Metric Dimension of Hamming Graphs and Applications to Computational Biology

Lucas Laird

Abstract

Genetic sequencing has become an increasingly affordable and accessible source of genomic data in computational biology. This data is often represented as k-mers, i.e., strings of some fixed length k with symbols chosen from a reference alphabet. In contrast, some of the most effective and well-studied machine learning algorithms require numerical representations of the data. This motivates the development of methods to embed k-mers into real vector spaces of low-dimension. The concept of metric dimension of the so-called Hamming graphs presents a promising way to address this issue.

A subset of vertices in a graph is said to be resolving when the distances to those vertices uniquely characterize every vertex in the graph. The metric dimension of a graph is the size of a smallest resolving subset of vertices. Unfortunately, finding the metric dimension of a general graph is a challenging problem, NP-complete in fact. Recently, however, an efficient algorithm for finding resolving sets in Hamming graphs has been proposed, which suffices to uniquely embed k-mers into a real vector space. Since the dimension of the embedding is the cardinality of the associated resolving set, determining whether or not a node can be removed from a resolving set while keeping it resolving is of great interest. This can be quite challenging for large graphs since only a brute-force approach is known for checking whether a set is a resolving set or not.

In this thesis, we characterize resolvability of Hamming graphs in terms of a linear system over a finite domain: a set of nodes is resolving if and only if the linear system has only a trivial solution over said domain. Since we can represent the domain as the roots of a polynomial system, the apparatus of Gröbner bases comes in handy to determine, algorithmically, whether or not a set of nodes is resolving. As proof of concept, we study the resolvability of Hamming graphs associated with octapeptides i.e. proteins sequences of length eight.

This project has been directed by Prof. Manuel Lladser and co-advised by Richard C. Tillquist.

1 Introduction

In recent years biological sequencing has become cheap and accessible and has brought with it an influx of genomic data. Computational biology researchers are turning to machine learning algorithms to analyze and discover patterns within these massive datasets. Analyzing these symbolic sequences is challenging however, since many machine learning algorithms require numerical representations of data.

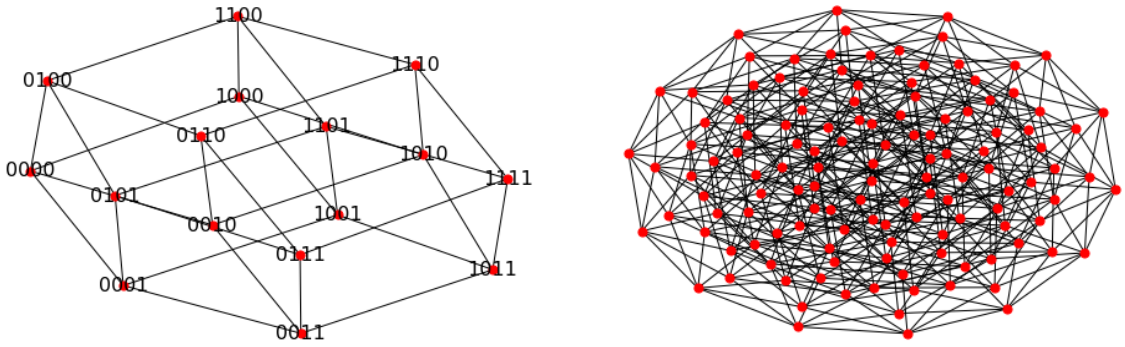
k-mers, strings of length k with letters from a reference alphabet, are frequently used to represent biological sequences. Hamming graphs are an intuitive and simple method of organizing k-mers into a graphical format.

We begin by formally defining the concepts of Hamming graphs and metric dimension. Hamming graphs are a specific family of connected simple graphs which are defined in terms of the Hamming distance.

Definition 1.1. *Given two strings s_1, s_2 of the same length, their **Hamming distance**, denoted $d(s_1, s_2)$, is defined as the number of positions where the strings differ.*

Definition 1.2. *Consider the set S that is the set of all strings of length k formed from a reference alphabet of size a; in particular, $|S| = a^k$. The **Hamming graph** $H_{k,a}$ has vertex set S such that any two vertices $x, y \in S$ are connected by an edge if and only if $d(x, y) = 1$.*

Hamming graphs are connected and regular simple graphs where, for any two vertices, the shortest path between them is equal to their Hamming distance. Although they are simply described, they have very intricate structure and grow in size very quickly as both a and k increase (see Figure 1).



(a) Hamming Graph $H_{4,2}$ (b) Hamming Graph $H_{7,2}$ (node labels omitted)

Figure 1: Examples of Hamming graphs

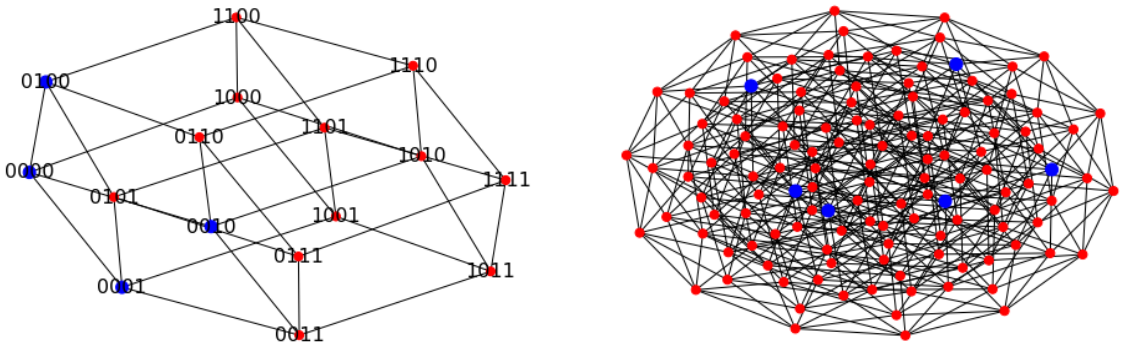
Informally, metric dimension is the minimum number of points required to uniquely identify every point in a metric space, such as a graph, by their distances to those points. For instance, the metric dimension of the Euclidean plane is three, because the plane can be trilaterated using three non-colinear points. In the context of graphs, we have the following definition.

Definition 1.3. *Let $G = (V, E)$ be a connected simple graph with vertex set V and edge set E. The distance between two vertices $d(x, y)$ is the shortest path between them. A vertex $v \in V$* ***resolves*** *two different vertices $x, y \in V$ if $d(v, x) \neq d(v, y)$. A set $R \subseteq V$ is a* ***resolving set*** *of G if every pair of vertices in V is resolved by some vertex in R. R is called a* ***minimal resolving set*** *if there is no other set smaller than R that is also resolving.*

Note that neither resolving sets nor minimal resolving sets are unique in general.

Definition 1.4. *The* ***metric dimension*** *of a graph $G = (V, E)$, denoted $\beta(G)$, is defined as the size of a minimal resolving set.*

For the graphs from Figure 1, $\beta(H_{4,2}) = 4$ with a minimal resolving set $\{0000, 0001, 0010, 0100\}$, and $\beta(H_{7,2}) = 6$ with a minimal resolving set $\{0000000, 0000001, 0000010, 0001100, 0010100, 0100100\}$ (see Figure 2).



(a) $H_{4,2}$ with minimal resolving set in blue

(b) $H_{7,2}$ with minimal resolving set in blue

Figure 2: Examples of minimally resolving sets in Hamming graphs

2 Related Work

Metric dimension was developed for metric spaces in 1951 [1] and was specialized to connected simple graphs by Slater [2] and Haray and Melter [3]. Finding the metric dimension of general connected simple graphs is a known NP-hard problem [4, 5], so most research is limited to specific families of graphs [2, 6] or approximation algorithms for finding nearly minimal resolving sets [7, 8, 9].

The metric dimension of Hamming graphs has been studied in many contexts, from coin-weighing problems [10] to studying optimal play strategies in the Mastermind game [11].

Studies of metric dimension of Hamming graphs have mostly focused on hypercubes (i.e. the case with $a = 2$). The exact metric dimension of these graphs are only known for strings with lengths of up to 8, and have been estimated up to length 17 [6, 8]. Recently, however, Tillquist and Lladser [7] developed a computationally efficient algorithm for finding resolving sets in arbitrary Hamming graphs, which can be used to embed k-mers. This embedding is competitive with other state of the art graph embedding methods such as Node2Vec [12] and MDS [13].

The chief goal of this paper is to answer the question:

Given a resolving set R on $H_{k,a}$, what vertices, if any, can be removed such that R remains resolving?

This could be easily achieved by removing a vertex and checking if R is still resolving. The only known method of checking resolvability on general Hamming graphs is a pairwise comparison of the distance vectors for every node in the graph. This brute-force approach scales as $O(n^2)$ where $n = a^k$, the number of vertices in $H_{k,a}$, making it infeasible on Hamming graphs whose vertex sets grow exponentially. Recently, it has been shown by A. F. Beardon that for hypercubes ($H_{k,2}$) [14], it is possible to construct a constrained linear system for a given set R such that R is resolving if and only if the linear system has only a trivial solution. Unfortunately, the constraints are non-linear which makes showing that no other solutions exist quite challenging. In this paper we present a similar but distinct approach which is applicable to general Hamming graphs rather than only hypercubes. Additionally, through simplification of our system, we are able to reproduce Beardon's system on the hypercubes. Finally, we characterize the constraints of the linear system as the roots to a polynomial solution and create an efficient algorithm for checking if only the trivial solution exists.

3 Resolvability on general Hamming graphs

We begin by defining a useful construction.

Definition 3.1. *The **one-hot encoding** of a vertex v in $H_{k,a}$ is an $(a \times k)$ binary matrix V where $V_{i,j} = 1$ if $i = v[j]$, otherwise $V_{i,j} = 0$.*

Observe that every column in a one-hot encoding has exactly one entry equal to 1 because there is only one letter represented at each position in a string.

Example: The one-hot encoding of 1213 in $H_{4,3}$ is the matrix $\begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$.

We can use one-hot encodings to calculate the Hamming distance between two strings. In what follows, $\mathrm{Tr}(\cdot)$ denotes the trace of a square matrix, and $\bar{A}$ is the logical negation (flip) of the entries in a binary matrix A.

Lemma 3.2. *For any two vertices a and b in $H_{k,a}$, if A and B denote their one-hot encodings, respectively, then the matrix $C = A^T B$ is a binary matrix such that its entries $c_{i,j} = 1$ if and only if $a[i] = b[j]$. In particular, $d(a, b) = k - Tr(A^T B) = Tr(A^T \bar{B})$.*

Informally, this theorem states that $\mathrm{Tr}(A^T B)$ is counting the number of positions where a and b are the same. Further, logically negating A results in counting the number of positions where a and b are not the same. A rigorous proof follows.

Proof: Let A_i and B_i be the i-th columns of A and B, respectively, and note that $A_i = B_j$ if and only if $a[i] = b[j]$. If $A_i = B_j$ then $A_i^T B_j = 1$ since there is exactly one 1 in both A_i and B_j with the rest of the entries being 0. If $A_i \neq B_j$ then $A_i^T B_j = 0$. Therefore, $\mathrm{Tr}(A^T B) = \sum_{i=1}^{k} A_i^T B_i$ counts the number of positions where $a = b$, i.e. $\mathrm{Tr}(A^T B) = k - d(a, b)$. Similarly, if we take $\bar{B}$ to be the logical negation of B then $\mathrm{Tr}(A^T \bar{B})$ counts the number of positions where $a \neq b$, which is by definition the Hamming distance between a and b. □

Since one-hot encodings are special types of real matrices, we note the following obvious result.

Lemma 3.3. *Let vec(A) denote the column-wise vectorization of a matrix A, i.e. the vector obtained by appending the columns of A to form a new column vector. For any two real valued matrices A and B of the same dimension, Tr$(A^T B)$ = $\langle$vec(A), vec$(B)\rangle$.*

Combining Lemmas 3.2-3.3, we obtain the following general characterization of resolvability in arbitrary Hamming graphs.

Theorem 3.4. *Let $R = \{v_1, \ldots, v_n\}$ be a subset of vertices in $H_{k,a}$ of cardinality n. If for each $1 \leq i \leq n$, V_i denotes the (column) vectorized one-hot encoding of v_i, and we define the matrix*

$$A := \begin{pmatrix} V_1^T \\ \vdots \\ V_n^T \end{pmatrix}$$

then R is resolving if and only if there is no non-trivial solution to the linear system $Az = 0$ satisfying the following constraints: if z is decomposed into k non-overlapping blocks of length a as follows $z = \big((z_1, \ldots, z_a), (z_{a+1}, \ldots, z_{2a}), ..., (z_{(k-1)a+1}, \ldots, z_{ka})\big)^T$ then each block is identically zero, or it has exactly one 1 and one (-1) entry and all other entries are 0.

Proof: To prove the theorem, consider first two distinct vertices x and y in $H_{k,a}$, with vectorized one-hot encodings X and Y, respectively. Recall that v_i resolves x and y if and only if $d(v_i, x) \neq d(v_i, y)$, i.e. $\langle V_i, X\rangle \neq \langle V_i, Y\rangle$. Therefore, $\langle V_i, X - Y\rangle = 0$ if and only if v_i does NOT resolve x, y.

For R to be a resolving set of $H_{k,a}$, every pair of vertices x, y must be resolved by some vertex in R. This means R is NOT resolving if and only if there exists some distinct vertices x, y such that $A(X - Y) = 0$.

Finally, since X, Y are vectorized one-hot encodings, each block corresponds to a column in the one-hot encoding. There can only be one 1 in each block for X and Y, therefore in $z := (X - Y)$, each block can have either all 0 entries or can have a 1 from X, a (-1) from $(-Y)$ and the rest of the entries are all 0. This shows the theorem. □

Example: In $H_{3,2}$, consider the set of vertices $R = \{100, 101, 001\}$. The one-hot encodings for R are:

$$100 \leftrightarrow \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 0 \end{bmatrix}; \quad 101 \leftrightarrow \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix}; \quad 001 \leftrightarrow \begin{bmatrix} 1 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

The matrix in Theorem 3.4 is therefore given by

$$A := \begin{bmatrix} 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 1 \end{bmatrix}.$$

By Theorem 3.4, R resolves $H_{3,2}$ if and only $Az = 0$ has no non-trivial solution z which satisfies the conditions from the theorem when writing $z = \big((z_1, z_2), (z_3, z_4), (z_5, z_6)\big)$. To determine whether this is the case, note the reduced row echelon form [15] of the matrix A:

$$\text{rref}(A) = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & -1 \end{bmatrix}.$$

In particular, we can express the pivots z_1, z_2, and z_5 in terms of the free variables z_3, z_4, and z_6 as: $z_1 = -(z_3 + z_6)$, $z_2 = -(z_3 + z_6)$, and $z_5 = z_6$. Exploring now the finite number of possibilities for the free variables, we are able to determine whether there is a non-trivial solution z that satisfies the constraints.

Starting with the variable z_6 we first recognize that z_5, z_6 are in the same block and $z_5 = z_6$. This means $z_5 = z_6 = 0$ since if $z_6 \neq 0$, the block constraints would require $z_5 = -z_6$.

Then we check z_3, z_4 since they are in the same block and both free variables. Notice that z_4 is actually a useless variable since no pivots depend on it. This means we only need to consider the three possibilities for z_3:

- If $z_3 = 0$ then $z_1 = 0$, $z_2 = 0$, and $z_4 = 0$ since it is in the same block as z_3. This is the trivial solution $z = 0$.
- If $z_3 = 1$ then $z_1 = -1$ and $z_2 = -1$ which is not allowed since if $z_1 = -1$, the block constraints require $z_2 = 1$.
- If $z_3 = -1$ then $z_1 = 1$, and $z_2 = 1$ which is not allowed since if $z_1 = 1$, the block constraints require $z_2 = -1$.

As a result, there is no non-trivial solution to $Az = 0$ which satisfies the constraints, so R resolves $H_{3,2}$.

This example was quite simple since no pivots depended on z_4, but these systems can be considerably more complex. In general, if the reduced row echelon of A has j free variables, then there could be up to 3^j possible solutions z to the system $Az = 0$, each of which would have to be checked for the theorem constraints. This exhaustive search could be very time consuming and difficult. Handling these constraints efficiently and algorithmically is the motivation for the next section.

4 Algorithmically Handling the Constraints

Solving the linear system $Az = 0$ from Theorem 3.4 is difficult because, while the system is linear, the constraints on the vector z are not. Nevertheless, the possible $z = \big((z_1, \ldots, z_a), \ldots, (z_{(k-1)a+1}, \ldots, z_{ka})\big)$ vectors can be characterized as the solutions to $p(z) = 0$, with $p \in P$, where:

$$P := \left\{ \begin{array}{c} \begin{array}{c} z_1(z_1-1)(z_1+1) \\ z_2(z_2-1)(z_2+1) \\ \vdots \\ z_{ka}(z_{ka}-1)(z_{ka}+1) \end{array} \\ \hline \begin{array}{c} \sum_{i=1}^{a} z_i \\ \sum_{i=a+1}^{2a} z_i \\ \vdots \\ \sum_{i=(k-1)a+1}^{ka} z_i \end{array} \\ \hline \begin{array}{c} (\sum_{i=1}^{a} z_i^2)(2 - \sum_{i=1}^{a} z_i^2) \\ (\sum_{i=a+1}^{2a} z_i^2)(2 - \sum_{i=a+1}^{2a} z_i^2) \\ \vdots \\ (\sum_{i=(k-1)a+1}^{ka} z_i^2)(2 - \sum_{i=(k-1)a+1}^{ka} z_i^2) \end{array} \end{array} \right. \tag{1}$$

Indeed, the set of polynomials of the form $z_i(z_i - 1)(z_i + 1) = 0$ enforce that $z_i \in \{-1, 0, 1\}$. The second set of polynomials of the form $\sum_{i=1}^{a} z_i = 0$ enforce that there is a (-1) for every 1 in each block of z. Finally, the third set of polynomials of the form $(\sum_{i=1}^{a} z_i^2)(2 - \sum_{i=1}^{a} z_i^2)$ ensure that there are exactly two non-zero terms or no non-zero terms in each block of z. These three sets of polynomials can now be combined to ensure all of the constraints on z from Theorem 3.4.

Therefore, if we define $\{P = 0\}$ to be the set of solutions z of the polynomial system, we can characterize the solution space in Theorem 3.4 as $ker(A) \cap S$. To find $ker(A) \cap S$, we use Gaussian elimination to find rref(A) and let L be the linear equations associated with the system $rref(A)z = 0$. These linear equations are polynomials so we append them to the polynomial system P since the solutions to $P \cup L$ are $ker(A) \cap \{P = 0\}$. We can show R is resolving by using algebraic geometry techniques like Gröbner bases to check for solutions to $P \cup F$.

Example. The polynomial system $P = 0$ associated with the Hamming graph $H_{3,2}$ is:

$$\left\{ \begin{array}{rl} \begin{array}{r} z_1(z_1-1)(z_1+1) \\ \vdots \\ z_6(z_6-1)(z_6+1) \end{array} & \begin{array}{l} = 0 \\ \\ = 0 \end{array} \\ \hline \begin{array}{r} z_1 + z_2 \\ z_3 + z_4 \\ z_5 + z_6 \end{array} & \begin{array}{l} = 0 \\ = 0 \\ = 0 \end{array} \\ \hline \begin{array}{r} (z_1^2 + z_2^2)(2 - z_1^2 - z_2^2) \\ (z_3^2 + z_4^2)(2 - z_3^2 - z_4^2) \\ (z_5^2 + z_6^2)(2 - z_5^2 - z_6^2) \end{array} & \begin{array}{l} = 0 \\ = 0 \\ = 0 \end{array} \end{array} \right.$$

4.1 Gröbner Bases

Gröbner bases are a fundamental part of algebraic geometry and are very useful to characterize solutions of polynomial systems, such as the one we encountered in the previous section. Gröbner bases serve a purpose for polynomial ideals similar to the one orthogonal bases serve to vector spaces.

In what follows, $z = (z_1, \ldots, z_k)$ is a k-dimensional variable.

Definition 4.1. *For a set of polynomials in the variable z, $P = \{p_1(z), \ldots, p_n(z)\}$, the associated polynomial ideal, denoted $I(P)$, is defined as the set of all polynomials of the form $f_1p_1 + \ldots + f_np_n$, where $f_i(z)$ are arbitrary polynomials.*

Consider a homogeneous polynomial system $p_i(z) = 0$, for $i = 1, 2, \ldots, n$. Define $P := \{p_1, \ldots, p_n\}$. It follows that z is a solution of the polynomial system if and only if $g(z) = 0$, for all $g \in I(P)$, i.e. z is a root of each polynomial in the ideal associated with the polynomials in the system. It is often more convenient to characterize the roots of the latter than those of the original system. This is done by using a Gröbner basis of $I(P)$. Before we can define Gröbner bases, we introduce some important concepts from [16, 17].

Definition 4.2. *A **monomial** in z is defined as any product of the form $z_1^{a_1} \cdots z_k^{a_k}$, with non-negative integers $a_1, \ldots, a_k$. This product is often written z^a where $a = (a_1, \ldots, a_k)$.*

Note that all polynomials in z are linear combinations of monomials.

Definition 4.3. *A **monomial ordering** $>$ is a total ordering of the monomials such that:*

1. *If $z^a > z^b$ then $z^a z^c > z^b z^c$ for any monomial z^c; and*
2. *$z^a > 1$ if $z^a \neq 1$.*

Some common monomial orderings are:

- **Lexicographic Ordering:** $z^a > z^b$ if there is an index i such that $a_j = b_j$ for all $1 \leq j < i$ but $a_i > b_i$.
- **Graded Lexicographic Ordering:** $z^a > z^b$ if either:
  1. $\sum_{i=1}^{k} a_i > \sum_{i=1}^{k} b_i$; or
  2. $\sum_{i=1}^{k} a_i = \sum_{i=1}^{k} b_i$, and z^a is greater than z^b under the lexicographic ordering.

The lexicographic ordering can also be reversed giving the **reversed lexicographic** and the **graded reverse lexicographic** orderings.

Definition 4.4. *Given a polynomial p and monomial ordering $>$; we can write $p = c_1m_1 + \ldots c_nm_n$ such that c_i are constants and m_i are monomials in descending order, i.e. $m_1 > \ldots > m_n$. With this, we define:*

- ***Leading monomial**: $LM(p) := m_1$*
- ***Leading coefficient**: $LC(p) := c_1$*
- ***Leading term**: $LT(p) := c_1m_1$*

Definition 4.5. *For a set of polynomials $P = \{p_1, \ldots, p_n\}$ and a monomial ordering $>$, every polynomial f can be written through polynomial long division as: $f = q_1p_1 + \ldots + q_np_n + r$, where all q_i, r are polynomials. Writing f in this form is called **reducing** f by P. The q_i are called **quotients**, while r is called the **remainder** or **reduction** of f by P.*

In what follows, we write $f \xrightarrow{P} r$ to mean that r is the reduction of f by P.

The steps of multivariate polynomial long division are performed as follows. First, set $r = 0$ and $g = f$ and look for a $LM(p_i)$ that divides the $LM(g)$. If multiple p_i exist we choose the one with the lowest value of i. With p_i set $g = g - LT(g)p_i/LT(p_i)$ so that the $LT(g)$ and the $LT(p_i)$ cancel out. If no such p_i exists then set $g = g - LT(g)$ and $r = r + LT(g)$. Continue this process until $g = 0$. This will produce a remainder r where no monomial of r is divisible by any $LM(p_i)$.

In general, reductions are not unique because by convention we choose the p_i with the lowest value of i. Re-ordering the polynomials in P may produce a different remainder for the same polynomial f. This non-uniqueness is the primary motivation for Gröbner bases. Indeed, a Gröbner basis will be a set of polynomials G such that for any polynomial f, the reduction of f by G is unique.

Definition 4.6. *Given two monomials z^a and z^b, their **least common multiple**, denoted $LCM(z^a, z^b)$, is the monomial z^c with $c = (\max\{a_1, b_1\}, \ldots, \max\{a_k, b_k\})$.*

Definition 4.7. *Given two polynomials p_1, p_2 where $LCM(LM(p_1), LM(p_2)) = z^c$, their **S-polynomial** is defined as*

$$Spoly(p_1, p_2) := \frac{z^c}{LT(p_1)}p_1 - \frac{z^c}{LT(p_2)}p_2.$$

The S-polynomial is an important object for constructing Gröbner bases. Notice that the S-polynomial is defined so that $LT(p_1)$ and $LT(p_2)$ cancel as they do in polynomial long division. Also note that $Spoly(p_1, p_2) \in I(\{p_1, p_2\})$.

With these elements in hand, Gröbner bases can now be defined.

Definition 4.8. *(Buchberger's Criterion.) A set of polynomials $G = \{g_1, \ldots, g_n\}$ is a **Gröbner basis** for a polynomial ideal I if and only if $I(G) = I$ and for all pairs $g_i, g_j \in G$:*

$$Spoly(g_1, g_2) \xrightarrow{G} 0.$$

*A Gröbner basis G is called **reduced** if for all distinct $g_i, g_j \in G$, $LC(g_i) = 1$ and no monomial of g_j is divisible by $LT(g_i)$.*

In general, Gröbner bases are not unique whereas reduced Gröbner bases are unique under a given ordering. Gröbner bases also have some important properties with regards to reductions; in particular, $f \xrightarrow{G} 0$ if and only if $f \in I(G)$, otherwise $f \xrightarrow{G} r$ for some $r \notin I(G)$.

4.2 Computing Gröbner Bases

The computation of Gröbner bases is an active field of research with many different classes of approaches. There are matrix reduction based algorithms, such as Faugére's F4 algorithm [18], as well as so-called "signature" based algorithms, like Faugére's F5 algorithm and its variants F5C and F5B [19, 20, 21]. The simplest and most well-studied approach is Buchberger's algorithm [16, 17].

While significantly slower than other methods, Buchberger's algorithm (Algorithm 1) is the most easily understood and adequately demonstrates the steps involved in computing general Gröbner bases. Just like Buchberger's criterion (Definition 4.8), the algorithm is based upon S-polynomial computations. The algorithm takes in two inputs, a set of polynomials P and a monomial ordering $>$.

Algorithm 1 has two bottlenecks: computing and reducing S-polynomials by the basis. Both steps rely on multivariable polynomial long division which can become very computationally intensive for high degree polynomials as well as for polynomials formed over a large variable set. Also, S-polynomials which eventually reduce to 0 are costly to compute but contain no new information for generating the Gröbner basis. Avoiding the computation and reduction of S-polynomials which will reduce to 0 is key to efficiently computing Gröbner bases.

Algorithm 1 Buchberger's algorithm for computing a Gröbner basis

function BUCHBERGER(P, $>$)
  $G = P$
  $SP = \{Spoly(g_i, g_j) \mid \forall i < j, g_i, g_j \in G\}$
  while SP not empty **do**
    $S = SP$.pop()
    $r = S \xrightarrow{G} r$
    if $r \neq 0$ **then**
      Add $Spoly(r, g_i)$ to SP for each $g_i \in G$
      $G = G \cup \{r\}$
  return G
end function

Buchberger's algorithm may find a non-reduced Gröbner basis. A further basis reduction algorithm (Algorithm 2) is then applied to retrieve the unique reduced basis for the specified ordering. Algorithm 2 starts by dividing out the lead coefficient of every polynomial of G. Then it reduces every polynomial by the Gröbner basis with the polynomial removed. Since G is a Gröbner basis, r is unique and, as a result, so is the reduced Gröbner basis.

Algorithm 2 Algorithm for reducing a Gröbner basis

function REDUCEGRÖBNER(G, $>$)
  For each $g_i \in G$, $g_i = \frac{g_i}{LC(g_i)}$
  for $g_i \in G$ **do**:
    $H = G \setminus \{g_i\}$
    $r = g_i \xrightarrow{H} r$
    if $r \neq 0$ **then**:
      $g_i = r$
    else
      $G = G \setminus \{g_i\}$
  return G
end function

4.3 Resolvability of Hamming Graphs in Terms of Gröbner Bases

Gröbner bases have many very useful properties for understanding polynomial ideals. In this work, we focus on the following important theorem.

Theorem 4.9. *(Hilbert's Weak Nullstellensatz [16, 22].) Let $P = \{p_1, \ldots, p_k\}$ be a set of polynomials. Then exactly one of the following statements is true:*

1. *$P = 0$ has a solution.*
2. *$1 \in I(P)$, i.e. there exist polynomials $f_1, \ldots, f_k$ such that $f_1 p_1 + \ldots + f_k p_k = 1$.*

Equivalently, $\{P = 0\} = \varnothing$ if and only if $\{1\}$ is the reduced Gröbner basis of $I(P)$.

Using Hilbert's Weak Nullstellensatz, we can check for the existence of solutions to any homogeneous polynomial system; in particular, Theorem 4.9 allows us to determine whether or not the linear system L together with the constraints represented as solutions to a polynomial system P given by Theorem 3.4 has solutions. This corresponds to establishing whether or not a given set of nodes in $H_{k,a}$ is resolving. There is a slight complication to this however. $z = 0$ is always a root of this linear/polynomial system. To get around this, recall that each block of $z = \big((z_1, \ldots, z_a), \ldots, (z_{(k-1)a+1}, \ldots, z_{ka})\big)$ either vanishes, or contains exactly one 1 and one (-1) entry while the remaining entries vanish. As a result, if $z \neq 0$ then $\sum_{i=1}^{ka} z_i^2 = 2i$ for some $1 \leq i \leq k$. This motivates to consider the polynomials $f_i = \sum_{i=1}^{ka} z_i^2 - 2i$ with $i = 1, \ldots, k$.

If we now define $P_i = P \cup f_i$ for $i = 1, \ldots, k$, then each P_i will have only non-trivial solutions—if any at all. This means that computing the reduced Gröbner bases G_i of $Az \cup P_i$ and applying Hilbert's Weak Nullstellensatz to each G_i individually can be used to check for the existence of non-trivial solutions. If any $G_i \neq \{1\}$, then there is a non-trivial solution to $Az \cup P = 0$ and R is not resolving.

Example. Recall from our previous example that $R = \{100, 101, 001\}$ resolves $H_{3,2}$. Moreover, the reduced row echelon form of the matrix A given by Theorem 3.4 is:

$$\text{rref}(A) = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & -1 \end{bmatrix}.$$

and the polynomials associated with the constraints on z in Theorem 3.4 is

$$P := \left\{ \begin{array}{c} z_1(z_1-1)(z_1+1) \\ \vdots \\ z_6(z_6-1)(z_6+1) \\ \hline z_1 + z_2 \\ z_3 + z_4 \\ z_5 + z_6 \\ \hline (z_1^2+z_2^2)(2-z_1^2-z_2^2) \\ (z_3^2+z_4^2)(2-z_3^2-z_4^2) \\ (z_5^2+z_6^2)(2-z_5^2-z_6^2) \end{array} \right\}.$$

Finally, we must consider the polynomials:

$$f_i = \sum_{j=1}^{6} z_j^2 - 2i,$$

for $i = 1, 2, 3$. For each of these i, it turns out the Gröbner bases of $Az \cup P \cup f_i$ is $\{1\}$; in particular, the set R is resolving. This result is not surprising since we showed previously that no non-trivial solutions exist for this system.

4.4 Reduced Gröbner Basis of P

We can improve the efficiency of Gröbner basis computations by examining the highly-structured set of polynomials P in equation (1). In fact, an in depth look at the structure of P provides a powerful intuition about how the complexity of the basis changes with the parameters k and a for the underlying Hamming graph $H_{k,a}$.

We can partition P into disjoint sets (which we call blocks) as follows. For each $i = 0, \ldots, (k-1)$, define:

$$P_i := \begin{cases} z_{ia+1}(z_{ia+1}-1)(z_{ia+1}+1) \\ z_{ia+2}(z_{ia+2}-1)(z_{ia+2}+1) \\ \vdots \\ z_{(i+1)a}(z_{(i+1)a}-1)(z_{(i+1)a}+1) \\ \dfrac{\sum_{j=ia+1}^{(i+1)a} z_j}{(\sum_{j=ia+1}^{(i+1)a} z_j^2)(2 - \sum_{j=ia+1}^{(i+1)a} z_j^2)} \end{cases} \tag{2}$$

Note that $P = (P_0 \cup \ldots \cup P_{k-1})$. Furthermore, each P_i is a set of polynomials in the variables $z_{ia+1}, \ldots, z_{(i+1)a}$, i.e. the i-th block of $z = \big((z_1, \ldots, z_a), \ldots, (z_{(k-1)a+1}, \ldots, z_{ka})\big)$. In particular, no variable and hence no polynomial is shared between the blocks of P. Our next result reveals the importance of this partition.

In what follows, all polynomial ideals are with respect to z, not only variables in certain blocks of z.

Lemma 4.10 ([16])**.** *Consider two polynomial sets Q_1 and Q_2 with disjoint variables x and y, and reduced Gröbner bases $G_1 \neq \{1\}$ and $G_2 \neq \{1\}$, respectively. Then $(G_1 \cup G_2)$ is a reduced Gröbner basis of $(Q_1 \cup Q_2)$.*

Proof: The core part of this lemma is the disjoint variable vectors x and y. Define $z := (x, y)$, $Q := (Q_1 \cup Q_2)$, and $G := (G_1 \cup G_2)$.

We first show that G is a Gröbner bases for Q. To start, note that $I(G) = I(Q)$. Indeed, since the ideal of a union is the sum of the ideals, and $I(Q_i) = I(G_i)$ for $i = 1, 2$ because G_i is a Gröbner bases for Q_i, it follows that $I(Q) = I(Q_1) + I(Q_2) = I(G_1) + I(G_2) = I(G)$.

Next, we show that if $p, q \in I(G)$ then $Spoly(p, q) \xrightarrow{G} 0$. This is certainly the case if $p, q \in G_1$ or $p, q \in G_2$ because G_1 and G_2 are Gröbner bases. Thus, without loss of generality, we may assume that $p \in G_1$ and

$q \in G_2$. In particular, $p = p(x)$ and $q = q(y)$, hence $LCM(LM(p), LM(q)) = LM(p)LM(q)$. As a result:

$$\begin{aligned} Spoly(p,q) &= \frac{LM(p)LM(q)}{LT(p)}p - \frac{LM(p)LM(q)}{LT(q)}q \\ &= \frac{LM(q)}{LC(p)}p - \frac{LM(p)}{LC(q)}q \end{aligned}$$

Then we write our polynomials as:

$$\begin{aligned} p &= LT(p) + f \\ q &= LT(q) + g \end{aligned}$$

We can then write $Spoly(p,q)$ as:

$$\begin{aligned} Spoly(p,q) &= \frac{LM(q)}{LC(p)}p - \frac{LM(p)}{LC(q)}q \\ &= \frac{LM(q)f}{LC(p)} - \frac{LM(p)g}{LC(q)} \\ &= \frac{LT(q)f}{LC(q)LC(p)} - \frac{LT(p)g}{LC(q)LC(p)} \\ &= \frac{(q-g)f - (p-f)g}{LC(q)LC(p)} \\ &= \frac{qf - pg}{LC(q)LC(p)} \end{aligned}$$

This formulation makes the reduction clear. qf is trivially reducible to 0 by q and pg is trivially reducible to 0 by p, therefore $Spoly(p,q)$ reduces to 0.

It remains to show that G is also reduced. For this note that $LC(g) = 1$ for each $g \in G$ because G is the union of reduced Gröbner bases.

Finally, we show that if $p, q \in G$ are distinct then no monomial of q is divisible by the $LT(p)$. This is clearly the case if $p, q \in G_1$ or $p, q \in G_2$ because G_1 and G_2 are both reduced Gröbner bases. To complete the proof, without loss of generality assume that $p \in G_1$ and $q \in G_2$. Note that $p, q \neq 1$, otherwise $G_1 = \{1\}$ or $G_2 = \{1\}$ by Hilbert's Weak Nullstellensatz (Theorem 4.9). Then, since $p = p(x)$ and $q = q(y)$, the leading term in p cannot divide any monomial of q. □

The last lemma easily extends by induction to any finite union of polynomial sets which do not share variables. This leads immediately to the following result.

In what remains of this section, G_i denotes the reduced Gröbner bases of P_i in equation (2).

Corollary 4.10.1. *$G = (G_0 \cup \ldots \cup G_{k-1})$ is the reduced Gröbner bases of P in equation (1).*

Finally, note that each P_i is—up to the change of variables $z_j \longrightarrow z_{ia+j}$ for $j = 1, \ldots, a$—equal to P_0. In particular, the reduced Gröbner bases G_i of P_i may be obtained simply by renaming the variables in G_0. This drastically reduces the computation time of Gröbner bases to address the resolvability of vertices in Hamming graphs.

We can further recognize that for $H_{k,a}$, P_0 only changes with a. A change in k corresponds to a change in the number of blocks whereas a change in a corresponds to a change in the structure of the blocks themselves.

This means we only need to study how P_0 and G_0 change with a without worrying about different values of k. In this regard, we have the following result.

Lemma 4.11. *The reduced Göbner bases of P_0 with respect to the lexicographic ordering is*

$$G_0 = \begin{cases} \sum_{i=1}^{a} z_i \\ z_2(z_2-1)(z_2+1) \\ \vdots \\ z_a(z_a-1)(z_a+1) \\ \hline z_2 z_3 (z_2+z_3) \\ \vdots \\ z_{a-1} z_a (z_{a-1}+z_a) \\ \hline z_2 z_3 z_4 \\ \vdots \\ z_{a-2} z_{a-1} z_a \end{cases}$$

Proof: G_0 can be shown to be a reduced Gröbner basis for an ideal by checking the 4 properties of reduced Gröbner bases and verifying that $I(G_0) = I(P_0)$. Demonstrating that G_0 satisfies the 4 properties of reduced Gröbner bases is a lengthy and uninformative process so we instead only show $I(G_0) = I(P_0)$.

To prove $I(G_0) = I(P_0)$ we show that the set of solutions to G_0 is the same as for P_0. The first polynomial simply requires that the sum of the variables is 0. The next set of polynomials of the form $z_i(z_i-1)(z_i+1) = 0$ for $i = 2, 3, \ldots, a$ forces $z_i \in \{-1, 0, 1\}$. The second set of polynomials $z_i^2 z_j + z_i z_j^2$ are formed for all pairs z_i, z_j when $2 \leq i < j$. This block enforces that for every pair of variables z_i, z_j, either $z_i = -z_j$ or at least one of the variables is 0. The last set of polynomials $z_i z_j z_k$ are formed over all triplets z_i, z_j, z_k when $2 \leq i < j < k$. This set of polynomials enforce that for every triplet of variables, at least one of them is 0. Since every triplet must contain at least one 0 element, there can be at most only one pair of non-zero elements. The constraints implied by these blocks are therefore identical to the constraints from Theorem 3.4 and $I(G_0) = I(P_0)$. □

Another optimization can be made during the computation of the Gröbner bases G_i of $G \cup f_i$. This basis computation can be improved by first performing the reduction $f_i \xrightarrow{G} r_i$ to speed up the subsequent S-polynomial computations and reductions. The polynomials f_i are all very similar. In particular, if $f = \sum_{j=1}^{ak} z_j^2$ then $f_i = f - 2i$ for $i = 1, 2, \ldots, k$. Using the following lemma, we can efficiently perform the reduction $f_i \xrightarrow{G} r_i$ for all f_i with just one reduction $f \xrightarrow{G} r$.

Lemma 4.12. *Consider a reduced Gröbner basis $G \neq \{1\}$ and a polynomial f with no constant monomials. Define $f_i = f - c_i$ with all c_i constants. If $f \xrightarrow{G} r$ then $f_i \xrightarrow{G} r - c_i$.*

Proof: Start by recognizing that $G \neq \{1\}$ implies $1 \notin I(G)$ and therefore $c_i \notin I(G)$ for any constant c_i. This implies that c_i is irreducible by G, i.e. $c_i \xrightarrow{G} c_i$.

If $f \xrightarrow{G} r$, we can write f in terms of its quotients q_j.

$$f = q_1 g_1 + q_2 g_2 + \ldots + q_n g_n + r$$

We also know that $f_i = f - c_i$ so we have:

$$\begin{aligned} f_i &= f - c_i \\ &= q_1 g_1 + q_2 g_2 + \ldots + q_n g_n + r - c_i \end{aligned}$$

Since both c_i and r are irreducible by G, we know that $r - c_i$ is also irreducibly by G. This implies that $f_i = (q_1 g_1 + q_2 g_2 + \ldots q_n g_n + r - c_i)$ is a valid reduction of f_i by G. This reduction is unique since G is a Gröbner basis and so $f_i \xrightarrow{G} (r - c_i)$. □

Using this lemma we arrive at the following result:

Corollary 4.12.1. *Let G be the reduced Gröbner basis of P from equation (1) and $f = \sum_{j=1}^{ak} z_j^2$ with $f \xrightarrow{G} r$. If $f_i := (f - 2i)$ for $i = 1, 2, \ldots, k$ then $f_i \xrightarrow{G} (r - 2i)$.*

5 Simplification on the Hypercubes

In this section we specialize the results from the previous sections to hypercubes, i.e. Hamming graphs with an alphabet of size $a = 2$. In this case, our findings and complexity simplify considerably to characterize resolving sets in $H_{k,2}$. We note that this simplification reproduces the result in [14].

Our next result is a simplified version of Theorem 3.4.

Corollary 5.0.1. *Let $R = \{v_1, \ldots, v_n\}$ be a set of nodes in $H_{k,2}$, and $\bar{v_i}$ denote the logical negation (flip) of v_i. Consider the matrix of dimensions $(n \times k)$ defined as:*

$$B := \begin{pmatrix} v_1 - \bar{v_1} \\ \vdots \\ v_n - \bar{v_n} \end{pmatrix}.$$

Then, R resolves $H_{k,2}$ if and only if the equation $Bz = 0$, with $z \in \{-1, 0, 1\}^k$, has only a trivial solution.

Proof: Recall the polynomial system $P = 0$ from Section 2.1, specialized to the hypercube of dimension k:

$$\begin{cases} z_1(z_1 - 1)(z_1 + 1) &= 0 \\ z_2(z_2 - 1)(z_2 + 1) &= 0 \\ \vdots & \\ z_{2k-1}(z_{2k-1} - 1)(z_{2k-1} + 1) &= 0 \\ z_{2k}(z_{2k} - 1)(z_{2k} + 1) &= 0 \\ \hline z_1 + z_2 &= 0 \\ \vdots & \\ z_{2k-1} + z_{2k} &= 0 \\ \hline (z_1^2 + z_2^2)(2 - (z_1^2 + z_2^2)) &= 0 \\ \vdots & \\ (z_{2k-1}^2 + z_{2k}^2)(2 - (z_{2k-1}^2 + z_{2k}^2)) &= 0 \end{cases}$$

Notice that $z_1 + z_2 = 0$ implies that $z_1 = -z_2$ and $z_1^2 = z_2^2$; in particular, $z_1(z_1 - 1)(z_1 + 1) = -z_2(z_2 - 1)(z_2 + 1)$. So the equation $z_1(z_1 - 1)(z_1 + 1) = 0$ is redundant and can be removed from the above polynomial system. Similarly, we find that $(z_1^2 + z_2^2)(2 - (z_1^2 + z_2^2)) = -4z_2 \cdot z_2(z_2 - 1)(z_2 + 1)$, in particular, the equation

$(z_1^2 + z_2^2)(2 - (z_1^2 + z_2^2)) = 0$ is also redundant and can be removed from the system too. The same reasoning can be followed starting with any of the equations in the middle block of the system. As a result, the polynomial system is equivalent to:

$$\begin{cases} z_2(z_2-1)(z_2+1) &= 0 \\ z_1 + z_2 &= 0 \\ \vdots & \\ z_{2k}(z_{2k}-1)(z_{2k}+1) &= 0 \\ z_{2k-1} + z_{2k} &= 0 \end{cases}$$

But notice that all equations of the form $z_{2i-1} + z_{2i} = 0$ with $i = 1, \ldots, k$ are linear constraints, so they can be pulled out of the polynomial system and instead encoded within the linear system $Az = 0$ in Theorem 3.4. This gives the simplified polynomial system:

$$\begin{cases} z_2(z_2-1)(z_2+1) &= 0 \\ \vdots & \\ z_{2k}(z_{2k}-1)(z_{2k}+1) &= 0 \end{cases}$$

Note that this system is equivalent to imposing that $(z_2, \ldots, z_{2k}) \in \{-1, 0, 1\}^k$.

Finally, we can encode the linear constraints $z_{2i-1}+z_{2i} = 0$ with $i = 1, \ldots, k$ by setting $z = (-z_2, z_2, \ldots, -z_{2k}, z_{2k})^T$. Now consider the matrix A from Theorem 3.4 whose i-th column is denoted A_i:

$$A := \begin{pmatrix} | & | & & | & | \\ A_1 & A_2 & \ldots & A_{2k-1} & A_{2k} \\ | & | & & | & | \end{pmatrix}.$$

Then $Az = z_2(A_2 - A_1) + \ldots + z_{2k}(A_{2k} - A_{2k-1})$, hence the linear system $Az = 0$ can be represented by the alternative linear system:

$$\begin{bmatrix} | & & | \\ (A_2 - A_1) & \ldots & (A_{2k} - A_{2k-1}) \\ | & & | \end{bmatrix} \begin{bmatrix} z_2 \\ \vdots \\ z_{2k} \end{bmatrix} = 0.$$

Recall however that from Theorem 3.4, the rows of A are vectorized one-hot encodings; in particular, every pair of columns A_{2i}, A_{2i-1} corresponds to the i-th positions of the strings in the resolving set. This implies that $A_{2i} = v_i$ and $A_{2i-1} = \bar{v}_i$, and the corollary follows. □

6 Applications of Results

We are now ready to tackle the motivating question of this work:

Given a resolving set R on $H_{k,a}$, what vertices, if any, can be removed such that R remains resolving?

We can determine whether a vertex $r \in R$ can be removed by checking if the set $R \setminus r$ is resolving. If it is, then r is removed, and this process is repeated until no nodes can be removed from R. Algorithm 3 shows an improved version of this idea. Since $H_{k,a}$ is fixed, the polynomial set P and the polynomials f_i are also fixed. Therefore, it is more efficient to pre-compute the reduced Gröbner bases G_i of $P \cup f_i$ before including the equations associated with the linear system $Az = 0$. Additionally, each row of A corresponds to a node in R. In particular, if $rank(A) < |R|$ then there are some linearly dependent rows of A. These linearly dependent rows correspond to vertices that can be removed from R without changing the system. Since the system does not change upon their removal, these can be immediately removed beforehand with no checks of resolvability.

Algorithm 3 Remove Vertices from Resolving Set

```
function reduceResolvingSet(R,k,a)
    Construct polynomial set P and polynomials f_i for H_{k,a}
    Construct matrix A for set R
    if rank(A) < |R| then
        remove linearly dependent rows of A and corresponding vertices in R
    Initialize shuffledR to be a randomly shuffled R
    G_p = precomputed reduced Gröbner basis of P
    for i = 1,2,...,k do
        G_i = reduced incremental Gröbner basis of G_P ∪ f_i
    for r in shuffledR do
        remove r from R
        remove row corresponding to r from A
        L = linear equations associated with rref(A)z
        for i = 1,2,...,k do
            G = G_i
            G = reduced incremental Gröbner basis of G ∪ L
            if G != {1} then
                Add removed row back into A and r into R
                Break
    Return R
end function
```

Unfortunately, the top-down approach of Algorithm 3 may be very time consuming because removing nodes amounts to removing polynomials from the Gröbner bases G_i. Removing polynomials from a Gröbner basis is significantly more challenging than adding a polynomial to it. With this insight, it is more efficient to instead consider the bottom-up approach of randomly adding nodes from R to an initially empty set R', until R' becomes resolving. The pseudocode for this approach can be found in Algorithm 4.

The insights for reducing known resolving sets can be used to generate resolving sets from scratch as well. Algorithm 5 implements this approach. The algorithm incrementally adds random nodes to an initially empty set R that increase the rank of the associated matrix A. This process is repeated until R is resolving. The algorithm guarantees construction of a resolving set since A is non-singular (i.e., has only the trivial solution) when $rank(A) = ak$. Since $rank(A) = |R|$, once ak vertices have been added into R, it is trivially resolving on $H_{k,a}$. We note that the resolving sets produced by this algorithm are not necessarily minimal.

Algorithm 4 Generative Approach for Reducing Resolving Sets

function REDUCERESOLVINGSET-GEN(R,k,a)
  Construct polynomial system P and polynomials f_i for $H_{k,a}$
  Construct matrix A for set R
  if rank(A) < |R| **then**
    remove linearly dependent rows of A and corresponding vertices in R
  Initialize shuffledR to be a randomly shuffled R
  G_p = precomputed reduced Gröbner basis of P
  for i = 1,2,. . . ,k **do**
    G_i = reduced incremental Gröbner basis of $G_P \cup f_i$
  Initialize R' = $\varnothing$
  for r in shuffledR **do**
    add r to R'
    Set L = linear equation associated to r
    for i = 1,2,. . . ,k **do**
      $G_i = G_i \cup L$
    if All G_i == $\{1\}$ **then**
      Return R'

Algorithm 5 Generate Resolving Sets

function GENRESOLVINGSET(k,a)
  Construct polynomial system P and polynomials f_i for $H_{k,a}$
  Precompute reduced Gröbner bases G_i for $P \cup f_i$
  Initialize V := vertex set of $H_{k,a}$
  R = some randomly selected $r \in V$
  while R is not resolving **do**
    Randomly select some $r \in V$ that has not been selected before
    Construct matrix A for $R \cup r$
    if rank(A) == |R|+1 **then**
      $R = R \cup r$
      for i = 1,2,. . . ,k **do**
        G = reduced incremental Gröbner basis of $G_i \cup \text{rref}(A)z$
        if G != $\{1\}$ **then**
          Break
        Set R as resolving
  Return R
end function=0

7 Preliminary run-time Analysis

We implemented the proposed Gröbner basis approach for checking resolvability on $H_{k,a}$ using SymPy [23], a Python computer algebra package. In this section we present run-time results of this implementation.

To test the theory, we have generated $4,359$ example sets for Hamming graphs with parameters $k = 1, \ldots, 10$ and $a = 2, \ldots, 5$. The graphs were restricted to those where $ak \leq 25$ in order to limit the complexity for these initial tests. The example sets consist of around 200 sets per graph, at least half of which are resolving on their associated Hamming graph.

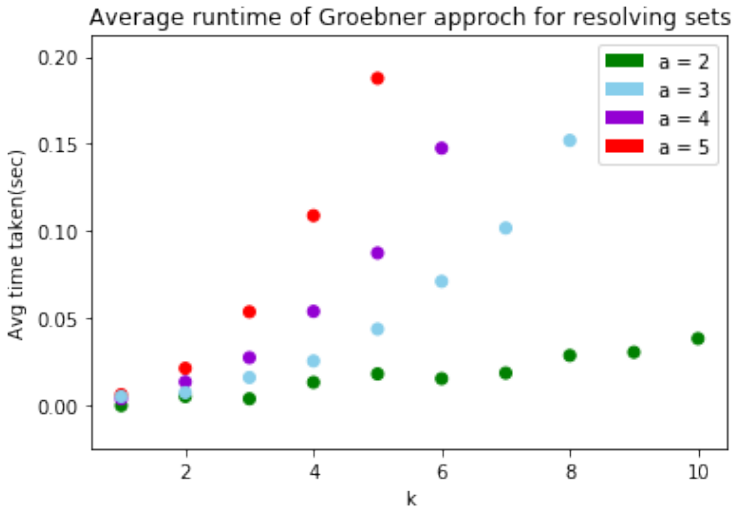
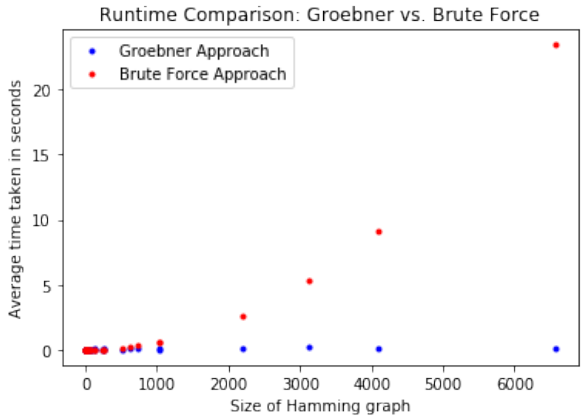


Figure 3: run-time of checking resolvability for resolving sets.

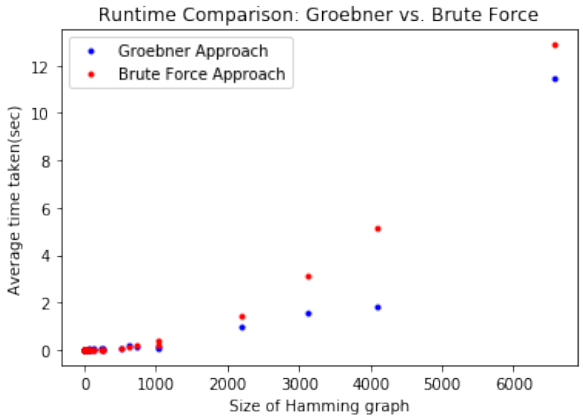
Figure 3 displays the average run-time for showing a set is resolving in $H_{k,a}$ (for various combinations of k and a) using the developed Gröbner basis algorithm. In this plot, there is a clear and promising polynomial growth in k, with almost linear behavior for $a = 2$. Additionally, the actual run-times are extremely impressive. All times are below 0.2 seconds—even for $H_{8,3}$ which contains over $6,500$ vertices. Note however that this is specifically for sets which are resolving on their associated Hamming graphs. As we will see throughout this section, resolving sets seem to produce linear systems with simple reduced Gröbner bases, drastically improving both the average run-time as well as the time consistency of our algorithm.

Next, we consider how the run-time of the brute-force algorithm and Gröbner basis algorithm compare as the size of the Hamming graph increases. As seen in Figure 4, the Gröbner basis algorithm vastly outperforms the brute-force algorithm for showing a set is resolving, boasting over 100 times faster results. By contrast, the Gröbner basis algorithm performs comparably to the brute-force algorithm for general sets where some are non-resolving. This shows that the Gröbner basis algorithm is faster for resolving sets than for non-resolving sets whereas the opposite is true with respect to the brute-force algorithm. A possible explanation is that resolving sets are structurally simpler than non-resolving sets. This results in a simpler system $P \cup Az = 0$ for resolving sets and the subsequent Gröbner basis computations are very quick. Non-resolving sets instead produce systems which have more than just the trivial solution whose associated Gröbner bases are much more cumbersome and complex to handle. The brute-force algorithm by comparison is much faster for non-resolving sets since it works by finding an unresolved pair of vertices. If no unresolved pair of vertices exist, the brute-force algorithm must go through all possible pairs, increasing the run-time.

Finally, we analyze the results of the Gröbner basis algorithm to get a better understanding of why the



(a) Gröbner basis algorithm significantly outperforms brute-force approach for resolving sets.



(b) Gröbner basis algorithm performs comparably to brute-force when also considering non-resolving sets

Figure 4

algorithm is slower on non-resolving sets. In Figure 5 there is a striking range of run-time results. While the vast majority of trials took under 10 seconds, one quarter of example sets took around 45 seconds. It turns out that those example sets were not resolving and were constructed with similar nodes, yielding similar linear systems and solutions. There are two possible explanations for this stark contrast in performance. As before, the linear systems constructed from these resolving sets may produce highly complex solution spaces which makes Gröbner basis computations very slow. It is odd however that there are almost no trials that land in between these two peaks in run-time. Another possible explanation is that SymPy Gröbner basis computations are unstable on these systems. Indeed, there are documented and unresolved issues with SymPy Gröbner basis computations where two systems of similar complexity take vastly different times to complete. The quarter of example sets with significantly higher runtime are very similar and produce nearly identical linear systems. It is possible that all of these systems were unstable for SymPy in the same ways. This could also explain the empty gap of run-times since systems are either drastically unstable and take a very long time to compute or they are stable and are completed within 10 seconds. A combination of both effects could certainly be at play, motivating potential future study to isolate the two different effects.

8 Discussion and Future Work

We have shown that resolvability on Hamming graphs can be characterized in terms of the solution space of a polynomial system specific to the Hamming graph, and a linear system constructed from the elements of a set of nodes R (Theorem 3.4). This characterization provides a novel framework with which to analyze resolving sets, connecting the theory of metric dimension on Hamming graphs to algebraic geometry and linear algebra. Our formulation can be simplified dramatically in hypercubes (Corollary 5.0.1). Through this simplification, we are able to retrieve the same linear system as Beardon [14], suggesting that our polynomial system somehow contains all the logical constraints implied by a binary alphabet. The characterization of resolvability has immediate utility in improving resolving set based applications such as k-mer embeddings, but it also illuminates a deep connection between algebraic geometry and Hamming graphs.

We have also exploited the block structure and symmetry of the polynomial system we encountered to explicitly define the Gröbner basis for any Hamming graph (Corollary 4.10.1). Further exploring the

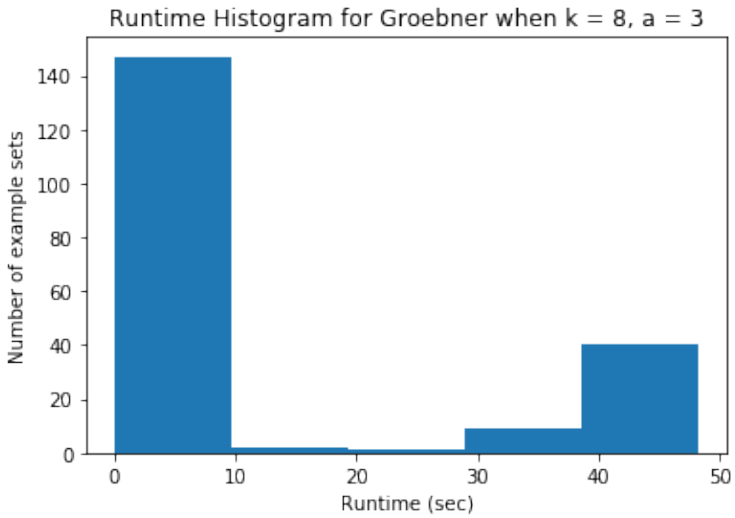


Figure 5: Histogram of Gröbner basis run-times on $H_{8,3}$. Almost $\frac{3}{4}$-ths of the trials took under 10 seconds with the last $\frac{1}{4}$ taking around 45 seconds.

block structure of the Gröbner bases could reveal potentially powerful optimizations of incremental basis computations, or further insights into resolvability at a structural level. Since these bases are nearly identically structured for all Hamming graphs, it might be possible to start with the Gröbner basis and try to derive a linear system which would cause it to reduce to $\{1\}$. Discovering such a derivation would amount to an explicit construction of resolving sets and possibly be the basis of proving the minimality of resolving sets.

Finally, we have analyzed the performance of the Gröbner basis theory for checking resolvability. The new method drastically outperforms the brute-force approach when the sets are resolving but there are significant reductions in speed for non-resolving sets. The specifics of these slow-downs suggest either highly complex solution spaces to the constructed linear systems or instability in the SymPy Gröbner basis computations for these specific examples. Testing this implementation in other computer algebra systems will be an important step in determining the root cause of the severe decrease in performance. Even with the potential SymPy instability, our new Gröbner basis algorithm performs better than brute-force on average while also providing a structural insight into resolvability on Hamming graphs.

References

[1] L. M. Kelly and E. A. Nordhaus, "Distance sets in metric spaces," *Trans. Amer. Math. Soc.*, vol. 71, pp. 440–456, 1951.

[2] P. J. Slater, "Leaves of trees," *Congressus Numerantium*, vol. 14, pp. 549–559, 1975.

[3] F. Haray and R. A. Melter, "On the metric dimension of a graph," *Ars Combinatoria*, vol. 2, pp. 191–195, 1976.

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

[5] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., 1990.

[6] J. Cáceres, C. Hernando, M. Mora, I. M. Pelayo, M. L. Puertes, C. Seara, and D. R. Wood, "On the metric dimension of cartesian products of graphs," *Siam Journal of Discrete Math*, vol. 21, no. 2, pp. 423–441, 2007.

[7] R. C. Tillquist and M. E. Lladser, "Low-dimensional representation of genomic sequences," *Journal of Mathematical Biology*, pp. 1–29, 3 2019.

[8] J. Kratica, V. Kovačević-Vujčić, and M. Čangalović, "Computing the metric dimension of graphs by genetic algorithms," *Computational Optimization and Applications*, vol. 44, no. 2, p. 343, 2007.

[9] M. Hauptmann, R. Schmied, and C. Viehmann, "Approximation complexity of metric dimension problem," *Journal of Discrete Algorithms*, vol. 14, pp. 214–222, 2012.

[10] R. K. Guy and R. J. Nowakowski, "Coin-weighing problems," *The American Mathematical Monthly*, vol. 102, no. 2, pp. 164–167, 1995.

[11] V. Chvátal, "Mastermind," *Combinatorica*, vol. 3, no. 3, pp. 325–329, 1983.

[12] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 855–864, ACM, 2016.

[13] W. J. Krzanowski, *Principles of Multivariate Analysis: A User's Perspective*. OUP Oxford, 2000.

[14] A. F. Beardon, "Resolving the Hypercube," *Discrete Applied Mathematics*, vol. 161, pp. 1882–1887, 2013.

[15] P. J. Olver and C. Shakiban, *Linear Algebraic Systems*, pp. 1–74. Cham: Springer International Publishing, 2018.

[16] D. A. Cox, J. Little, and D. O'Shea, *Gröbner Bases*, pp. 49–119. Cham: Springer International Publishing, 2015.

[17] D. Cox, J. Little, and D. O'Shea, *Using Algebraic Geometry*, vol. 1 of *Graduate Texts in Mathematics*. Springer-Verlag New York, 1998.

[18] J.-C. Faugère, "A new efficient algorithm for computing Gröbner bases F4," *Journal of Pure and Applied Algebra*, vol. 139, no. 1, pp. 61 – 88, 1999.

[19] J. C. Faugère, "A New Efficient Algorithm for Computing GröBner Bases Without Reduction to Zero (F5)," in *Proceedings of the 2002 International Symposium on Symbolic and Algebraic Computation*, ISSAC '02, (New York, NY, USA), pp. 75–83, ACM, 2002.

[20] C. Eder and J. E. Perry, "F5C: A variant of Faugère's F5 algorithm with reduced Gröbner bases," *J. Symb. Comput.*, vol. 45, no. 12, pp. 1442–1458, 2010. MEGA'2009.

[21] Y. Sun and D. Wang, "The F5 algorithm in Buchberger's style," *CoRR*, vol. abs/1006.5299, 2010.

[22] T. Tao, "Hilbert's nullstellensatz," November 2007.

[23] A. Meurer, C. P. Smith, M. Paprocki, O. Čertík, S. B. Kirpichev, M. Rocklin, A. Kumar, S. Ivanov, J. K. Moore, S. Singh, T. Rathnayake, S. Vig, B. E. Granger, R. P. Muller, F. Bonazzi, H. Gupta, S. Vats, F. Johansson, F. Pedregosa, M. J. Curry, A. R. Terrel, v. Roučka, A. Saboo, I. Fernando, S. Kulal, R. Cimrman, and A. Scopatz, "SymPy: symbolic computing in Python," *PeerJ Computer Science*, vol. 3, p. e103, Jan. 2017.