k) \\ &= \begin{cases} k_2, & 0 < k < k_2/2 \text{ and } \alpha \neq 0; \\ 2k, & 0 < k < k_2/2 \text{ and } \alpha = 0; \\ 2(k_2 - k), & k_2/2 \leq k < k_2. \end{cases} \end{aligned}$$

In particular, $d(\alpha^{k_2-k}1^k, 0^{k_2-k}1^k) \neq d(\iota(\alpha^{k_2-k}1^k), \iota(0^{k_2-k}1^k))$, which is a contradiction because ι must preserve distances. So, $\iota(\alpha^{k_2-k}1^k) = \alpha^{k_2-k}1^k$ for all $\alpha \neq 1$ and $0 < k < k_2$.

Finally, let $\alpha \neq \beta$ be arbitrary characters in the alphabet. If $\alpha = 1$ let $\gamma = 0$, otherwise let $\gamma = 1$. Through our second and third cases we have shown that $\iota(\alpha^{k_2-k}\gamma^k) = \alpha^{k_2-k}\gamma^k$ for all $0 < k < k_2$. Next, assume that $\iota(\alpha^{k_2-k}\beta^k) \neq \alpha^{k_2-k}\beta^k$ for some $0 < k < k_2$. Then, as we have argued before we find that:

$$\begin{aligned} d(\alpha^{k_2-k}\beta^k, \alpha^{k_2-k}\gamma^k) &= h(\alpha^{k_2-k}\beta^k, \alpha^{k_2-k}\gamma^k) = k[\![\beta \neq \gamma]\!] \ ; \\ d(\iota(\alpha^{k_2-k}\beta^k), \iota(\alpha^{k_2-k}\gamma^k)) &= h(\beta^k\alpha^{k_2-k}, \alpha^{k_2-k}\gamma^k) \\ &= \begin{cases} k_2, & k_2/2 \leq k < k_2 \text{ and } \beta \neq \gamma; \\ 2(k_2 - k), & k_2/2 \leq k < k_2 \text{ and } \beta = \gamma; \\ 2k, & 0 < k < k_2/2. \end{cases} \end{aligned}$$

But then, once again we find that $d(\alpha^{k_2-k}\beta^k, \alpha^{k_2-k}\gamma^k) \neq d(\iota(\alpha^{k_2-k}\beta^k), \iota(\alpha^{k_2-k}\gamma^k))$, which is not possible. Consequently, for all $\alpha \neq \beta$ and $0 < k < k_2$, $\iota(\alpha^{k_2-k}\beta^k) = \alpha^{k_2-k}\beta^k$.

Thus far, we have shown that if u is a string where $|u| = k_2$ and $r(u) \leq 2$ then $\iota(u) = u$.

Let $R_{k_2,a} = \{r_1, \ldots, r_n\}$ be as defined by Equation 5. Note, for any $r_i \in R_{k_2,a}$ that $|r_i| = k_2$ and $r(r_i) = k_2$, implying that $\iota(r_i) = r_i$. Further, from Lemma 5.2, the transformation $\Phi(u) := \big(d(u, r_1), \ldots, d(u, r_n)\big)$ is one-to-one over nodes of length k_2. Consider an arbitrary node u such that $|u| = k_2$. From Theorem 6.5, we know that $|\iota(u)| = k_2$. As a result:

$$\begin{aligned} \Phi(u) &= \big(d(u, r_1), \ldots, d(u, r_n)\big) \\ &= \big(d(\iota(u), \iota(r_1)), \ldots, d(\iota(u), \iota(r_n))\big) \\ &= \big(d(\iota(u), r_1), \ldots, d(\iota(u), r_n)\big) \\ &= \Phi\big(\iota(u)\big). \end{aligned}$$

In particular, since Φ is one-to-one over vectors of length k_2, $\iota(u) = u$ for all node u such that $|u| = k_2$.

Finally, we prove by induction k, with $k_1 \leq k \leq k_2$, that $\iota(v) = v$ for all $v \in V_{k,k_2;a}$. The base case with $k = k_2$ was just shown above. Next, consider a $k_1 \leq k < k_2$ and suppose that $\iota(v) = v$, for all $v \in V_{k+1,k_2;a}$. If $k = 0$, property 2 of Lemma 6.5 implies that $\iota(\epsilon) = \epsilon$; in particular, $\iota(v) = v$ for all $v \in V_{k,k_2;a}$. Instead, if $k > 0$, consider a string u of length k. From Proposition 6.2, u has $a + |u|(a-1) \geq 3$ neighbors of length $k+1$. Let v_1, v_2, and v_3 be different neighbors of u of length $k+1$. By the inductive hypothesis: $\iota(v_i) = v_i$, for $1 \leq i \leq 3$. So, since ι is an automorphism, v_1, v_2, and v_3 are also neighbors of $\iota(u)$. But, from Lemma 6.6, we know that $\iota(u)$ and u can be identified uniquely by these three neighbors. Hence, $\iota(u) = u$ for all $|u| = k$, i.e. $\iota(v) = v$ for all $v \in V_{k,k_2;a}$.

The above shows that $\iota = \psi^{-1} \circ \xi^{-1} \circ \sigma$ is the identity. In particular, $\sigma = \xi \circ \psi$, where ξ is a character bijection and ψ is either the string reversion or the identity. This completes the proof of the theorem. □

7 Determining Sets

In Section 6 we found the automorphism group of an arbitrary Levenshtein graph. To prove Theorem 6.7, we used determining sets on individual subgraphs. While we no longer need to discover the automorphism group of Levenshtein graphs, we have now the tools to find their determining number. Since determining sets are a relatively new topic that are useful for studying automorphisms, we discuss the determining number of Levenshtein graphs and compare it to other known families of graphs. (A graph with a trivial automorphism group has a determining number of 0.)

Following the notation of Section 6, we split Levenshtein graphs into three categories. The determining number of the complete graph K_n (Type 3) is known, specifically $\text{Det}(K_n) = (n-1)$ [3]. Tight bounds on the determining number of Hamming graphs (Type 2) and other Cartesian products of graphs is given in [4].

This Section is devoted to discussing the determining number of Levenshtein graphs of Type 1. In Theorem 6.7, we showed implicitly for these graphs that $\{0^{k_2}, 1^{k_2}, \ldots, (a-1)^{k_2}, 0^{k_2-1}1\}$ is a determining set; in particular, $\text{Det}(\mathbb{L}_{k_1,k_2;a}) \leq a+1$. Furthermore, we showed that any automorphism of these Levenshtein graphs can be written as $\sigma = \mu \circ \psi$, where μ is in the character bijection group and ψ is in the string reversal group. For the rest of this section we commit to the following definition of determining set.

Definition 7.1. *For a graph $G = (V, E)$, a set of nodes $D \subset V$ is called determining when the identity is the only automorphism $\sigma \in \mathbb{A}(G)$ such that $\sigma(x) = x$, for all $x \in D$.*

First, for the degenerate case with $a = k_2 = 2$ and $k_1 \neq k_2$, it can be shown by an exhaustive test that $\text{Det}(\mathbb{L}_{k_1,2;2}) = \lceil \frac{a}{k_2} \rceil + 1 = 2$. In this case, $\{01, 00\}$ is one of a few minimal determining sets. In what remains of this section, we characterize the determining number of the remaining Type 1 Levenshtein graphs.

Lemma 7.2. *If $k_1 \neq k_2$ and $k_2 \geq 2$ then at least $(a-1)$ of the a alphabet characters must be represented in a determining set of $\mathbb{L}_{k_1,k_2;a}$.*

Proof. Let $D = \{d_1, ..., d_n\}$, with $n \geq 1$, be a determining set, and S the set of alphabet characters that occur at least once in D, i.e., $S = \{(d_i)_j : 1 \leq i \leq n, 1 \leq j \leq |d_i|\}$. If $|S| < a-1$ then there would exist at least two distinct alphabet characters $\alpha, \beta \notin S$. Let μ be the character bijection that swaps α and β, i.e. $\mu(\alpha) = \beta$ and $\mu(\beta) = \alpha$, but acts as the identity on every other character. Then, $\mu(d) = d$, for all $d \in D$; in particular, since μ is not the identity, D could not be a determining set. Since this is not possible, $|S| \geq a-1$, which shows the lemma. □

Lemma 7.3. *If $k_1 \neq k_2$ and $k_2 \geq 2$ then* $\text{Det}(\mathbb{L}_{k_1,k_2;a}) \geq \left\lceil \frac{a}{k_2} \right\rceil$.

Proof. Let $D = \{d_1, ..., d_n\}$, with $n \geq 1$, be a determining set, and S the set of alphabet characters that occur at least once in D. Define $\ell_0 = 0$ and $\ell_i = \sum_{j=1}^{i} |d_i|$ for $1 \leq i \leq n$.

We claim that $\ell_n \geq a$. By contradiction, assume that $\ell_n < a$. Since $\ell_n \geq |S|$, Lemma 7.2 implies that $\ell_n = |S| = a-1$. In particular, up to a character bijection, we may assume that $S = \{0, \ldots, a-2\}$, and that $d_i = \ell_{i-1} \ldots (\ell_i - 1)$ for $1 \leq i \leq n$. Consider the character bijection μ such that $\mu(a-1) = a-1$, and $\mu(j) = \ell_i + \ell_{i-1} - 1 - j$ for $\ell_{i-1} \leq j \leq \ell_i - 1$ and $1 \leq i \leq n$. In particular, μ acts as a reversal on each string in D. Then $(\mu \circ \rho)(d_i) = d_i$, for all $1 \leq i \leq n$, hence $(\mu \circ \rho)$ must be the identity. However, this is not possible because $(\mu \circ \rho)(0(a-1)) = (a-1)(a-2)$. Hence $\ell_n \geq a$, which implies the lemma because $n \cdot k_2 \geq \sum_{i=1}^{n} |d_i| = \ell_n \geq a$.

□

Theorem 7.4. *If $k_1 \neq k_2$, $k_2 \geq 2$, and $(k_2, a) \neq (2, 2)$, then* $\text{Det}(\mathbb{L}_{k_1,k_2;a}) = \left\lceil \frac{a}{k_2} \right\rceil$.

Proof. Define $n := \lceil \frac{a}{k_2} \rceil$; in particular, $n \geq 1$. Due to Lemma 7.3, it suffices to construct a determining set of size n, for which we consider three cases. First, if $k_2 \geq a$, define $D := \{d\}$ where

$$d := \begin{cases} 0^{k_2-1}1, & a = 2; \\ 0^{k_2-a+2}1 \cdots (a-2), & a \geq 3. \end{cases}$$

Since at least $a-1$ alphabet characters are represented in d, the identity is the only character bijection that preserves d. On the other hand, if $\sigma = \mu \circ \rho$, where μ is any character bijection then, for $a = 2$, $\sigma(d) = \mu(1)\mu(0)^{k_2-1}$ with $k_2 - 1 \geq 2$; in particular $\sigma(d) \neq d$. Similarly, if $a \geq 3$ then $\sigma(d) = \mu(a-2)\cdots\mu(1)\mu(0)^{k_2-a+2}$ with $k_2 - a + 2 \geq 2$, and again $\sigma(d) \neq d$. Therefore, D is a determining set.

Second, if $2 < k_2 < a$, let $D := \{d_1, \ldots, d_n\}$ be of cardinality n such that $d_1 := 0012\ldots(k_2-2)$, $d_1, \ldots, d_n$ are of length k_2, and every character in $\{0, \ldots, a-2\}$ is used by at least one node in D. Since $a-1$ alphabet characters are represented in D, the identity is the only character bijection that maps each d_i to itself. However, if $\sigma = \mu \circ \rho$, where μ is any character bijection, then $\sigma(d_1) = \mu(k_2-2)\cdots\mu(1)\mu(0)^2 \neq d_1$. So, D is a determining set.

Finally, if $k_2 = 2$; in particular, $a \geq 3$, let $D = \{d_1, \ldots, d_n\}$ be of cardinality n such that $d_1 := 01$, $d_2 := 12$, $d_1, \ldots, d_n$ are of length 2, and every character in $\{0, \ldots, a-2\}$ is used by at least one node in D. Once again, since at least $a-1$ alphabet characters are represented in D, the identity is the only character bijection that maps each d_i to itself. Next, let $\sigma = \mu \circ \rho$, where μ is any character bijection. If $\sigma(01) = 01$ then $\mu(1) = 0$. If this is the case then $\sigma(12) = \mu(2)0 \neq 12$, i.e. either $\sigma(01) \neq 01$ or $\sigma(12) \neq 12$. Hence D is determining and the theorem follows. $\square$

8 DNA Resolving Sets

As an illustrative example, we apply the results from Section 5.2 to genomic data.

First, consider the task of embedding DNA strings of the fixed length $k = 105$. To do this, we will multilaterate the Levenshtein graph $\mathbb{L}_{k,k;4} = \mathbb{H}_{k;4}$. Using Corollary 5.2.1 we can construct the resolving set

$$R_{k;4} = \left\{ A^{\omega}C^{k-\omega}, G^{\omega}T^{k-\omega} : \omega = 0, 1, 2, \ldots, k \right\}$$

of size $2(k+1) = 212$. For comparison, if we were to use the recursive method described in Section 2.1 (where we first set $s = a-1$ and $r = k-1$, with $a = 4$), we would uncover a resolving set of size 211. Additionally, we could use the asymptotic method described in Section 2.1 to construct a resolving set of $\mathbb{H}_{k;a}$ of size 116. While the resolving set produced by Corollary 5.2.1 is the largest of these methods, it is still comparable in size. In contrast, if we were to embed our strings numerically with k-mer counts using a sliding windows of size 8 the resulting dimension of our embedding would have $4^8 = 65536$ dimensions. Instead, using one-hot encodings we would need a vector of dimension $4 \cdot 105 = 420$.

Next, consider the task of representing DNA strings of length between $k_1 = 90$ and $k_2 = 120$ as numerical vectors. This requires studying the metric dimension of the Levenshtein graph $\mathbb{L}_{k_1,k_2;4}$, which can no longer be understood through known methods for Hamming graphs. Applying Corollary 5.2.1 gives the resolving set

$$R_{k_1,k_2;4} = \bigcup_{k=k_1}^{k_2} R_{k;4}$$

of size $2\sum_{k=k_1}^{k_2}(k+1) = 6572$. The number of nodes in $\mathbb{L}_{k_1,k_2;4}$ is $\sum_{k=k_1}^{k_2} 4^k \approx 2.4 \cdot 10^{72}$. Therefore, it is infeasible to use a brute force algorithm to remove nodes of $R_{k_1,k_2;4}$ and verifying the resolvability of the resulting set. However, it is possible to remove approximately half the nodes of $R_{k_1,k_2;4}$ while keeping it resolving. We include the construction here but omit the proof.

Observe the DNA character bijection defined by $\theta(A) := C$, $\theta(C) : G$, $\theta(G) := T$, and $\theta(T) := A$. Then the set

$$R'_{k_1,k_2;4} = \bigcup_{i=0}^{(k_2-k_1)/2} \theta^i(R_{k_2-2i;4})$$

resolves also $\mathbb{L}_{k_1,k_2;4}$ and is of size $2\sum_{i=0}^{(k_2-k_1)/2}(k_2 - 2i + 1) = 3392$. It is possible to remove a few more nodes as follows. Assume u and v are nodes that are unresolved by $X = \{A^{k_2}, C^{k_2}, G^{k_2}, T^{k_2}\}$. Then following the proof of Corollary 5.2.1, we know that $|u| = |v|$. Additionally, for each DNA base α: $\ell(\alpha^{k_2}, u) = k_2 - N_\alpha(u) = k_2 - N_\alpha(v) = \ell(\alpha^{k_2}, u)$, implying that $N_\alpha(u) = N_\alpha(v)$. Lemma 4.1 then implies that $\ell(\alpha^k, u) = \ell(\alpha^k, v)$, for every character α and $k \geq 0$, because $\ell(\alpha^k, w)$ only depends on $|w|$ and $N_\alpha(w)$. Therefore u and v are unresolved by every single run string, and every single run string outside of X is unnecessary. Removing $\{A^k, C^k, G^k, T^k : k = 90, 92, \ldots, 118\}$ accounts for approximately 2% of the nodes in $R'_{k_1,k_2;a}$ resulting in a resolving set of size 3332.

With a substantial amount of training data, one could use dna2vec to find an embedding of k-mers of varying length [22]. This embedding has many nice properties, e.g. addition of vectors from dna2vec [22] is related to concatenation. In contrast, multilateration of Levenshtein graphs has the benefit of creating low dimensional embeddings that are readily available.

9 Discussion and Future Work

Motivated by biological classification problems, we have provided a preliminary study of Levenshtein graphs to show that they can be used to embed symbolic data as numerical vectors. Leveraging strings with at most two runs, we were able to demonstrate multiple properties of Levenshtein graphs including their automorphism group and bounds on their metric dimension. Unfortunately, smaller resolving sets of Levenshtein graphs than those discovered in Sec. 5 and Sec. 8 have remained elusive.

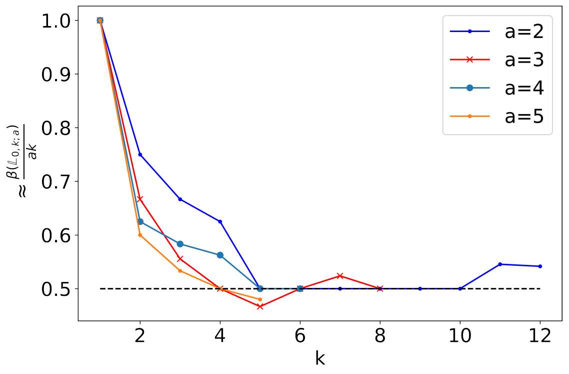


Figure 5: Plot of the approximate metric dimension of $\mathbb{L}_{k;a}$ for various values a and k using the ICH algorithm. These values are normalized by ak and appear to converge to 0.5.

Numerical trials through the ICH algorithm suggest that the true metric dimension of the graph $\mathbb{L}_{k;a}$ may not be quadratic but sublinear in the maximal string length k (see Figure 5). Of course, the data for these trials is limited by the exponential growth of Levenshtein graphs, which has prevented us to test this claim for larger values of the parameters k and a. Nevertheless, we venture to conjecture the following.

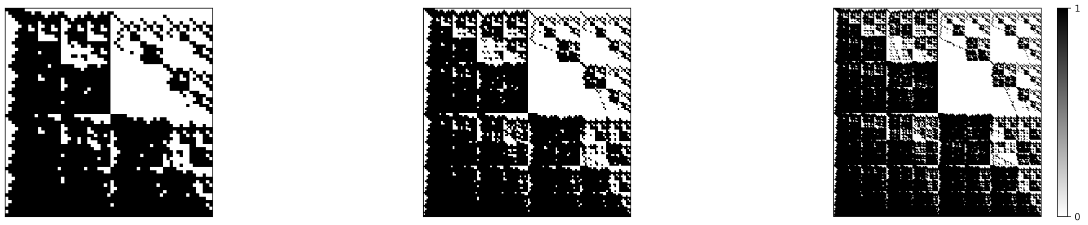


Figure 6: Binary matrices of values $\ell(1u,0v)-\ell(u,v)$ for $u,v\in V_{k;2}$ with $k=5$ (left), $k=6$ (middle), and $k=7$ (right). White pixels denote a 0, black pixels a 1, and in each case the nodes in $V_{k;2}$ are sorted in lexicographic order.

Conjecture 9.1.
$$\beta(\mathbb{L}_{k;a})=o(ak^2).$$

To motivate this claim, we provide an incomplete method for generating resolving sets of Levenshtein graphs. This method relies on the recursion used for calculating edit distances. In particular, if $R=\{r_1,r_2,\dots,r_n\}$ resolves $\mathbb{L}_{k_1,k_2;2}$ then $R^+:=\{0\}\times R=\{0r_1,0r_2,\dots,0r_n\}$ is effective at resolving $\mathbb{L}_{k_1+1,k_2+1;2}$. To generalize this further, note it is trivial to extend a resolving set on $\mathbb{L}_{1,k;2}$ to $\mathbb{L}_{k;2}$. From Ukkonen [30], if α and β are characters and u and v are strings then
$$\ell(\alpha u,\beta v)\in\{\ell(u,v),\ell(u,v)+1\}. \tag{8}$$

When restricted to $\alpha=\beta$, Equation 8 simplifies to $\ell(\alpha u,\alpha v)=\ell(u,v)$. Thus, if the pair $u,v\in V_{k_1,k_2;a}$ is resolved by R, then $0u$ and $0v$ are resolved by R^+. Additionally, one can verify through Lemma 4.2 that every pair of strings with different first character, say $0u$ and $1v$, is resolved by the set $\{01^{k_2},1^{k_2+1}\}$. However, due to Equation 8, strings of the form $1u$ and $1v$ have embeddings of the form (see Equation 1):
$$\Phi_{R^+}(1u)=\Phi_R(u)+\vec{e}_u;$$
$$\Phi_{R^+}(1v)=\Phi_R(v)+\vec{e}_v;$$

where $\vec{e}_u,\vec{e}_v$ are binary vectors of dimension $|R|$. So, it is possible for $1u$ and $1v$ to be not be resolved by R^+—but only when the embedding $\Phi_R(u)$ is within a binary perturbation of $\Phi_R(v)$. By studying these binary perturbation we may be able to understand which pairs of strings are unresolved $R^+\cup\{01^{k_2},1^{k_2+1}\}$ and find new strings that resolve them (see Figure 6). Depending on the number of nodes required to adapt this set into a resolving set of $\mathbb{L}_{k_1+1,k_2+1;a}$, it may be possible to develop resolving sets that satisfy Conjecture 9.1.

To better characterize the metric dimension of Levenshtein graphs, it may be useful to consider a random graph model with a "community structure." Indeed, note that nodes of length i in $\mathbb{L}_{k;a}$ can only connect to nodes of length i and $i\pm1$ (see Figure 4) This resembles a so-called "ordered" Stochastic Block Model (SBM) network. This random graph model splits n nodes into k communities according to a probability vector $p\in(0,1)^k$, where a node is in community i with probability p_i. Further, a pair of nodes in communities i and j are joined by an edge with probability $Q(i,j)$ [1]. In the context of Levenshtein graphs, p_i should be proportional to a^i, and $Q\in[0,1]^{k\times k}$ should be a symmetric matrix such that $Q(i,j)=0$ when $|i-j|>1$. Following Proposition 6.2, we should set $Q(i,i):=\frac{|u|(a-1)}{a^i}$ and $Q(i,i+1):=\frac{|u|(a-1)+a}{a^{i+1}}$.

Due to the various real-world applications of the general SBM, such as email and political blog networks, learning to multilaterate these graphs with high probability may elucidate new node classification algorithms in networks that are well-described by an SBM.

Acknowledgement. This research has been partially funded by the NSF IIS grant No. 1836914.

References

[1] E. Abbe, *Community detection and stochastic block models: recent developments*, The Journal of Machine Learning Research, 18 (2017), pp. 6446–6531.

[2] O. Arbell, G. M. Landau, and J. S. Mitchell, *Edit distance of run-length encoded strings*, Information Processing Letters, 83 (2002), pp. 307 – 314.

[3] D. L. Boutin, *Identifying graph automorphisms using determining sets*, The Electronic Journal of Combinatorics, (2006), pp. R78–R78.

[4] ——, *The determining number of a Cartesian product*, Journal of Graph Theory, 61 (2009), pp. 77–87.

[5] J. Cáceres, D. Garijo, M. L. Puertas, and C. Seara, *On the determining number and the metric dimension of graphs*, The Electronic Journal of Combinatorics, (2010), pp. R63–R63.

[6] F. A. Chaouche and A. Berrachedi, *Automorphisms group of generalized Hamming graphs*, Electronic Notes in Discrete Mathematics, 24 (2006), pp. 9 – 15. Fifth Cracow Conference on Graph Theory USTRON ’06.

[7] G. Chartrand, L. Eroh, M. A. Johnson, and O. R. Oellermann, *Resolvability in graphs and the metric dimension of a graph*, Discrete Applied Mathematics, 105 (2000), pp. 99 – 113.

[8] S. A. Cook, *The complexity of theorem-proving procedures*, in Proceedings of the Third Annual ACM Symposium on Theory of Computing, STOC ’71, New York, NY, USA, 1971, ACM, pp. 151–158.

[9] J. Cáceres, M. C. Hernando, M. Mora, I. Pelayo, M. Puertas, C. Seara, and D. Wood, *On the metric dimension of Cartesian products of graphs*, SIAM J. Discrete Math., 21 (2007), pp. 423–441.

[10] R. Durbin, S. R. Eddy, A. Krogh, and G. Mitchison, *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids*, Cambridge University Press, 1998.

[11] D. Erwin and F. Harary, *Destroying automorphisms by fixing nodes*, Discrete Mathematics, 306 (2006), pp. 3244 – 3252.

[12] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W. H. Freeman & Co., New York, NY, USA, 1979.

[13] F. Harary and R. A. Melter, *On the metric dimension of a graph*, Ars Combin, 2 (1976), p. 1.

[14] M. Hauptmann, R. Schmied, and C. Viehmann, *Approximation complexity of metric dimension problem*, Journal of Discrete Algorithms, 14 (2012), pp. 214 – 222. Selected papers from the 21st International Workshop on Combinatorial Algorithms (IWOCA 2010).

[15] Z. Jiang and N. Polyanskii, *On the metric dimension of Cartesian powers of a graph*, Journal of Combinatorial Theory, Series A, 165 (2019), pp. 1 – 14.

[16] S. Khuller, B. Raghavachari, and A. Rosenfeld, *Landmarks in graphs*, Discrete Applied Mathematics, 70 (1996), pp. 217 – 229.

[17] L. Laird, R. C. Tillquist, S. Becker, and M. E. Lladser, *Resolvability of Hamming graphs*, arXiv preprint arXiv:1907.05974, (2019).

[18] V. I. Levenshtein, *Binary codes capable of correcting deletions, insertions, and reversals*, in Soviet Physics Doklady, vol. 10, 1966, pp. 707–710.

[19] V. I. Levenshtein, *Efficient reconstruction of sequences from their subsequences or supersequences*, J. Comb. Theory Ser. A, 93 (2001), pp. 310–332.

[20] V. Mäkinen, E. Ukkonen, and G. Navarro, *Approximate matching of run-length compressed strings*, Algorithmica, 35 (2003), pp. 347–369.

[21] S. B. Needleman and C. D. Wunsch, *A general method applicable to the search for similarities in the amino acid sequence of two proteins*, Journal of Molecular Biology, 48 (1970), pp. 443–453.

[22] P. Ng, *dna2vec: Consistent vector representations of variable-length k-mers*, 2017.

[23] N. Pisanti, E. Et, and V. D. Diderot, *Recent duplications in genomes: A graph theory approach*, (1998).

[24] F. Sala, R. Gabrys, C. Schoeny, and L. Dolecek, *Three novel combinatorial theorems for the insertion/deletion channel*, in 2015 IEEE International Symposium on Information Theory (ISIT), IEEE, 2015, pp. 2702–2706.

[25] P. J. Slater, *Leaves of trees*, Congr. Numer, 14 (1975), p. 37.

[26] S. Söderberg and H. S. Shapiro, *A combinatory detection problem*, The American Mathematical Monthly, 70 (1963), pp. 1066–1070.

[27] F. Stahlberg, *Discovering Vocabulary of a Language through Cross-Lingual Alignment*, PhD thesis, Karlsruhe Institute of Technology, 2011.

[28] R. C. Tillquist, R. M. Frongillo, and M. E. Lladser, *Metric Dimension*, Scholarpedia, 14 (2019), p. 53881. revision #190769.

[29] R. C. Tillquist and M. E. Lladser, *Low-dimensional representation of genomic sequences*, Journal of Mathematical Biology, 79 (2019), pp. 1–29.

[30] E. Ukkonen, *Algorithms for approximate string matching*, Information and Control, 64 (1985), pp. 100–118.

[31] L. R. Varshney, J. Kusuma, and V. K. Goyal, *On palimpsests in neural memory: An information theory viewpoint*, IEEE Transactions on Molecular, Biological and Multi-Scale Communications, 2 (2016), pp. 143–153.

[32] R. A. Wagner and M. J. Fischer, *The string-to-string correction problem*, Journal of the ACM (JACM), 21 (1974), pp. 168–173.

[33] X. Zhong, F. Heinicke, and S. Rayner, *miRBaseMiner, a tool for investigating miRBase content*, RNA biology, 16 (2019), pp. 1534–1546.