Fall 2019

Model Selection and Experimental Design of Biological Networks with Algebraic Geometry

Anyu Zhang
Southern Methodist University, anyuz@smu.edu

MODEL SELECTION AND EXPERIMENTAL DESIGN OF BIOLOGICAL NETWORKS WITH ALGEBRAIC GEOMETRY

Approved by:

Dr. Brandilyn Stigler
Associate Professor of Mathematics

Dr. Johannes Tausch
Professor of Mathematics

Dr. Amnon Meir
Professor of Mathematics

Dr. Elena S. Dimitrova
Associate Professor of Mathematics

MODEL SELECTION AND EXPERIMENTAL DESIGN OF BIOLOGICAL NETWORKS WITH ALGEBRAIC GEOMETRY

A Dissertation Presented to the Graduate Faculty of the

Dedman College

Southern Methodist University

in

Partial Fulfillment of the Requirements

for the degree of

Doctor of Philosophy

with a

Major in Computational and Applied Mathematics

by

Anyu Zhang

B.S., Mathematics, Beijing University of Chemical Technology
M.S., Computational and Applied Mathematics, Southern Methodist University

December 21, 2019

ACKNOWLEDGMENTS

I am highly grateful to the Department of Mathematics at Southern Methodist University for its contribution to my time in graduate school. More specifically, I would like to thank: my advisor, Dr. Brandilyn Stigler; my thesis committee members, Dr. Johannes Tausch, Dr. Amnon Meir, Dr. Elena S. Dimitrova; my fiancé, Dr. Shijie Sheng, without whom this thesis would not have been possible.

Zhang, Anyu B.S., Mathematics, Beijing University of Chemical Technology
M.S., Computational and Applied Mathematics, Southern Methodist University

Model Selection and Experimental Design of
Biological Networks with Algebraic Geometry

Advisor: Dr. Brandilyn Stigler

Doctor of Philosophy degree conferred December 21, 2019

Dissertation completed November 25, 2019

Model selection based on experimental data is an essential challenge in biological data science. In decades, the volume of biological data from varied sources, including laboratory experiments, field observations, and patient health records has seen an unprecedented increase. Mainly when collecting data is expensive or time-consuming, as it is often in the case with clinical trials and biomolecular experiments, the problem of selecting information-rich data becomes crucial for creating relevant models.

Motivated by certain geometric relationships between data, we partitioned input data sets, especially data sets that correspond to a unique basis, into equivalence classes with the same basis to identify a unique algebraic model. The analysis of the data relationships and properties will facilitate the computations, storage, and access to sizable discrete data sets.

TABLE OF CONTENTS

CHAPTER

APPENDIX

LIST OF FIGURES

LIST OF TABLES

This thesis is dedicated to my parents, Ms. Xuehua Mao and Mr. Jianping Zhang and to my grandparents, Mr. Zhangyuan Zhang and Ms. GuoZhen Xu.

Chapter 1
INTRODUCTION

A general data science life cycle in Figure 1.1 contains five stages: 1. data capture, which includes data acquisition, data extraction and data entry. In this step, researchers and data scientists pay more attention to improve the data quality; 2. data maintain, which contains data processing, data cleansing and data architecture. Especially for large data sets, an efficient data maintenance system is essential to data readability and sustainability; 3. data process contains data mining, classification, data modelling. Insights can be generated by models and algorithms, such as the classification of data; 4. data analysis includes predictive analysis, qualitative analysis. Predictive/regression models should be applied to analyze data properties, such as the associative relationships between data points; 5. data communication, which has data reporting, data visualization, decision making, is an essential step to extract insights from data analysis results to help to make a future decision.

1.1. Gene Regulatory Networks

Gene, an increasingly popular topic nowadays, has been in our life every day by holding the information of building the cells in our body. In other words, genes contain instructions for the mechanisms of cellular processes. *Gene regulation* is a process of genetic information extracting and utilizing. The mathematical model of gene regulation is extremely complicated as it involves genes, DNA, RNA, proteins and small molecules. We call the network that contains gene regulation a *gene regulation network* (GRN). A GRN is a collection of regulators: proteins, genes and enzymes. These regulators will interact with each other to fulfil some functionalities. An earrly example of a GRN is considered to be the *lac* operon in 1961. In this GRN, proteins in lactose metabolism are expressed by *E. coli*.

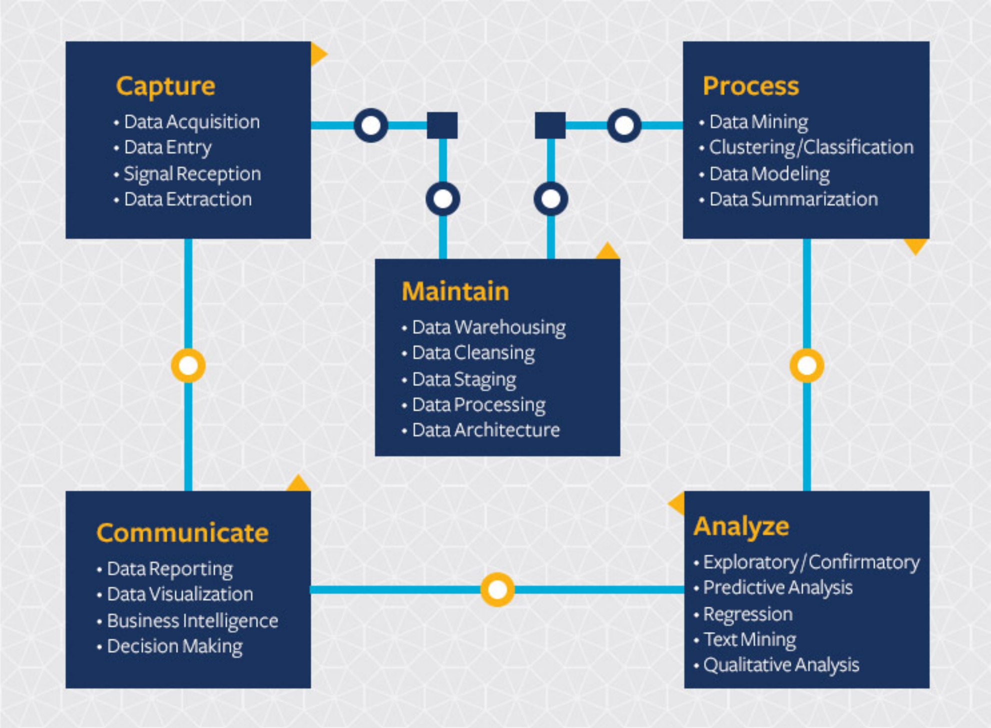


Figure 1.1: Life Cycle of Data Science [7].

1.2. Model Selection

Network inference in systems biology is plagued by too few input data and too many candidate models which fit the data. However, not all candidate models reflect the behaviours of real biological systems. Different model selection strategies have been developed to restrict the model selection to more appropriate dynamic networks. Some strategies put restrictions of the network topology [25]. Some other model selection strategies will restrict functions to be biologically motivated, such as the chain functions [22] or biologically meaningful functions [38]. When the data are discrete, models can be written as a linear combination of *finitely* many monomials. The problem of selecting a model can be reduced to selecting an appropriate monomial basis.

To implement the model selection on GRN, standard monomials of GRN models and data set partitions are used. Based on the traditional five stages of the data science life cycle, our work in model selection on GRN and model classification is contained in three steps: process, analyze and communicate.

- Process

  Discrete data sets, such as inputs in the *lac* operon [49] network, which are based on the experiments on the gene activities, are used to construct gene network models and make a model selection. Since model selection can be applied to all possible data sets in each case with the specific number of states p, the number of variables n and the number of points m, all possible data sets are generated in different cases based on supporting algorithms written in Python and Macaulay2. The outlier data sets and the problems related to data collection will not be considered in this work.

- Analyze

  First, the properties of models based on geometic characterization and Gröbner bases (GB) theorems are studied. To have a better estimation of models, the formulas for the number of GBs and a more accurate upper bound of the number of models based on algebraic geometry are generated. With a better estimate for the number of models, scientists can calculate the approximate time they need to construct a new model. In this way, scientists can avoid computational redundancy.

  Then, the relationship between input data sets and the relationship between input data sets and model bases are considered. From data sets to models, different GBs are generated through coding in Macaulay2 and C. We also improved the algorithms in calculating GBs in Python. From the calculation of GBs, it is common for different data sets to share the same GBs. This property gave us the motivation to study the relationship between data sets. Data sets partition/classification research is expensive, considering the high experimental cost with redundancy model information from different data sets.

  Lastly, the relationship between data sets geometrically was explored and the characterization for unique Gröbner basis data sets and non-unique Gröbner basis data sets were considered. Then, we applied linear algebra methods to create a relationship matrix between data sets. This idea proved to be practical as recently affine transformations were used to partition input data into equivalence classes on the same basis.

To better represent different classes, a “standard position” was proposed for data sets. Data set distance was proposed to measure how far a set is from being in a standard position.

- Communicate

  According to the life cycle of data science in Figure 1.1, data visualization, decision making or data reporting are better ways to communicate meaningful data analysis results to society. The website was created DoEMS to help extract data analysis reports and statistical results. More details and functionality will be shown in Chapter 5. With DoEMS, researchers can analyze a model with standard monomials and equivalence classes of data sets to make an experimental design.

1.3. Experimental Design

Gene regulatory networks often contain a significant amount of uncertainty. The methodology is desirable for prioritizing potential experiments to reduce network uncertainty optimally, considering the cost and time required for biological experiments. This process of prioritizing biological experiments to reduce the uncertainty of GRN is called *experimental design* [13].

As suggested in two review papers of computational and experimental approaches in GRN [24, 41], to proceed, researchers should consider the following steps: 1. Construct candidate mathematical models of gene regulation. 2. Design experiments that produce maximally informative observations. 3. Develop methodologies for choosing a candidate model that ‘best’ fits the observations. 4. Analyze and validate the model. Use the model to formulate and test new hypotheses about the structure and function of gene regulation.

- Capture

  The capture step in the life cycle of data science based on the experience gained from the previous steps is the experimental design step. For example, considering the results from the analyze step in Section 1.2, equivalence classes are generated on the same basis. The implication of this work is a guide for biologists in designing experiments to collect data that result in a unique model (set of predictions), thereby reducing ambiguity in modelling and improving predictions.

Moreover, we can get further details or guidance from DoEMS [50] developed in the communicating step as researchers can make a comparison of ECs with their desirable data sets. Then they can plan to add some extra points or start from specific data sets, which associate with specific models or even choose the data sets with fewest active GRN nodes or the data sets with the most active GRN nodes.

Chapter 2

BACKGROUND

2.1. Algebraic Geometry

Most definitions and known results in this section can be found in [10].

Let $\mathbb{F}$ be a finite field of characteristic p. We will typically consider the finite field $\mathbb{Z}_p = \{0, 1, \ldots, p-1\}$, that is the field of remainders of integers upon division by p with modulo-p addition and multiplication. Let $R = \mathbb{F}[x_1, \ldots, x_n]$ be a polynomial ring over the finiete field $\mathbb{F}$. Finally let m denote the number of points in a subset of $\mathbb{F}^n$.

Definition 2.1 *A* monomial order $\prec$ *(sometimes called a term order or an admissible order) is a total order on the set of all (monic) monomials in a given polynomial ring, satisfying*

If $u \prec v$ and w is any other monomial, then $uw \prec vw$.

The choice of a term order $\prec$ on the monomials allows for sorting the terms of a polynomial. Let us take a set of monomials $s = \{x_1^2, x_2^2, x_1x_3, x_1, 1\}$, as an example to show some well known monomial orders.

Example 2.1 *Lexicographic* order (lex), with $x_1 > x_2 > \ldots > x_n$, first compares exponents of x_1 in the monomials, and in the case of equality compares exponents of x_2, and so forth. Then, in lex order

$$x_1^2 > x_1x_3 > x_1 > x_2^2 > 1.$$

□

Example 2.2 *Graded lexicographic order* (grlex, or deglex), with $x_1 > x_2 > \ldots > x_n$, first compares the total degree (sum of exponents) and in the case of same total degree applies lexicographic order. Then, in grlex order

$$x_1^2 > x_1x_3 > x_2^2 > x_1 > 1.$$

□

Example 2.3 *Graded reverse lexicographic order* (grevlex, or degrevlex), with $x_1 > x_2 > \ldots > x_n$, compares the total degree first, then uses a reverse lexicographic order in the case of same total degree, it reverses the outcome of the lexicographic such that larger monomials of the same degree are considered to be smaller in grevlex. Then, in grevlex order

$$x_1^2 > x_2^2 > x_1x_3 > x_1 > 1.$$

□

Definition 2.2 *A subset $I \subset \mathbb{F}[x_1, \ldots, x_n]$ is an* ideal *if it satisfies:*

- $0 \in I$.
- *If f, $g \in I$, then $f + g \in I$.*
- *If $f \in I$ and $h \in \mathbb{F}[x_1, \ldots, x_n]$, then $hf \in I$.*

Definition 2.3 *If $f_1, \ldots, f_s$ are polynomials in $\mathbb{F}[x_1, \ldots, x_n]$, then we set*

$$\langle f_1, \ldots, f_s \rangle = \Big\{ \sum_{i=1}^{s} h_i f_i : h_1, \ldots, h_s \in \mathbb{F}[x_1, \ldots, x_n] \Big\}$$

Here, $\langle f_1, \ldots, f_s \rangle$ is an ideal of $\mathbb{F}[x_1, \ldots, x_n]$. We call $\langle f_1, \ldots, f_s \rangle$ the *ideal generated by* $f_1, \ldots, f_s$.

Definition 2.4 *The* leading term *of a polynomial $g \in \mathbb{F}[x_1, \cdots, x_n]$ is thus the term of the largest monomial for the chosen monomial order $\prec$, written $LT_\prec(g)$. Also, we call $LT_\prec(I) = \langle LT_\prec(g) : g \in I \rangle$ the* leading term ideal *for an ideal I.*

Notice also if f and g are nonzero polynomials, then $deg(f) \leq deg(g) \iff LT(f)$ divides $LT(g)$. Based on the division algorithm, every $f \in R$ can be written as $f = q \cdot g + r$, where $q, r \in R$, and either $r = 0$ or $deg(r) < deg(g)$.

Example 2.4 With grevlex order and variable order $x_1 > x_2 > x_3$, the polynomial $g = x_1x_2^2 + x_1^3 + x_1x_2 + x_1 + 1$ has the leading term $x_1x_2^2$. While with grlev order, the leading term of g is x_1^3. □

Definition 2.5 *The monomials which do not lie in $LT_\prec(I)$ are called* standard monomials, *denoted $SM_\prec(I)$.*

Example 2.5 Let $LT_\prec(I) = \{x_1^2, x_2\}$ be a set of leading terms of an ideal I. The set of standard monomial is $SM_\prec(I) = \{1, x_1\}$. □

Definition 2.6 *Let $f_1, \ldots, f_s \in R$. Then the set*

$$\mathbf{V}(f_1, \ldots, f_s) \coloneqq \{(a_1, \ldots, a_n) \in \mathbb{F}^n : f_i(a_1, \ldots, a_n) = 0, 1 \leq i \leq s\}$$

is the affine variety defined by $f_1, \ldots, f_s$.

Definition 2.7 *Given input-output data $V = \{(s_1, t_1), \ldots, (s_m, t_m)\} \subset \mathbb{F}^n \times \mathbb{F}^n$, then the set I of polynomial functions which vanish on the inputs, called the* ideal of points *is computed as*

$$I(\{s_1, \ldots s_m\}) \coloneqq \{h \in R \mid h(s_i) = 0 \, \forall i\} = \cap_{i=1}^m \langle x_j - s_{ij} \rangle \text{ where } s_i \in (s_{i1}, \ldots, s_{in}).$$

Here, s_i, t_i are vectors, $i = 1, \ldots, m$.

Definition 2.8 *Let $\prec$ be a term order on the monomials in R and let I be an ideal in R. Then $G \subset I$ is a* Gröbner basis *for I with respect to $\prec$ if $\langle G \rangle = I$ for all $f \in I$ there exists $g \in G$ such that the leading term $LT_\prec(g)$ divides $LT_\prec(f)$.*

Definition 2.9 *A* reduced Gröbner basis *for a polynomial ideal I is a Gröbner basis G for I such that:*

- *Leading coefficient of g is 1, for all $g \in G$.*

- *For all $g \in G$, no monomial of g lies in $\langle LT(G - \{g\})\rangle$.*

Definition 2.10 (Normal Form) *Let $G = \{g_1, \ldots, g_t\}$ be a Gröbner basis w.r.t. $\prec$ for an ideal $I \subset \mathbb{F}[x_1, \ldots, x_n]$ and let $f \in \mathbb{F}[x_1, \ldots, x_n]$. Then there is a unique polynomial $r \in \mathbb{F}[x_1, \ldots, x_n]$, called the* normal form *of f w.r.t. $\prec$ with following two properties:*

- *No term of r is divisible by any of $LT(g_1), \ldots, LT(g_t)$.*

- *There is $g \in I$ such that $f = g + r$.*

Theorem 2.1 (Buchberger Algorithm) *Let $I = \langle f_1, \ldots, f_s\rangle \neq \{0\}$ be a polynomial ideal. Then a Gröbner basis for I can be constructed in a finite number of steps by the following algorithm:*

Input: A set of polynomials $F = \{f_1, \ldots, f_s\}$ that generates I; a monomial order $\prec$.

Output: A Gröbner Basis G for I.

1. *$G \coloneqq F$*

2. *$g_i \coloneqq LT_{\prec}(f_j), \forall f_j \in G$ and $a_{ij} \coloneqq LCM(g_i, g_j)$ (least common multiple of g_i and g_j).*

3. *Choose two polynomials f_i and f_j in G and let $S_{ij} \coloneqq (a_{ij}/g_i)f_i - (a_{ij}/g_j)f_j$ (leading terms will cancel by construction).*

4. *Reduce S_{ij} until the result is not further reducible, with multivariate division algorithm [10]. Then add non-zero result to G.*

5. *Repeat Steps 1-4 for all possible S_{ij} and new polynomials generated at Step 4.*

6. *Output G.*

Example 2.6 Considering input data set $S_1 = \{(0,0),(0,1)\}$, the ideal of points is $I = (x_1, x_2) \bigcap (x_1, x_2 - 1)$. By applying Algorithm 2.1, we can get a unique GB $\{\underline{x_2^2} - x_2, \underline{x_1}\}$ Then S_1 is associated with a unique standard monomial basis $\{1, x_2\}$ (see Definition 2.5). Also, for the data set $S_2 = \{(1,0),(1,1)\}$, the ideal of points is $I = (x_1 - 1, x_2) \bigcap (x_1 - 1, x_2 - 1)$.

By applying Algorithm 2.1, we can get $GB(I) = \{\underline{x_2^2} - x_2, \underline{x_1}\}$ and the standard monomial basis is $\{1, x_2\}$. □

Example 2.7 Consider two inputs $S_3 = \{(0,0),(1,1)\} \subset (\mathbb{Z}_2)^2$. The corresponding ideal $I = (x_1, x_2) \bigcap (x_1 - 1, x_2 - 1)$ of the points in S_3 has two distinct reduced Gröbner bases, namely

$$G_1 = \{\underline{x_1} - x_2, \underline{x_2^2} - x_2\}, G_2 = \{\underline{x_2} - x_1, \underline{x_1^2} - x_1\}$$

Here, '_' marks the leading terms of the polynomials in a Gröbner basis.

First, leading coefficients of elements in bases G_1 and G_2 are all 1. Second, no monomial in any element of the basis is in the ideal generated by the leading terms of the other elements of the basis. For example, for $g_1 = x_1 - x_2$ in G_1, neither x_1 nor x_2 lies in $\langle x_2^2 \rangle$. The leading term of $g_2 = x_2^2 - x_2$ is x_2^2 as indicated. So G_1 and G_2 are two reduced Gröbner bases associated with input S. Then the standard monomial basis of G_1 is $\{1, x_2\}$ and the standard monomial basis of G_2 is $\{1, x_1\}$. □

To save the computational cost of GRN with numerous input data sets, some specific data structure, especially structures related to unique Gröbner basis. The most important structure is called a staircase.

Definition 2.11 *A* staircase *is a nonempty subset $\lambda \subseteq \mathbb{F}^n$ such that if $u \in \lambda$ and $v \leq u$ (coordinate-wise), then $v \in \lambda$.*

Example 2.8 $\lambda_1 = \{(0,0),(1,0),(0,1)\}$ is a staircase. However, $\lambda_2 = \{(1,0),(0,1),(1,1)\}$ is not a staircase, as point $(0,0)$ which is smaller than $(1,0)$ is not in λ_2. □

Let $\binom{\mathbb{F}^n}{m}$ denote the collection of all sets of m points in $\mathbb{F}^n$. Then for $\lambda = \{\lambda_1, \ldots, \lambda_m\} \in \binom{\mathbb{F}^n}{m}$, let $\sum \lambda$ denote the vector sum $\sum_{i=1}^m \lambda_i \in \mathbb{F}^n$. Let Λ denote the set of all staircases in $\binom{\mathbb{F}^n}{m}$. The *staircase polytope* of Λ is the convex hull of all points $\sum \lambda$ where $\lambda \in \Lambda$ (see [6, 35] for more details).

Definition 2.12 *For an ideal I, we call $\mathcal{P}$ the* staircase polytope of *I if $\mathcal{P}$ is the staircase polytope of the exponent vectors of the standard monomial sets associated to I for any monomial order.*

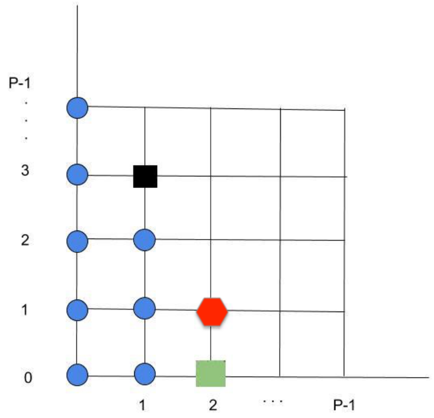


Figure 2.1: The lattice graph of a staircase. By adding either black or green square points, we obtain a new staircase. However, adding the red hexagonal point will break the rules of staircases.

The number of reduced Gröbner bases for an ideal is in bijection with the number of vertices of the staircase polytope, which was proved for ideals of points in [35] and for all other zero-dimensional ideals in [6].

Theorem 2.2 (**[21]**) *Let $S \subseteq \mathbb{F}^n$ and $I(S)$ be the ideal of the points in $\mathbb{F}$. Then $|S| = \dim_{\mathbb{F}} R/I(S)$.*

An ideal is *zero dimensional* if $\dim_K R/I < \infty$; when K is algebraically closed and $|S| = m < \infty$, $m = \dim_K R/I(S)$. A field K is *algebraically closed* iff the only irreducible polynomials in the polynomial ring $K[x]$ are those of degree one. When working over a finite field, extensions of classic results in algebraic geometry state that when the number m of input points is finite, then m coincides with the dimension of the vector space $R/I(S)$ over $\mathbb{F}$ [21], which is stated in Theorem 2.2.

Example 2.9 Let $S = \{(1,1),(2,3),(3,5),(4,6)\} \subset \mathbb{R}^2$. So $\dim_{\mathbb{R}} \mathbb{R}[x,y]/I(S) = 4$. Also $\Lambda(I(S)) = \{(1,x,x^2,x^3),\ (1,x,x^2,y),\ (1,x,y,y^2),\ (1,y,y^2,y^3)\}$. So the number of reduced Gröbner bases for $I(S)$ is four. Note that there are five staircases in $\binom{\mathbb{F}^2}{4}$, namely $\Lambda = \{\{(0,0),(1,0),(2,0),(3,0)\},\ \{(0,0),(1,0),(2,0),(0,1)\},\ \{(0,0),(1,0),(0,1),(1,1)\},\ \{(0,0),(1,0),(0,1),(0,2)\},\ \{(0,0),(0,1),(0,2),(0,3)\}\}$. The staircase polytope of Λ is the convex

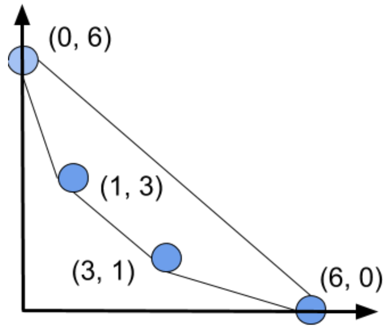


Figure 2.2: The staircase polytope graph of $S = \{(1,1),(2,3),(3,5),(4,6)\}$. Vertices are calculated as the sums of the coordinates of the five staircases in this example.

hull of the vector sums {(6,0), (3,1), (2,2), (1,3), (0,6)}, which has vertices (6,0), (3,1), (1,3), and (0,6), corresponding to the four standard monomial sets of $I(S)$. □

2.2. Polynomial Dynamical Systems

Definition 2.13 *A* polynomial dynamical system *(PDS) over* $\mathbb{F}$ *is a function* $f = (f_1, \ldots, f_n) : \mathbb{F}^n \to \mathbb{F}^n$ *where* $f_i \in R$*.*

It is well known that GBs exist for every $\prec$ and make multivariate polynomial division well defined in that remainders are unique. A set of standard monomials $SM_{\prec}(I)$ for a given term order forms a basis for R/I as a vector space over $\mathbb{F}$.

Given input-output data $V = \{(s_1,t_1),\ldots,(s_m,t_m)\} \subset \mathbb{F}^n \times \mathbb{F}^n$, find all PDSs [29] that fit V and select a minimal PDS with respect to polynomial division. The general strategy is as follows: For each x_j, compute one interpolating function $f_j \in R$ such that $f_j(s_i) = t_{ij}$, then compute the ideal $I = I(\{s_1,\ldots,s_m\})$ of the input points.

The *model space* for V is the set

$$f + I := \{(f_1 + h_1, \ldots, f_n + h_n) : h_i \in I\}$$

of all PDSs which fit the data in V and where $f = (f_1,\ldots,f_n)$ is as computed above. A PDS can be selected from $f + I$ by choosing a monomial order $\prec$, computing a Gröbner basis G

for I, and then computing the normal form (remainder), denoted $\overline{f}^{G}$, of each f_i by dividing f_i by the polynomials in G. We call

$$(\overline{f_1}^{G}, \overline{f_2}^{G}, \ldots, \overline{f_n}^{G})$$

the *minimal* PDS with respect to $\prec$, where G is a Gröbner basis for I with respect to $\prec$. Changing the monomial order may change the resulting minimal PDS. While it is possible for two reduced Gröbner bases to give rise to the same normal form (see [29]), it is still the case that in general, a set of data points may have *many* GBs associated with it. In this way, the number of distinct reduced GBs of I gives an upper bound for the number of different minimal PDSs. Therefore, we aim to find the number of distinct reduced Gröbner bases for a given data set.

Example 2.10 From Example 2.7, $S_3 = \{(0,0),(1,1)\} \subset (\mathbb{Z}_2)^2$ has two distinct reduced Gröbner bases

$$G_1 = \{\underline{x_1} - x_2, \underline{x_2^2} - x_2\}, G_2 = \{\underline{x_2} - x_1, \underline{x_1^2} - x_1\}$$

Hence, there are two resulting minimal models: any minimal PDS with respect to G_1 will be in terms of x_2 only as x_1 is the leading term and all x_1's in PDS are divided out, which results standard monomial basis to be $\{1, x_2\}$. while, any minimal PDS with respect to G_2 will be in terms of x_1 only as x_2 is the leading term and all x_2's in PDS are divided out. Then the standard monomial basis is $\{1, x_1\}$.

If the inputs set is $\{(0,0),(0,1)\}$, then its associated ideal I has a unique GB, $\{\underline{x_2^2} - x_2, \underline{x_1}\}$, resulting in a unique minimal PDS.

The polynomial $g = x_1 - x_2$ has different leading terms for different monomial orders. In fact, for monomial orders with $x_1 \succ x_2$, the leading term of g will be x_1, while for orders with $x_2 \succ x_1$ the opposite will be true. We say that g has *ambiguous* leading terms. We will mark only ambiguous leading terms.

□

As the elements of the quotient ring, R/I are equivalence classes of functions defined over the inputs $\{s_1, \ldots s_m\}$ and since a set of standard monomials is a basis for R/I, it follows that each reduced polynomial $\overline{f}^G$ is written in terms of standard monomials.

A combinatorial structure that contains information about all reduced Gröbner bases of a polynomial ideal I is the *Gröbner Fan* of I. It is a polyhedral complex of cones, each corresponding to an initial ideal of I. [15, 33]

The cones are in one-to-one correspondence with the reduced GB of I.

2.3. Linear Shifts of Data Sets

The linear shift was first introduced in [14] and [26] as a new term to describe results after affine transformation for any data sets in $\mathbb{F}$. Both of them defined the linear shift using mapping functions. However, Definition 2.14 provides a ring-centered definition of a linear shift while Definition 2.15 provides a data-centered definition.

Definition 2.14 ([14]) *Let $a_1, \ldots, a_n \in \mathbb{F} \setminus \{0\}$, let $b_1, \ldots, b_n \in \mathbb{F}$, and let $\Phi : R \to R$ be the homomorphism defined by $x_i \mapsto a_i x_i + b_i$ for $i = 1, \ldots, n$. Then Φ is called a* linear shift *of R.*

Definition 2.15 ([26]) *Let $S_i = \{\mathbf{x}_1^i, \ldots, \mathbf{x}_n^i\}, S_j = \{\mathbf{x}_1^j, \ldots, \mathbf{x}_n^j\} \subset \mathbb{F}^n$ be data sets. We say that S_i is a* linear shift *of S_j, denoted $S_i \sim S_j$, if there exists $\phi = (\phi_1, \ldots, \phi_n) : \mathbb{F}^n \to \mathbb{F}^n$ such that $\phi_k(\mathbf{x}_k^i) = a_k \mathbf{x}_k^i + b_i = \mathbf{x}_k^j$, $a_k \in \mathbb{F}^\times, b_k \in \mathbb{F}$, $k = 1, \ldots, n$, and $S_i = \phi(S_j)$. We will denote input data sets as*

$$S_i = \{(x_{11}^i, x_{21}^i, \ldots, x_{n1}^i), \ldots, (x_{1m}^i, x_{2m}^i, \ldots, x_{nm}^i)\}$$

and

$$S_j = \{(x_{11}^j, x_{21}^j, \ldots, x_{n1}^j), \ldots, (x_{1m}^j, x_{2m}^j, \ldots, x_{nm}^j)\}.$$

Here, $\mathbf{x}_k^i = (x_{k1}^i, x_{k2}^i, \ldots, x_{km}^i)$ is the vector of the k-th coordinate in the data set S_i, and $\mathbf{x}_k^j = (x_{k1}^j, x_{k2}^j, \ldots, x_{km}^j)$ is the vector of k-th coordinate in the data set S_j.

As the research in [26] is more focused on the relationship between data sets, which is more applicable for GRN model research with input data sets, we will use the point version of Definition 2.15 instead of the ring version of Definition 2.14 for the discussion in Chapter 3, Chapter 4 and Chapter 6.

Example 2.11 For $n = 2, m = 3, p = 3$, in Figure 2.3, the data set (black points) in left, center and right is the linear shift of data set (green points) by function set $\phi = (\phi_1, \phi_2)$. □

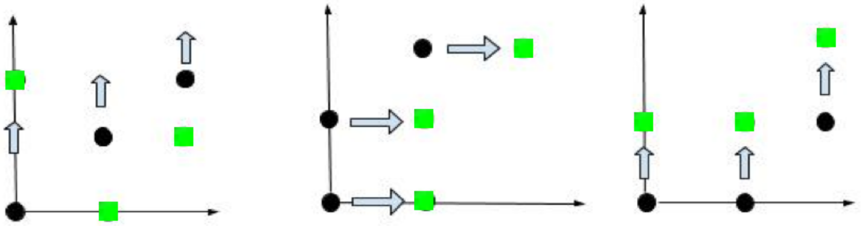


Figure 2.3: The data set (black points) $\{(0,0),(1,1),(2,2)\}$ in the left plot are linear shifts of the data set (green points) $\{(0,2),(1,0),(2,1)\}$ via $\phi = (\phi_1,\phi_2) = (x, x+2)$.
The data sets (black points) $\{(0,0),(0,1),(1,2)\}$ in the center plot is the linear shift of the data set (green square points) $\{(1,0),(1,1),(2,2)\}$ via $\phi = (\phi_1,\phi_2) = (x+1, x)$.
The data sets (black points) $\{(0,0),(1,0),(2,1)\}$ in the right plot is the linear shift of the data set (green square points) $\{(0,1),(1,1),(2,2)\}$ via $\phi = (\phi_1,\phi_2) = (x, x+1)$.

Theorem 2.3 ([26]) *If $S_1 \sim S_2$, then $I(S_1)$ and $I(S_2)$ have the same number of reduced Gröbner bases. In particular, when $I(S_1)$ has a unique reduced Gröbner basis, $I(S_2)$ will also have a unique reduced Gröbner basis.*

Example 2.12 In Example 2.6, we know that the inputs $S_1 = \{(0,0),(0,1)\}$ have a unique GB $\{\underline{x_2^2} - x_2, \underline{x_1}\}$. Notice that S_1 and S_2 return the same GB and standard monomial $SM = \{1, x_2\}$. Based on Theorem 2.3, $S_1 = \{(0,0),(0,1)\}$ is the linear shift of the points in $S_2 = \{(1,1),(0,1)\}$,

$$\{(0,0),(0,1)\} \xRightarrow{(\phi_1,\phi_2)} \{(1,1),(0,1)\}$$

with mapping functions $\phi_1 = x+1$ and $\phi_2 = x$. By applying the function f_1 to the first coordinate and f_2 to the second coordinate of the points in the data set $\{(0,0),(0,1)\}$, we will shift the data set to $\{(1,1),(0,1)\}$. □

Theorem 2.4 ([26]) *If $S \subset \mathbb{F}^n$ is a staircase, then $I(S)$ has a unique reduced Gröbner basis.*

Based on Theorem 2.4 and Theorem 2.3, we can get a sufficient condition for unique reduced Gröbner basis.

Proposition 2.3.1 *If $S \subset \mathbb{F}^n$ is a linear shift of some staircase $\lambda \subset \mathbb{F}^n$, then $I(S)$ has a unique Gröbner basis.*

Example 2.13 $S_1 = \{(0,0),(1,0),(0,1)\}$ is a staircase in $\mathbb{F}^2$ with $p=3$. $S_2 = \{(0,1),(2,2),(0,2)\}$ is a subset of $\mathbb{F}^2$ and $S_1 \sim S_2$, since $S_2 = \Phi(S_1)$, where $\phi_1 = 2x$, $\phi_2 = 2x+2$. $I(S_2)$ has a unique reduced GB, $\{x_1x_2 + x_1, x_2^2 - 1, x_1^2 + 2\}$. □

However, a linear shift of a staircase is not a necessary condition for $I(S)$ to have a unique reduced GB. See the next example.

Example 2.14 Data set $S = \{(0,0,0),(0,0,1),(0,1,0),(1,0,1)\}$ is not a linear shift of a staircase in $\mathbb{F}^3$ with $p=2$. However, $I(S)$ has a unique reduced GB, $\{x_3^2 + x_3, x_2x_3, x_2^2 + x_2, x_1x_3 + x_1, x_1x_2, x_1^2 + x_1\}$. □

The observation in Example 2.6 and Example 2.14 motivates us to study relationships between input data sets. We already know that if two data sets are linear shift of each other, they will have same number of GBs based on Theorem 2.3.

The linear shift is meaningful in model selection and data partition as we can use partition to a cluster data sets only by checking if one data set is a linear shift of all the others. Based on Theorem 2.3, only by checking the linear shift relationship between input data sets, we can distinguish different model basis instead of calculating each basis. In this chapter, we made data partition by linear shift relationship. In the next chapter, we will dive into each equivalence class associated with different Gröbner basis concerning model selection in the finite fields.

Chapter 3

MODEL ESTIMATION

In this chapter, we will focus on estimating the number of reduced Gröbner bases and the number of models associated with a data set. As each Gröbner basis possibly gives a distinct model [14], we use GBs as a way of estimating the number of models for a data set. So we will count the number of GBs associated with a data set. In this chapter, we will start our discussion from the geometric characterization of data sets, then extend to the estimation of the number of GBs. We will go through the formulas in different finite fields and a general upper bound in any finite field to discover the relationship between data sets associated with models in the finite field $\mathbb{Z}_p$.

The work of the chapter has been published in [44].

3.1. Introduction

In recent decades, scientists and mathematicians developed many efficient solvers in the polynomial system [3, 9, 32, 43, 47] based on Gröbner bases. In all these applications, researchers focused on generating models but did not offer solutions to distinguish between models with different input data sets.

There are many novel applications of Gröbner bases. For example, in [30], the authors introduced a new method that goes beyond Gröbner bases in the field of 3D scenery image mining. In [2], the authors showed there are applications of Gröbner bases in robotics engineering. Gröbner bases and polynomial ideal theory are used to solve the problem of the S-packing colouring of a finite undirected and unweighted graph. However, these new applications still have to enumerate all monomial bases in the computation process. The computational efficiency of Gröbner bases is an urgent problem based on review [16] of

Gröbner bases. More specifically, in the process of improving GB computation efficiency, scientists focused on two directions: 1. improve the algorithm, such as F_4 [8] and F_5 [17], which are traditional GB algorithms. The authors in [31] introduced a new algorithm M4GB for computing GBs based on the F4 algorithm. 2. Estimation of computational cost [35], The authors developed theorems with polytope of input points and proved an upper bound for the number of GBs.

The traditional method is to enumerate all GBs of ideals [20]. Researchers have found new algorithms to improve the complexity [28]; however, we know that GB computation still has exponential complexity with numerous inputs.

Surprisingly there is no closed form for the number of reduced GBs for an ideal. What is known is an upper bound [35], which is not sharp, especially when the characteristic of the field is positive. In this work, we will characterize the number of GBs in a finite field in 3 ways:

1. geometrically characterize the number of GBs. We will generate geometric characterization of input data sets which are associated with a specific number of GBs. Individually, we will pay attention to data set structure associated with unique reduced GBs.

2. creates and prove the formulas of number of GBs in simple cases. We will extend our discussion on the number of distinctly reduced Gröbner bases for different cases based on geometric observations.

3. prove a better upper bound of the number of GBs in a finite field. Comparing our new upper bound with previous works [35] shows a better estimation on the number of GBs of sizable data sets. The new upper bound can help estimate the cost of computing reduced Gröbner bases before real experiments.

Considering the $\mathbb{Z}_p^n$ which contains p^n points. For $n = 1$, all ideals have a unique reduced GB since all polynomials are single-variate and as such are factor closed. We consider cases for $n > 1$. We say that a polynomial $f \in R$ is *factor closed* if every monomial $m \in supp(f)$

is divisible by all monomials smaller than m with respect to an order $\prec$. The following result gives an algebraic description of ideals with unique reduced GBs for any monomial order.

Theorem 3.1 ([14]) *A reduced Gröbner basis G with factor-closed generators is reduced for every monomial order; that is, G is the unique reduced Gröbner basis for its corresponding ideal.*

For empty sets or singletons in $\mathbb{Z}_p^n$, it is straightforward to show that the ideal of points has a unique reduced GB for any monomial order; that is, for a point $s = (s_1, \ldots, s_n)$, the associated ideal of s is $I = \langle x_1 - s_1, \ldots, x_n - s_n \rangle$ whose generators form a Gröbner basis which is unique (via Theorem 3.1). According to Theorem 3.3, the same applies to $p^n - 1$ points. In the rest of this work, we consider the number of reduced Gröbner bases for an increasing number of points.

Note that over a finite field, the relation $x^p - x$ always holds.

3.2. Characterizing Data Sets Geometrically

The description of geometric characteristics can express essential features in many fields like dynamical systems. Scientists can explore nuances of specific dynamical systems based on their geometric behaviour with different boundary conditions. In our problem of counting the number of Gröbner bases, the geometric characteristics in unique GB and nonunique GBs models can provide scientists with a more intuitive way in exploring the experimental data sets. We aim to identify a connection between the geometric configuration of data sets and the number of associated GBs.

3.2.1. Geometric Characteristics of Two Points

Example 3.1 Consider two points in $\mathbb{Z}_2^2$. The left graph in Figure 3.1 is the plot of all points in $\mathbb{Z}_2^2$. By decomposing the 2-square on which they lie, we find that pairs of points that lie along horizontal lines have unique reduced Gröbner bases for any monomial order; see Figure 3.2. For example, $\{(0, 0), (0, 1)\}$ has ideal of points $\langle x_1, x_2^2 - x_2 \rangle$. By Theorem

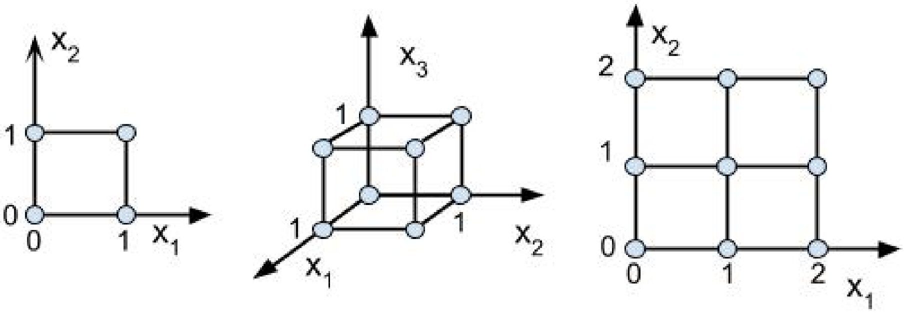


Figure 3.1: The lattice of points in $\mathbb{Z}_2^2$ (left), in $\mathbb{Z}_2^3$ (center), and in $\mathbb{Z}_3^2$ (right).

3.1 we see that the generators of I form a unique reduced GB. Similarly $\{(1,0),(1,1)\}$ has ideal of points $\langle x_1 - 1, x_2^2 - x_2\rangle$, which also has a unique reduced GB. Note that while they have different GBs, they have the same leading term ideal, namely, $\langle x_1, x_2^2\rangle$. In the same way, pairs of points that lie along vertical lines have unique reduced GBs: sets $\{(0,0),(1,0)\}$ and $\{(0,1),(1,1)\}$ have the unique leading term ideal $\langle x_1^2, x_2\rangle$. In each case, these sets have points with one coordinate change.

On the other hand, pairs of points that lie on diagonals have 2 distinct reduced Gröbner bases as such points have two coordinate changes. For example, the set of points $\{(0,0),(1,1)\}$ has GBs $\{\underline{x_1} - x_2, x_2^2 - x_2\}$ and $\{x_1^2 - x_1, \underline{x_2} - x_1\}$ with leading term ideals $\langle x_1, x_2^2\rangle$ and $\langle x_1^2, x_2\rangle$ respectively. Similarly the set $\{(0,1),(1,0)\}$ has $\{\underline{x_1} - x_2 - 1, x_2^2 - x_2\}$ and $\{x_1^2 - x_1, \underline{x_2} - x_1 - 1\}$ as Gröbner bases with leading term ideals $\langle x_1, x_2^2\rangle$ and $\langle x_1^2, x_2\rangle$ respectively. □

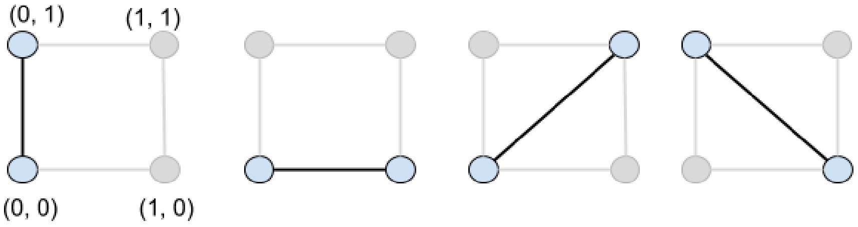


Figure 3.2: Four configurations of pairs of points in $\mathbb{Z}_2^2$. From left to right: $\{(1,0),(0,1)\}$ and $\{(0,0),(1,0)\}$ each have 1 GB, while $\{(0,0),(1,1)\}$ and $\{(1,0),(0,1)\}$ have 2 distinct GBs.

Example 3.2 Now consider two points in $\mathbb{Z}_2^3$. The center graph in Figure 3.1 is the plot of all points in $\mathbb{Z}_2^3$. In Figure 3.3, pairs of points that lie on edges of the 3-cube have 1 reduced Gröbner basis, as the points have one coordinate change: for example, the set $\{(1,0,1),(1,1,1)\}$ (first from the left in Figure 3.3) has the unique reduced GB $\{x_1-1, x_2^2-x_2, x_3-1\}$ and $\{(0,0,0),(0,0,1)\}$ (second) has the unique GB $\{x_1, x_2, x_3^2-x_3\}$. Points that lie on faces of the 3-cube have 2 GBs as they have 2 coordinate changes: the third set $\{(1,1,1),(0,1,0)\}$ in Figure 3.3 has GBs $\{\underline{x_1}-x_3, x_2-1, x_3^2-x_3\}$ and $\{x_1^2-x_1, x_2-1, \underline{x_3}-x_1\}$. Finally points that lie on lines through the interior have 3 GBs as they have 3 coordinate changes: the fourth set $\{(1,0,1),(0,1,0)\}$ has GBs $\{\underline{x_1}-x_3, \underline{x_2}-x_3-1, x_3^2-x_3\}$, $\{\underline{x_1}-x_2-1, x_2^2-x_2, \underline{x_3}-x_2-1\}$, and $\{x_1^2-x_1, \underline{x_2}-x_1-1, \underline{x_3}+x_1\}$. □

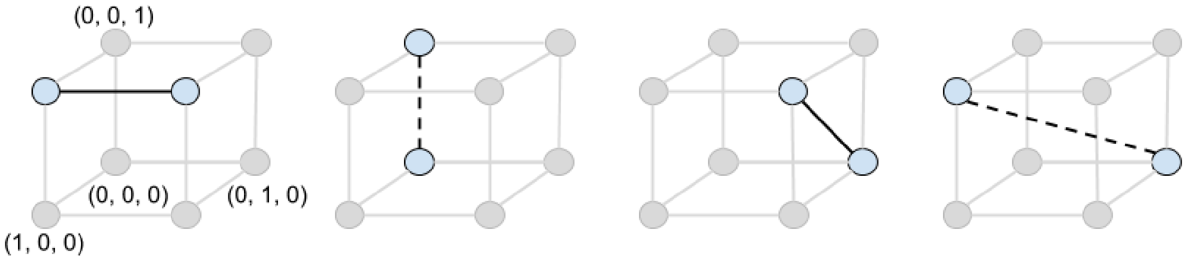


Figure 3.3: Four configurations of pairs of points in $\mathbb{Z}_2^3$. From left to right: $\{(1,0,1),(1,1,1)\}$ and $\{(0,0,0),(0,0,1)\}$ have 1 GB; $\{(1,1,1),(0,1,0)\}$ has 2 GBs; and $\{(1,0,1),(0,1,0)\}$ has 3 GBs.

Next, we consider data over the field $\mathbb{Z}_3$.

Example 3.3 Let $p=3$ and $n=2$. The right graph in Figure 3.1 is the plot of all points in $\mathbb{Z}_3^2$. Similar to the Boolean case in Figure 3.2, pairs of points that lie on horizontal or vertical lines have one associated reduced Gröbner basis for any monomial order, while pairs of points that lie on any skew line have two distinct GBs. For example, the set $\{(0,0),(0,2)\}$ in Figure 3.4 has ideal of points $\langle x_1, x_2^2+x_2\rangle$, which has a unique reduced Gröbner basis via Theorem 3.1. On the other hand, the set of points $\{(1,2),(2,1)\}$ has two GBs, namely $\{\underline{x_1}+x_2, x_2^2+1\}$ and $\{x_1^2-1, \underline{x_2}+x_1\}$ with leading term ideals $\langle x_1, x_2^2\rangle$ and $\langle x_1^2, x_2\rangle$ respectively. □

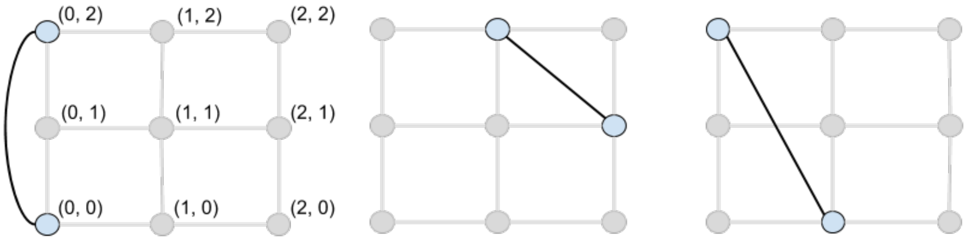


Figure 3.4: Three configurations of points in $\mathbb{Z}_3^2$. From left to right: $\{(0,0),(0,2)\}$ has 1 GB, while $\{(1,2),(2,1)\}$ and $\{(0,2),(1,0)\}$ each have 2 distinct GBs.

In the case of $m = 2$ points, we see data that lie on horizontal or vertical edges have ideals of points with unique Gröbner bases, that is unique models, while data whose coordinates change simultaneously have multiple models associated with them. Though the number n of coordinates impacts the number of resulting models, the field cardinality p does not.

3.2.2. Geometric Characteristics of Three Points

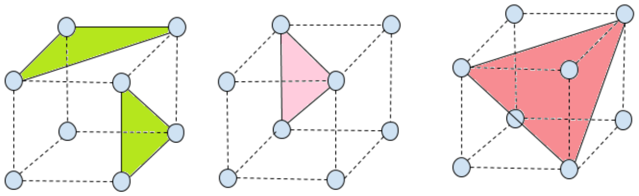


Figure 3.5: Configurations of sets of 3 points in $\mathbb{Z}_2^3$ corresponding to different numbers of GBs. Points that are in configurations similar to the green triangles (left) have a unique reduced Gröbner basis for any monomial order; the pink triangle (center) has two distinct GBs; and the red triangle (right) has three distinct GBs.

The example illustrates that points that lie on faces of the 3-cube have 1 Gröbner basis; points forming a triangle which lies in the interior with two collinear vertices having two distinct GBs, and points in other configurations have 3 GBs.

Now we consider data in $\mathbb{Z}_3$.

Example 3.4 Let $p = 3$ and $n = 2$. By Theorem 3.3, we have that $NGB(3,2,3) \leq 3$. Consider the point configurations in Figure 3.6. The data set corresponding to the green triangle (left) is $S_1 = \{(0,0),(0,1),(1,1)\}$ and has a unique reduced Gröbner basis:

$\{x_2^2 - x_2, x_1x_2 - x_1, x_1^2 - x_1\}$. The data set corresponding to the pink triangle (center) is $S_2 = \{(0,1),(1,2),(2,0)\}$ and has two distinct associated reduced GBs:

$$\{x_2^3 - x_2, \underline{x_1} - x_2 + 1\}, \{-x_1 + \underline{x_2} - 1, x_1^3 - x_1\}.$$

The data set corresponding to the pink triangle (right) is $S_3 = \{(0,0),(1,1),(2,0)\}$ and also has two GBs:

$$\{x_2^3 - x_2, x_1x_2^2 - x_1x_2 + x_2^2 - x_2, x_1^2 - x_1x_2 + x_1 - x_2\}, \{x_2^3 - x_2, -x_1^2 + x_1x_2 - x_1 + x_2, x_1^3 - x_1\}.$$

□

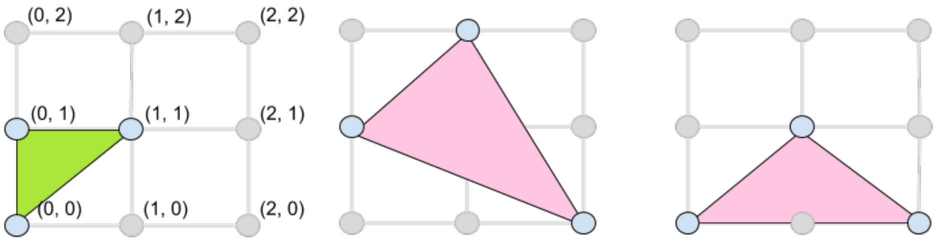


Figure 3.6: Configurations of sets of 3 points in $\mathbb{Z}_3^2$ corresponding to unique and non-unique Gröbner bases. Points that are in configurations similar to the green triangle (left) have a unique reduced Gröbner basis for any monomial order; the pink triangles (center and right) have two distinct GBs.

Using Figure 3.6, we see that 3 points that are collinear or have two adjacent collinear points have unique Gröbner bases, while other configurations result in 2 distinct ones. There are no data sets of 3 points in $\mathbb{Z}_3^2$ that have 3 associated Gröbner bases (data not shown). Therefore the upper bound in Theorem 3.3 is not sharp for $p = 3, n = 2$.

3.2.3. Larger Data Sets

In this section, we offer empirical observations for the number r of distinct reduced Gröbner bases for data sets of m points, where $2 \leq m \leq 6$. Furthermore, we state a conjecture for decreasing r by adding points in so-called linked positions, using the geometric insights from $m = 2, 3$ points.

To generalize the observations from small data sets to larger data sets, we start with configurations of two points and then consider changes in r as points are added. To generalize the geometric pattern from small data sets to larger data sets, we start with configurations of 2 points, then consider changes in the number of Gröbner bases as points are added.

Definition 3.1 Given a set S of points, we say that a point q is in a *linked* position with respect to the points in S if q is adjacent to a point in S and has a minimal sum of distances to the points in S. □

Figure 3.7 shows the changes in the number of Gröbner bases when points are added at either linked or non-linked positions.

Example 3.5 Consider the set $S = \{(0, 1), (1, 2)\}$, which has $r = 2$ Gröbner bases associated to it. We aim to add a point so that the augmented set has $r = 1$. There are four points adjacent to the points in S, namely $(0, 0)$, $(0, 2)$, $(1, 1)$ and $(2, 2)$; see the green points in the top panel of Figure 3.7. The sum of the distances between $(0, 0)$ and the points in S is $\sqrt{5} + 1$; similarly for $(2, 2)$. On the other hand, $(0, 2)$ and $(1, 1)$ both have a distance sum of 2. So $(0, 2)$ and $(1, 1)$ are in linked positions with respect to S. Note that inclusion of either $(0, 2)$ or $(1, 1)$ to S reduces r to 1, while inclusion of either of $(0, 0)$ or $(2, 2)$ keeps $r = 2$. □

Example 3.6 Consider the the set $S = \{(0, 1), (1, 1)\}$, which has a unique Gröbner basis. There are five points adjacent to S, namely $(0, 0)$, $(0, 2)$, $(1, 0)$, $(1, 2)$, and $(2, 1)$; see the green points in the bottom panel of Figure 3.7. The first four points have a distance sum of $\sqrt{2} + 1$, while the last point $(2, 1)$ has a distance sum of 3. So these four points are in linked positions with respect to S and inclusion of any one of them keeps $r = 1$. On the other hand, $(2, 1)$ is not in linked position; nevertheless adding it to S results in a unique Gröbner basis due to it being collinear to the points in S. □

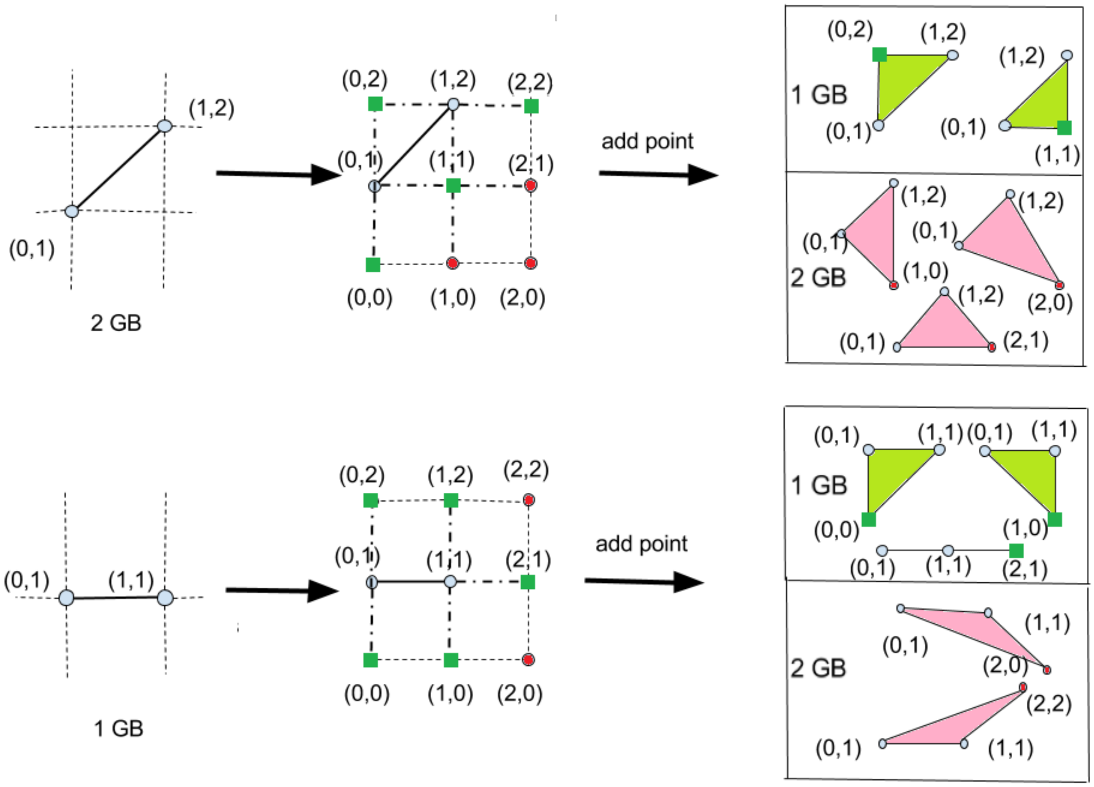


Figure 3.7: The green square points are in linked position with respect to the blue points. Green triangles are associated with unique GB, while pink triangles are associated with non-unique GBs.

Based on the geometric characteristics in the above figures and results, we summarize rules to aid in decreasing the number of candidate models as enumerated by the number of Gröbner bases:

1. *For two points*, fewer changing coordinates in the data points will lead to fewer GBs. In the simplest case, if only one coordinate changes, a unique model will be generated.

2. *For three points*, more points lying on horizontal or vertical edges will reduce the number of GBs. A unique Gröbner basis arises when the data lie on a horizontal line, a vertical line or form a right triangle.

3. *In the process of adding points*, to decrease or keep the number of minimal models, the better candidates of new data points are those in linked positions with respect to an

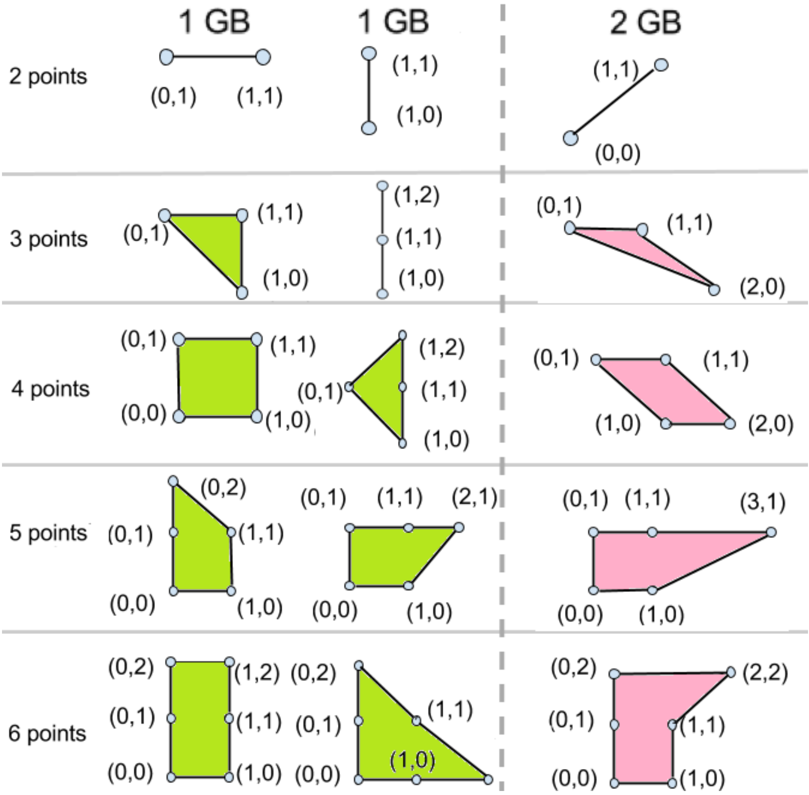


Figure 3.8: Point configurations based on the number of Gröbner bases for $m = 2, \ldots, 6$. The left two columns contain points that form green polygons and correspond to a unique Gröbner basis. The right column contains the pink polygons corresponding to non-unique GBs.

existing data set: this guarantees more points lying on horizontal or vertical edges.

By adding points in linked positions, data sets with multiple Gröbner bases can be transformed into data sets with a unique one.

We end this discussion with a conjecture about linked positions.

Conjecture 3.2.1 *Let S be a set of points, q is a point not in S, and $T = S \cup \{q\}$. If q is in a linked position and the convex hull of the points in T does not contain "holes" (i.e, lattice points not in T), then $\#GB(T) \leq \#GB(S)$.*

3.3. Number of Gröbner Bases in Finite Fields

As we conclude the geometric characterization of input data sets, we know that a specific data set structure is associated with a unique Gröbner basis, and adding the appropriate points to data sets with non-unique Gröbner bases will help to reduce the number of GBs. In this section, we will generate formulas of the number of Gröbner bases given two-point data sets in $\mathbb{Z}_p^n$ and three-point data sets in $\mathbb{Z}_p^n$ based on the discussion of leading term ideals in the finite field.

3.3.1. Formula of Two Points

The following formula is proposed for the number of GBs for data sets of 2 points.

Theorem 3.2 *Let $P = (p_1, \ldots, p_n), Q = (q_1, \ldots, q_n) \subset \mathbb{Z}_p^n$ where $P \neq Q$, and let $I \subset \mathbb{Z}_p[x_1, \ldots, x_n]$ be the ideal of the points P, Q. The number of distinct reduced Gröbner bases for I is given by*

$$NGB(p, n, 2) = \sum_{\substack{p_i \neq q_i \\ i=1,\ldots,n}} 1 \tag{3.1}$$

Proof: Let $V = \{P, Q\} \subset \mathbb{Z}_p^n$ with $P = (p_1, \ldots, p_n), Q = (q_1, \ldots, q_n)$. Let $I \subset \mathbb{Z}_p[x_1, \ldots, x_n]$ be the ideal of the points in V. By Theorem 2.2, the number of elements of any set of standard monomials for I is $|V| = 2$. Since sets of standard monomials must be closed under division, the only option for such a set is $\{1, x_i\}$ for some $i = 1, \ldots, n$. So the possible associated minimally generated leading term ideals are of the form $\langle x_1, \ldots, x_{i-1}, x_i^2, x_{i+1}, \ldots, x_n \rangle$. We consider the number of leading term ideals in regards to the number of coordinate changes between the points.

If P and Q only have one different coordinate, say $p_1 \neq q_1$, then the only possible minimal generating set for the leading term ideal of I is $\{x_1^2, x_2, \ldots, x_n\}$. If P, Q have two different coordinates, say $p_i \neq q_i$ for $i = 1, 2$, then the possible minimal generating sets for the leading term ideal of I are $\{x_1^2, x_2, \ldots, x_n\}$ when $x_1 \prec x_2$ and $\{x_1, x_2^2, x_3, \ldots, x_n\}$ when $x_2 \prec x_1$. Increasing the number of coordinate changes will add another leading term ideal. In general, if $p_i \neq q_i$ for $i = 1, \ldots, k$ where $k \leq n$, then the possible minimal generating sets for the leading term ideal of I are as follows:

1. $\{x_1^2, x_2, \ldots, x_n\}$ when x_1 is the smallest variable in the monomial order among $x_1, \ldots, x_k$
2. $\{x_1, x_2^2, x_3, \ldots, x_n\}$ when x_2 is smallest among $x_1, \ldots, x_k$

   $\vdots$

k. $\{x_1, \ldots, x_{k-1}, x_k^2, x_{k+1}, \ldots, x_n\}$ when x_k is smallest among $x_1, \ldots, x_k$.

□

Lemma 3.1 *The maximum number of distinct reduced Gröbner bases for an ideal of two points in $\mathbb{Z}_p^n$ is $NGB(p, n, 2) \leq n$.*

With different choices of smallest coordinate, there are up to n different sets of standard monomials, each corresponding to a distinct reduced Gröbner basis. So, there are up to n reduced Gröbner bases, with the maximum achieved by two points with no coordinates in common.

In applications, modelling is often driven by data. So geometric descriptions of data sets can reveal essential features in the underlying network. We illustrate the above results by considering different configurations of points. We begin with data over $\mathbb{Z}_2$.

3.3.2. Formula for Three Points

Theorem 3.3 *The number of distinct reduced Gröbner bases for ideals of 3 points in $\mathbb{Z}_p^n$ is*

$$NGB(p, n, 3) \leq \begin{cases} \frac{n(n-1)}{2} & \text{for } p = 2 \\ \\ \frac{n(n+1)}{2} & \text{for } p \geq 3. \end{cases} \tag{3.2}$$

Proof: We begin by considering the Boolean base field. By Theorem 2.2, the form of a set of standard monomials for an ideal of 3 points is $\{1, x_i, x_j\}$ for $x_i \neq x_j$. Considering the choice of x_i and x_j, there are up to $\frac{n(n-1)}{2}$ different standard monomial sets, each corresponding to a distinct reduced Gröbner basis by Theorem 3.5.

For a base field with $p > 2$, the two possible forms of standard monomial sets are $\{1, x_i, x_j\}$ for $x_i \neq x_j$, and $\{1, x_i, x_i^2\}$. As we showed above, there are up to $\frac{n(n-1)}{2}$ distinct reduced Gröbner bases corresponding to $\{1, x_i, x_j\}$. Further, the maximum number for the standard monomial form $\{1, x_i, x_i^2\}$ is n. As the two standard monomial forms can both be associated to the same data set, the upper bound for a non-Boolean field is $\frac{n(n-1)}{2} + n = \frac{n(n+1)}{2}$.
□

Example 3.7 Let $p = 2$ and $n = 2$. Then $NGB(2, 2, 3) \leq 1$; that is, all ideals of 3 points in $\mathbb{Z}_2^2$ have a unique reduced Gröbner basis, which is corroborated by Theorem 3.3 and the fact that ideals of a single point have only one distinct Gröbner basis for any monomial order.
□

Unlike the bound for 2 points, there are cases of sets of 3 points for which the upper bound is not sharp. For example, when $n = 4$, the upper bound is $NGB(2, 4, 3) \leq 6$; however, the maximum number is 5, which we tested exhaustively (data not shown).

3.3.3. Upper Bound for the Number of Gröbner Bases

The following results provide an upper bound for the number of reduced Gröbner bases for an ideal over any field. In [5, 6, 35], the upper bound of the number of Gröbner bases is not applicable for a large number of coordinates with respect to the computational cost estimation. In this case, a tighter upper bound should be provided before GRN model selection and experimental design.

Lemma 3.2 ([5]) *The number of vertices of a lattice polytope $P \subset \mathbb{R}^n$ is $\#vert(P) = O\left(vol(P)^{(n-1)/(n+1)}\right)$.*

Theorem 3.4 ([6, 35]) *Let I be an ideal such that $dim_K R/I = m$. Let $\Lambda(I)$ be the set of standard monomial sets for I over all monomial orders. Then the number of distinct reduced Gröbner bases of I is in bijection with the number of vertices of the staircase polytope of I; that is, $\#GBs = O\left(m^{2n\frac{n-1}{n+1}}\right)$.*

Now we summarize the bijective correspondences for the number of reduced Gröbner bases for an ideal of points.

Theorem 3.5 *Let I be an ideal. There is a one-to-one correspondence among the following:*

1. *distinct reduced Gröbner bases of I*
2. *leading term ideals of I*
3. *sets of standard monomials for I*
4. *vertices of the staircase polytope of I.*

Proof: Equivalence $1 \iff 2$ is a result in [11]; $2 \iff 3$ is by construction of standard monomials; and $1 \iff 4$ was proved in [35] for ideals of points and in [6] for other zero-dimensional ideals. □

As the authors in [35] showed that the sum of the coordinates of a staircase of m points (see the left panel in Figure 3.9) corresponds to a vertex of a certain polytope, we must count the number of ways to place m points on the lattice.

Suppose r blue points have been placed (see Figure 3.10). We wish to count the number of ways to place the next point. The red point violates the staircase property. The only choice is green or black point. Note that the black point maximizes the sum of the coordinates.

We now focus on the general setting of subsets of any size m in $\mathbb{Z}_p^n$, for any p and any n.

In Theorem 3.4, the stated upper bound for the number of Gröbner bases for an ideal I of m points in K^n is $m^{2n\frac{n-1}{n+1}}$, where K is any field; furthermore the number of Gröbner bases coincides with the number of vertices of the staircase polytope of I. When the base field is finite, however, this bound becomes unnecessarily large for even small m. Unlike in characteristic-0 fields, all coordinates in positive-characteristic fields are bounded above by p; for example, see Figure 3.9. We will use the fact that staircases in a finite field are contained in a hypercube of volume p^n to modify the bound. The only part of the construction of the staircase polytope that is affected by the characteristic is the maximum value of any vertex. As a vertex is a vector sum $\sum \lambda$ of points in a staircase λ, the modification comes from placing staircase points aimed to maximize the sum.

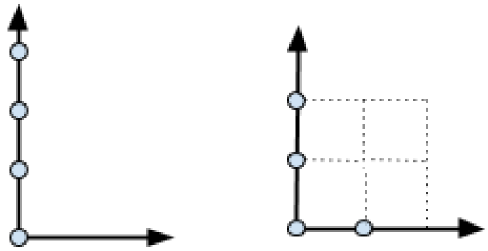


Figure 3.9: The staircase $\lambda \subset \mathbb{R}^2$ (left) has $\sum \lambda = (0,6)$ while the staircase $\lambda \subset \mathbb{Z}_3^2$ (right) has $\sum \lambda = (1,3)$.

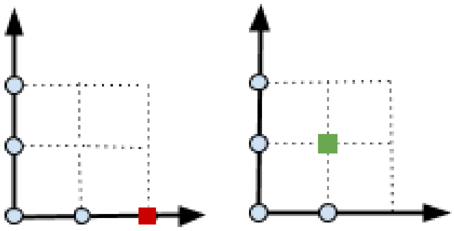


Figure 3.10: The staircase $\lambda \subset \mathbb{Z}_3^2$ with a red square point (left) has $\sum \lambda = (3,3)$ while the staircase $\lambda \subset \mathbb{Z}_3^2$ with a green square point (right) has $\sum \lambda = (2,4)$.

Consider any staircase λ of 5 elements. In the following discussion, we will consider the placement of points so that the vector sum is maximized. We proceed in a "greedy" manner by maximizing a fixed coordinate. Suppose four (blue) points have already been placed so as to maximize the value of the second coordinate of $\sum \lambda$; see Figure 3.10. Placing the green point $(1,1)$ contributes 1 to the running sum, that is, $\sum_{j=1}^{m} \lambda_{j2} = 4$, while placing the redpoint $(2,0)$ keeps the sum of the coordinate unchanged. In fact, to maximize the sum of the second coordinate, choose any point whose second coordinate is largest among the available positions, so that the configuration continues to be a staircase. For the k^{th} point, the coordinate of e_j is $(k-1) \bmod p$.

$$M = \left(\frac{p(p-1)}{2} \lfloor m/p \rfloor + \frac{(m \bmod p)(m \bmod p - 1)}{2} \right).$$

Next we state a result about data sets and their complements.

Lemma 3.3 ([14]) *Let I be the ideal of input points S, and let I^c be the ideal of the complement $\mathbb{F}^n \setminus S$ of S. Then we have $SM_{\prec}(I) = SM_{\prec}(I^c)$ and $LT_{\prec}(I) = LT_{\prec}(I^c)$ for a given monomial order $\prec$. Hence, we have $\#GB(S) = \#GB(\mathbb{F}^n \setminus S)$.*

Theorem 3.6 *The number of distinct reduced Gröbner bases for an ideal of m points in $\mathbb{Z}_p^n$ is*

$$NGB(p,n,m) = \begin{cases} O\left((p^2\lfloor m/p\rfloor + (m \bmod p)^2)^{\frac{n-1}{n+1}}\right) & : 0 < m \leq \lfloor p^n/2\rfloor \\ O\left((p^2\lfloor (p^n-m)/p\rfloor + (-m \bmod p)^2)^{n\frac{n-1}{n+1}}\right) & : \lfloor p^n/2\rfloor \leq m < p^n \\ 1 & : m = 0, p^n. \end{cases} \tag{3.3}$$

Proof: Let I be an ideal of m points in $\mathbb{Z}_p^n$. Recall that the number of Gröbner bases of I is bijective with the number of vertices of the staircase polytope $\mathcal{P}$ of I by Theorem 3.5. The cases $m = 0, p^n$ are trivial. So we proceed with $0 < m \leq \lfloor p^n/2\rfloor$.

As $\mathcal{P}$ is the convex hull of the points $\sum\lambda$ where λ is a staircase corresponding to the exponent vectors of the standard monomial sets of I, we will show that the staircase polytope of I is contained in a larger convex body whose volume we can compute easily.

Let $\lambda = \{\lambda_1, \ldots, \lambda_m\}$. Then $\sum\lambda = \sum_{i=1}^m \lambda_i = \sum_{i=1}^m \left(\sum_{j=1}^m \lambda_{ji}\right) e_i$ where λ_{ji} denotes the i-th coordinate of the j-th point and e_i is the standard basis vector. Note that the maximum sum of the i-th coordinate is

$$\begin{aligned} \max\sum_{j=1}^m \lambda_{ji} &= \underbrace{(1+\ldots+p-1)\lfloor m/p\rfloor}_{p\lfloor m/p\rfloor \text{ points}} + \underbrace{(1+\ldots+m \bmod p-1)}_{\text{remaining } m \bmod p \text{ points}} \\ &= \frac{p(p-1)}{2}\lfloor m/p\rfloor + \frac{(m \bmod p)(m \bmod p-1)}{2} \end{aligned}$$

which we denoted by M. So the staircase polytope $\mathcal{P} \subset \mathbb{R}^n$ is contained in the hypercube $[0, M]^n$, which has volume M^n. Therefore $vol(\mathcal{P}) \leq M^n$. By Lemma 3.2 and Theorem 3.4, we have that

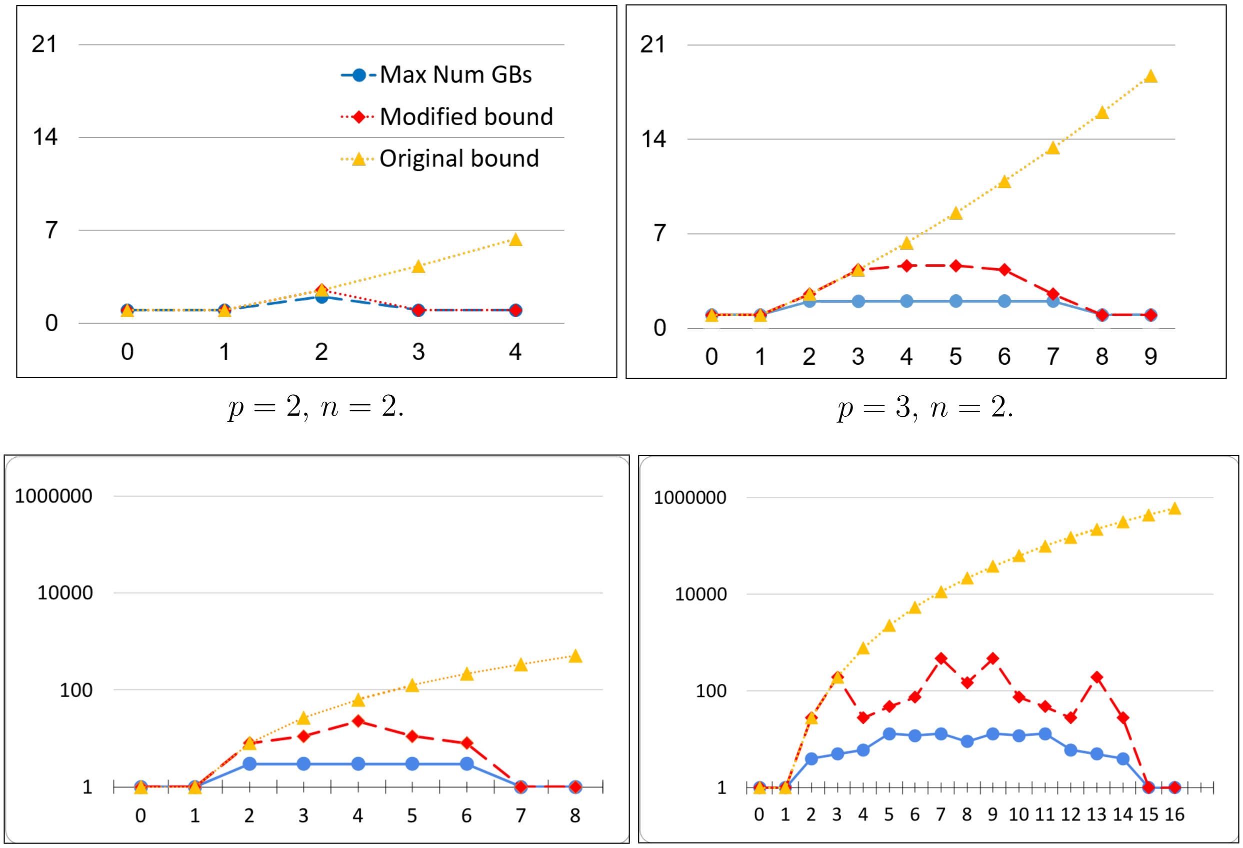
$$\begin{aligned} NGB(p,n,m) &= O\left(vol(\mathcal{P})^{(n-1)/(n+1)}\right) \\ &= O\left((M^n)^{\frac{n-1}{n+1}}\right) \\ &= O\left(\left(p^2\lfloor m/p\rfloor + (m \bmod p)^2\right)^{n\frac{n-1}{n+1}}\right). \end{aligned} \tag{3.4}$$

For the final case when $m \geq \lfloor p^n/2 \rfloor$, the number of Gröbner bases can be computed by plugging $p^n - m$ into the second argument of the above bound, according to Theorem 3.3. □

It is straightforward to show that our bound grows much slower than $O\left(m^{2n\frac{n-1}{n+1}}\right)$ reported in [35], which we have also verified computationally. In Appendix A, there is a table of selected numerical results of the new upper bound in comparison to the values of the original upper bound in [35]. Figure 3.11 provides a comparison for selected cases among $p = 2, 3$ and $n = 2, 3, 4$.

The values from Equation 3.3 are closer to the actual number of GBs according to Theorem 3.3, which makes our bound retain the symmetric nature of the maximum number of Gröbner bases for ideals of points in $\mathbb{Z}_p^n$. For example, for $p = 2$, $n = 4$, and $m = 5$ in Figure 3.11, the original bound is over 2000, while the modified bound is in the same order of magnitude as the actual maximum number of GBs.

The significance of this result is that Equation 3.3 provides a more accurate representation of the maximum number of models associated to a data sets, which may aid in experimental design. It is straightforward to show that our bound grows much slower than the bound $O\left(m^{2n\frac{n-1}{n+1}}\right)$ reported in [35], which we have also verified computationally. Below is a table of selected numerical results of the new upper bound in comparison to the values of the original upper bound in [35].



$p = 2,\ n = 2.$

$p = 3,\ n = 2.$

$p = 2,\ n = 3.$

$p = 2,\ n = 4.$

Figure 3.11: Plots comparing the maximum number of Gröbner bases. The caption in each plot indicates the values of p and n for $\mathbb{Z}_p^n$. In each case, all subsets of size m are computed, where $m = 0 \ldots p^n$, and listed on the horizontal axis. The vertical axis is the maximum number of GBs for a set of size m. The blue solid line with dots shows the actual maximum number of GBs. The yellow dotted line with triangles is the original upper bound given by Theorem 3.4, where the red dashed line with squares is the modified upper bound given by Equation 3.3.

# of points	max # of GBs	original bound	modified bound
1	1	1	1
2	4	27.86	27.86
3	5	195.07	47.59
4	6	776.05	147.03
5	13	2264.94	195.07
6	12	5434.08	389.08
7	13	11388.61	471.48
8	9	21618.82	389.08

Table 3.1: For cases $p = 2$, $n = 4$ and $m = 1, \ldots, 8$, we compare the actual maximum number of GBs with the original bound [35] and the modified bound in Theorem 3.6.

Chapter 4

MODEL SELECTION

The work of ECs theorems in this Chapter is under the publish preparation.

4.1. Introduction

As the model space can be large [28], this creates ambiguity in predictions. We can eliminate ambiguity by reducing the number of GBs. While our ultimate goal is to identify data sets with a unique GB [14], a new method of deleting redundant information is motivated by Example 2.12. By checking if an unknown data set is the linear shift of a known data set, the information of the model type and standard monomial basis will be found without additional computational cost. For discovering the relationship between data sets, we started with a linear shift, which was first defined in [26]. Considering the model selection process in [4], it is essential to reduce the search space of numerous input data sets that are associated with different distinct network models.

The following contributions are made in the process. First, all possible data sets are calculated with a fixed number of variables (n) and the number of states (p). As theorems and properties of the relationship between data set were proved in previous work [26, 39, 14] and this dissertation, all equivalence classes (EC) are generated by applying linear shifts to data sets using Python. Two important properties were found:

- relationship between data set and complement data set
- relationship between the new data set and original data set by adding points, especially to help to reduce the number of GBs.

Then, to better characterize and compare different equivalence classes to enable the model selection in each EC, two new definitions are created:

- Data set distance based on the Euclidean distance of data set
- Representative of each equivalence class based on the data set distance defined.

Defining a representative is essential for model selection as it reduces the search space to a few data sets instead of the numerous data sets in $\binom{\mathbb{F}^n}{m}$ without picking up representatives. Furthermore, the concise comparison, that uses representative data sets of different models can help researchers get the most valuable predictive model.

Lastly, a specific geometric configuration called a staircase, which is associated with unique Gröbner bases, was studied with algebraic geometry methods. An important finding to help model selection is that if a staircase exists in one EC, it is a representative and the only staircase in that EC. We can directly pick a staircase as a representative and generate other data sets in the same EC by linear shift. Then instead of calculating GBs one by one for all of the data sets in the EC, the other GBs can be generated, applying linear shift mapping function to the GB of the representative. All the data sets in this EC are associated with unique GB.

Note that linear shift is a bijection and we prove that it is an equivalence relation. The properties and definitions were implemented into the Python package in Appendix A. This package contains many functions such as making partitions of data sets and finding a set's EC and incorporates parallel computing.

4.2. Model Selection with Complement Data Sets

In the process of model selection, with limitation of computing resources, it is a better choice to compute complement data sets, if original data sets have too many points. The next theorem shows that if two sets are in the same equivalence class, then so are their complements.

Theorem 4.1 *If $S_1 \overset{\psi}{\sim} S_2$, then there exists a linear shift ϕ such that the complement data set of S_i can linearly shift the complement data set of S_j, $S_1^c \overset{\phi}{\sim} S_2^c$.*

Proof: Recall that linear shifts are bijective, which means all linear shift functions are one-to-one mappings from $\mathbb{F}^n$ onto $\mathbb{F}^n$.

Suppose $S_1 \overset{\psi}{\sim} S_2$ and $|S_1| = |S_2| = m$. As $\psi : S_1 \to S_2$ is a bijection and $S_1, S_2 \subseteq \mathbb{F}^n$, ψ can be extended as a bijection on $\mathbb{F}^n$. As S_1^c, S_2^c are also contained in $\mathbb{F}^n$, the restriction of ψ to S_1^c is a bijection from S_1^c onto S_2^c. Hence $S_1^c \overset{\psi|_{S_1^c}}{\sim} S_2^c$ and thus S_1^c and S_2^c are in the same equivalence class, $\phi = \psi|_{S_1^c}$. □

Theorem 4.2 *Let $\overline{S} = \{S_1, \ldots S_r\}$ be an equivalence class of sets of m points. Then $\{S_1^c, \ldots S_r^c\}$ is an equivalence class of sets of $p^n - m$ points, denoted $\overline{S^c}$.*

Proof: Suppose $\overline{S} = \{S_1, \ldots S_r\}$ is an equivalence class. By construction, two properties hold: 1) for all $S_i, S_j \in \overline{S}$, we have that $S_i \overset{\phi}{\sim} S_j$; and 2) if $S_i \in \overline{S}$ and $S_i \overset{\phi}{\sim} T$, then $T \in \overline{S}$.

We show that 1) and 2) hold for $\overline{S^c} = \{S_1^c, \ldots S_r^c\}$.

1) Let $S_i^c \neq S_j^c \in \{S_1^c, \ldots, S_r^c\}$. We know $S_i, S_j \in \overline{S}$, and so $S_i \sim S_j$. By Theorem 4.1, $S_j^c \sim S_j^c$.

2) If $S_i^c \notin \binom{\mathbb{F}^n}{m}$, but there is some $S_j^c \sim S_i^c$, it means that $S_i \sim S_j$ by Theorem 4.1. Then as $\overline{S}$ is an equivalence class, $S_j \in \overline{S}$. So $S_i \sim S_k, \forall S_k \in \overline{S}$. By construction of $\overline{S}$ and $\overline{S^c}$, for $S_i \in \overline{S}$, then $S_i^c \in \overline{S^c}$. □

We get the following results.

Theorem 4.3 *Let $\overline{S}$ be an equivalence class for a fixed number of points. Then $|\overline{S}| = |\overline{S^c}|$.*

Proof: Suppose $|\overline{S}| > |\overline{S^c}|$. Then there exists $S_i \in \overline{S}$ such that its complement set $S_i^c \notin \overline{S^c}$. It follows that $S_i^c \not\sim S_j^c, \forall S_j^c \in \overline{S^c}$. For $S_i \sim S_j, \forall S_j \in \overline{S}$ and based on Theorem 4.1, we know that $S_i^c \sim S_j^c, \forall S_j^c \in \overline{S^c}$. So $|\overline{S}| \not> |\overline{S^c}|$. Similarly we find that $|\overline{S}| \not< |\overline{S^c}|$. Hence, $|\overline{S}| = |\overline{S^c}|$.
□

Theorem 4.3 indicates that the number of data sets in an equivalence class and the complement equivalence class is the same. Let us step back to the *set* of ECs: will the number of all ECs and number of all complement ECs be the same?

Theorem 4.4 *Let $\mathcal{S} = \{\overline{S_1}, \ldots, \overline{S_{k_1}}\}$ be the collection of equivalence classes for sets with m points, and $\mathcal{T} = \{\overline{T_1}, \ldots, \overline{T_{k_2}}\}$ the collection of equivalence classes for sets with $p^n - m$ points. Then $k_1 = k_2$.*

Proof: Suppose $\mathcal{S} = \{\overline{S_1}, \ldots, \overline{S_{k_1}}\}$ is the collection of equivalence classes for sets with m points, and $\mathcal{T} = \{\overline{T_1}, \ldots, \overline{T_{k_2}}\}$ is the collection of equivalence classes for sets with $p^n - m$ points. The total number of subsets of size m is given by $\binom{p^n}{m}$. As $\mathcal{S}$ partitions the collection of all subsets of size m,

$$\binom{p^n}{m} = |\overline{S_1}| + \ldots + |\overline{S_{k_1}}|.$$

Likewise

$$\binom{p^n}{p^n - m} = |\overline{T_1}| + \ldots + |\overline{T_{k_2}}|.$$

From Theorems 4.2 and 4.3, we know that $\overline{S_i^C}$ is an equivalence class of sets of size $p^n - m$ and $|\overline{S_i}| = |\overline{S_i^C}|$ for each i. So $\overline{S_i^C} = \overline{T_j}$ for some index j and $|\overline{S_i}| = |\overline{T_j}|$. As $\binom{p^n}{m} = \binom{p^n}{p^n-m}$, then it follows that $k_1 = k_2$.

□

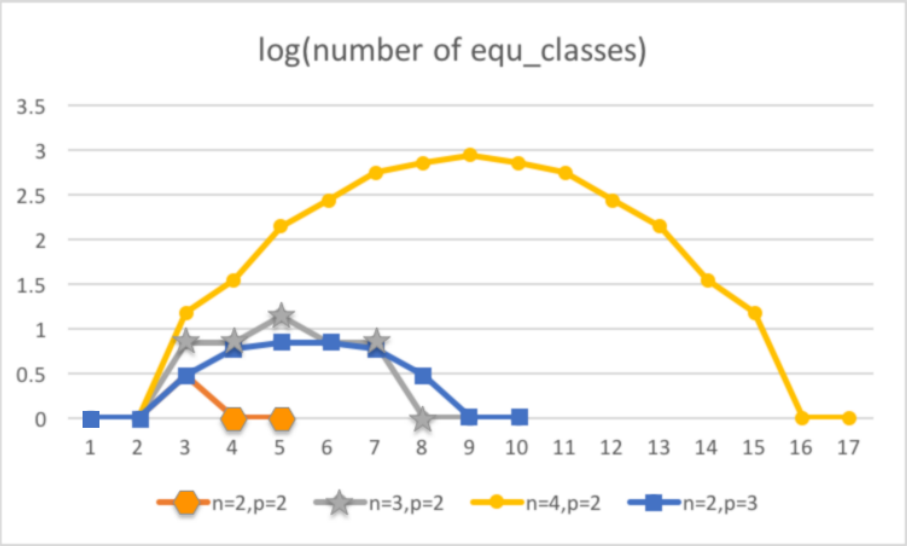
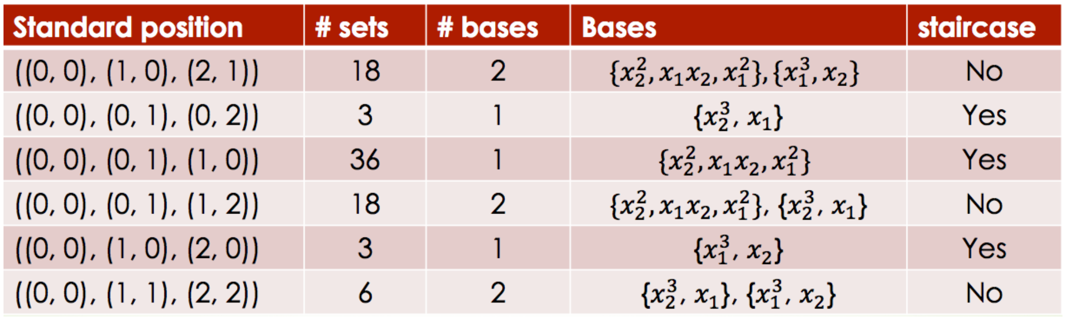


Figure 4.1: The number of equivalence classes for a fixed number of points. The x-axis is the number m of points for 4 combinations of p and n, and the y-axis is the log of the number of ECs for a data set with m points.

The last theorem says that the number of equivalence classes is symmetric with respect to increasing set size (number of points). From Figure 4.1, we know in four cases, the data set and its complement data set will have the same number of ECs. In [14] it was shown that a set and its complement have the same number of leading term ideals and the standard monomial basis for $I(S_i)$ with respect to a fixed monomial order $\prec$ is the "complement" of the leading term ideal for $I(S_i^C)$. So the two sets have the same number of standard monomial bases and the number of equivalence classes for sets and their complements correspond.

4.3. Model Selection by Representatives

From the properties of the linear shift in Section 4.1, the data sets can be partitioned to different equivalence classes. In each equivalence class, implementing model selection using representatives is meaningful as representatives are associated with the most concise model bases. As shown in Figure 4.2, we change the search of all sets in the finite field, which contains $|\binom{\mathbb{F}^2}{3}| = 84$ data sets, to 6 representatives.



Standard position	# sets	# bases	Bases	staircase
((0, 0), (1, 0), (2, 1))	18	2	$\{x_2^2, x_1x_2, x_1^2\}, \{x_1^3, x_2\}$	No
((0, 0), (0, 1), (0, 2))	3	1	$\{x_2^3, x_1\}$	Yes
((0, 0), (0, 1), (1, 0))	36	1	$\{x_2^2, x_1x_2, x_1^2\}$	Yes
((0, 0), (0, 1), (1, 2))	18	2	$\{x_2^2, x_1x_2, x_1^2\}, \{x_2^3, x_1\}$	No
((0, 0), (1, 0), (2, 0))	3	1	$\{x_1^3, x_2\}$	Yes
((0, 0), (1, 1), (2, 2))	6	2	$\{x_2^3, x_1\}, \{x_1^3, x_2\}$	No

Figure 4.2: Summary of data sets partition with three states, two coordinates, three points.

Definition 4.1 *Let $S \subset \mathbb{F}^n$. Then the* set distance *of S, denoted $D(S, 0)$, is the sum of the Euclidean distances of all points in S to the origin.*

Definition 4.2 *The data set S_R is a* representative *for its equivalence class if S_R has a minimum set distance.*

Example 4.1 For $p = 2$, $n = 3$ and $m = 3$, there are 8 data sets associate with the same standard monomial set $SM_1 = \{\{1, x_3, x_2\}\}$. Based on Definition 4.2, the data set, which has the smallest set distance, is $S_1 = \{(0, 0, 0), (0, 0, 1), (0, 1, 0)\}$. S_1 is a linear shift of other 7 data sets associate with SM_1. For example, $S_2 = \{(0, 0, 0), (0, 0, 1), (0, 1, 1)\}$ can linearly shift to S_1 with $\Phi = (x, x, x + 1)$. □

From Definitions 4.1 and 4.2, we first set p to be a large prime number and fix the number of points m to 2 and let the number of variables change from $n = 2$ to k to get the properties of the representative with increasing number of variables. Our discussion below will show how to construct representatives and count number of representatives for the case of two points.

Formula 4.1 *For any 2 points* $\{(x^i_{11}, \dots, x^i_{n1}), (x^i_{12}, \dots, x^i_{n2})\} \subset \mathbb{F}^2$ *with* n *variables and number of statesn* $p \geq 2$*, the total number of representatives is*

$$\#Representatives = 2^n - 1.$$

Proof:

1. When $n = 1$, the representative, which has smallest set distance, is $S^R = \{(0), (1)\}$. So the number of representatives is 1.

2. When $n = 2$, we should consider the representative $S^R = \{P^R_1, P^R_2\}$ that is a linear shift from data set $S_i = \{(x^i_{11}, x^i_{21}), (x^i_{12}, x^i_{22})\} \subseteq \mathbb{F}^n$ and has minimal set distance. Let us start with constructing mapping functions of the origin point $(0, 0) = P^R_1$ (minimal distance point) to one point $(x^i_{11}, x^i_{21}) \in S_i$:

$$\phi_k(0) = a_k \cdot 0 + b_k = b_k = x^i_{k1}, \forall k \in \{1, 2\}$$

So, we get the value of one parameter in mapping functions, $b_1 = x^i_{11}$ and $b_2 = x^i_{12}$. We denote the other point in the representative set as $P^R_2 = (x^R_{12}, x^R_{22})$. Then (x^i_{12}, x^i_{22})

is a linear shift from the representative's point P_2^R:

$$\phi_k(x_{k2}^R) = a_k x_{k2}^R + b_k = a_k x_{k2}^R + x_{k1}^i = x_{k2}^i, \forall k \in \{1, 2\}.$$

Then by substituting x_{k1}^i for b_k from first step results, we can get the expression of a_k with given data set:

$$a_k x_{k2}^R = x_{k2}^i - x_{k1}^i, \forall k \in \{1, 2\}.$$

From the definition of linear shift, we know that $a_1 \neq 0, a_2 \neq 0$. So three cases exist:

(a) $x_{12}^i = x_{11}^i$ and $x_{22}^i \neq x_{21}^i$: we will have $x_{12}^R = 0$ and $x_{22}^R \neq 0$ for $a_1 \neq 0, a_2 \neq 0$. Hence the data set can only be shifted to the representative $S_R = \{(0,0),(0,1)\}$, as $(0,1)$ has the smallest distance.

(b) $x_{12}^i \neq x_{11}^i$ and $x_{22}^i = x_{21}$: we will have $x_{12}^R \neq 0$ and $x_{22}^R = 0$ for $a_1 \neq 0, a_2 \neq 0$. Hence $S_R = \{(0,0),(1,0)\}$, as $(1,0)$ has the smallest distance.

(c) $x_{22}^i \neq x_{21}^i$ and $x_{12}^i \neq x_{11}^i$: we will have $x_{12}^R \neq 0$ and $x_{22}^R \neq 0$ for $a_1 \neq 0, a_2 \neq 0$. Hence $S_R = \{(0,0),(1,1)\}$, as $(1,1)$ has the smallest distance.

So, for any data sets with 2 coordinates, it can linearly shift to one of the representatives: $\{(0,0),(0,1)\}, \{(0,0),(1,0)\}, \{(0,0),(1,1)\}$. We now know the number of representatives is 3.

3. When $n = 3$, for 2 points $S_i = \{(x_{11}^i, x_{21}^i, x_{31}^i), (x_{12}^i, x_{22}^i, x_{32}^i)\}$ with large enough number of states p, do the same process as with 2 variables: first, shift the origin $(x_{11}^R, x_{21}^R, x_{31}^R) = (0,0,0) \in S_R$ to $(x_{11}^i, x_{21}^i, x_{31}^i)$

$$\phi_k(0) = a_k \cdot 0 + b_k = b_k = x_{k1}^i, \forall k \in \{1, 2, 3\}.$$

So we know the values of b_1, b_2 and b_3. Then apply the following linear shift mapping functions to $(x_{12}^i, x_{22}^i, x_{32}^i)$:

$$\phi_k(x_{k2}^R) = a_k x_{k2}^R + b_k = a_k x_{k2}^R + x_{k1}^i = x_{k2}^i, \forall k \in \{1, 2, 3\}$$

We get equations for the representative point: $a_k x^R_{k2} = x^i_{k2} - x^i_{k1}, \forall k \in \{1,2,3\}$. The total number of representatives is the number of combinations of each coordinate's equality of two points in the given data set. More specifically, if $x^i_{k1} \neq x^i_{k2}$, $x^R_{k2} = 1$ then $a_k = x^i_{k2} - x^i_{k1}$. If $x^i_{k1} = x^i_{k2}$, $x^R_{k2} = 0$ then a_k can be any nonzero number. Considering all combinations for n coordinates, the maximum number of equal coordinates of given data sets is $n-1$ since we cannot make all coordinates equal for two different points. For each coordinate the representative value can be 0 or 1 and we should eliminate the case $(0,0,0)$ as it's already a point in S^R. Then the number of all possible representatives is $2^3 - 1$. The representative data sets are $\{(0,0,0),(1,0,0)\}$, $\{(0,0,0),(0,1,0)\}$, $\{(0,0,0),(0,0,1)\}$, $\{(0,0,0),(1,1,0)\}$, $\{(0,0,0),(1,0,1)\}$, $\{(0,0,0),(0,1,1)\}$, $\{(0,0,0),(1,1,1)\}$.

4. When $n > 3$, with the same process as for $n = 3$, the number of representatives $2^n - 1$ can be counted by considering all combinations of coordinate values to be 0 or 1. We can always construct a mapping function from the origin $P^R_1 = (0,\ldots,0)$ to one point in data set $S^i \in \mathbb{F}$ with the mapping functions parameters:

$$b_k = x^i_{k1}$$

$$a_k = \begin{cases} 0 & \text{when } x^i_{k1} = x^i_{k2} \\ x^i_{k2} - x^i_{k1} & \text{when } x^i_{k1} \neq x^i_{k2}. \end{cases} \tag{4.1}$$

Here, $k = 1,\ldots,n$. For $P^R_2 = (P^R_{11},\ldots,P^R_{n2})$, $P^R_{k2} = 1$ if $x^i_{k1} \neq x^i_{k2}$, else $P^R_{k2} = 0$ for $k = 1,\ldots,n$.

□

From the above discussion of choosing representatives for different data sets with varying numbers of variables, we know in each case for n, m, p, a data set will be a (linear shift of a) representative. These results motivated us to distinguish different representatives not only from the input data set's structure (as the previous discussion) but also from the monomial basis and leading terms of the inputs. As researchers discussed before (see [14]), a staircase is an essential structure as regards to standard monomials bases, which will always return a unique Gröbner basis.

4.4. Model Selection by Staircases

As we already defined the representative in each EC, the relationship between representative and unique data structure: the staircase was studied to enable the model selection on unique GB only by picking staircase representatives.

Theorem 4.5 **([26])** *If S_1, $S_2 \subseteq \mathbb{F}^n$ are both staircases and $S_1 \sim S_2$, then $S_1 = S_2$.*

Example 4.2 As staircases are all associated with unique GBs, are they linear shifts of each other? No. In Figure 4.3, where $n = 2$, $m = 3$ and $p = 3$, the staircase with black points $\{(0,0),(0,1),(0,2)\}$ on the left cannot be shifted to the staircase with black points $\{(0,0),(1,0),(2,0)\}$ on the right, for there is no map for the first coordinate with single value 0 on the left to the three different values of the first coordinate in the data set on the right plot. □

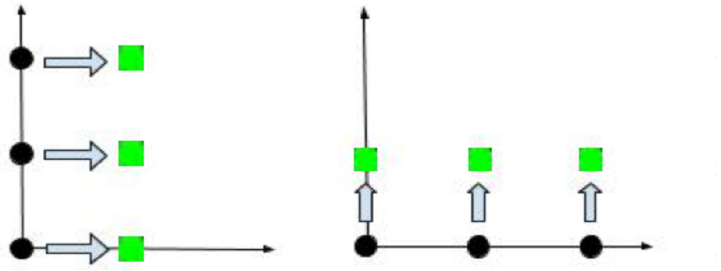


Figure 4.3: The black points $\{(0,0),(0,1),(0,2)\}$ on the left cannot be shifted to the black points $\{(0,0),(1,0),(2,0)\}$ on the right.

Lemma 4.1 *If $S_1 \neq S_2$ are two staircases in $\binom{\mathbb{F}^n}{m}$, then for any term order $LT_{\prec}(I(S_1)) \neq LT_{\prec}(I(S_2))$ and they have different standard monomial bases. There is no mapping function Φ that can linear shift S_1 to S_2 or S_2 to S_1.*

Proof: If there exists two staircases S_i, S_j with same standard monomials and same leading terms, then there exists a point $u \in S_i$ and $u \notin S_j$. There are two cases:

1. $u > P, \forall P \in S_j$. Suppose the number of points in S_i is $|S_i| \geq |S_l| + 1$, where $S_l = \{w \in S_i : w < u\}$. As S_i is a staircase, S_l will contain all the points below u. But for $u > P, \forall P \in S_j$, we have that $|S_j| \leq |S_l| < |S_i|$ So, $|S_i| \neq |S_j|$, contradicting the assumption that S_i and S_j have the same number of points.

2. $u \leq P, \exists P \in S_j$. If $u \notin S_j$, then S_j is not a staircase.

As both cases lead to contradictions, different staircases have different leading terms and standard monomials. By the proof of case 1 and case 2, S_1 and S_2 have different standard monomial bases and $LT_{\prec}(I(S_1)) \neq LT_{\prec}(I(S_2))$ as S_1 and S_2 are different staircases. But if $S_1 \sim S_2$ (there exists a linear shift between S_1 and S_2), we know $LT_{\prec}(I(S_1)) = LT_{\prec}(I(S_2)$ by Theorem 2.3. So, there is no linear shift between staircases. □

Theorem 4.6 *Any equivalence class will contain at most one staircase.*

Proof: Based on Lemma 4.1, we know that a staircase cannot be linearly shifted to another staircase. So any equivalence class will contain at most one staircase. □

Theorem 4.7 *If an equivalence class contains a staircase, then this staircase is the representative.*

Proof: Suppose $S_1 = \{(x_{11}^1, \ldots, x_{n1}^1), \ldots, (x_{1m}^1, \ldots, x_{nm}^1)\}$ is a staircase of m points and $S_2 = \{(x_{11}^2, \ldots, x_{n1}^2), \ldots, (x_{1m}^2, \ldots, x_{nm}^2)\}$ is in the same equivalence class. Then the set distance of S_2 to the origin is as follows:

$$
\begin{aligned}
D(S_2,0) &= \sum_{t=1}^{m} \sqrt{(x_{1t}^2)^2 + \ldots + (x_{nt}^2)^2} \geq \sum_{t=1}^{m} \sqrt{(x_{1t}^2 - x_{1t_1}^2)^2 + \ldots + (x_{nt}^2 - x_{nt_1}^2)^2} \\
&\geq \sum_{t=1}^{m} \sqrt{(a_1 x_{1t}^1 + b_1 - x_{1t_1}^2)^2 + \ldots + (a_n x_{nt}^1 + b_n - x_{nt_1}^2)^2} \\
&\geq \sum_{t=1}^{m} \sqrt{(a_1 x_{1t}^1 + b_1 - b_1)^2 + \ldots + (a_n x_{nt}^1 + b_n - b_n)^2} \\
&\geq \sum_{t=1}^{m} \sqrt{(a_1 x_{1t}^1)^2 + \ldots + (a_n x_{nt}^1)^2} \\
&> D(S_1,0)
\end{aligned}
$$

Here, a_i,b_i are the linear shift complements from staircase S_1 to S_2, $(x_{1t_1}^2, \ldots, x_{nt_1}^2) = (b_1, \ldots, b_n)$ is the shifted origin point in S_2. We know the distance of S_1 to origin will always be smaller than S_2 to origin for $a_i \neq 0$ for $i = 1, \ldots, m$. So, we know that S_1 is the representative in its equivalence class and we already know from Theorem 4.6 there is only one staircase in each equivalence class. Then we have that S_1 as a staircase is the representative. □

Theorem 4.8 *For any non-staircase representative $S_{R_1} \subset \mathbb{F}^n$, there is a staircase representative $S_s \subset \mathbb{F}^n$ such that: $D(S_s, 0) < D(S_{R_1}, 0)$.*

Proof: For any non-staircase representative S_{R_1}, there exists a staircase data set, denoted S_s, with a smaller distance to origin based on the Definition 2.11. Based on Theorem 4.7, S_s is the representative of an EC. So, S_s is a staircase representative. □

Example 4.3 For any non-staircase representative S_{R_i}, such as the triangle blue points $\{(0,0),(1,0),(3,0)\}$ or $\{(0,0),(1,1),(2,2)\}$ in Figure 4.4 below, we can find a staircase set S_{R_j} (circle green points) $\{(0,0),(1,0),(2,0)\}$ with $D(S_{R_i},0) > D(S_{R_j},0)$. From Theorem 4.7 we know if S_{R_j} is a staircase in its EC, then it will be a representative. For any non-staircase representative set S_{R_i} we can find a staircase S_{R_j} with a smaller set distance which is a representative in a different EC. So, for any non-staircase representative there exists a staircase representative that has a smaller set distance than the non-staircase representative.

□

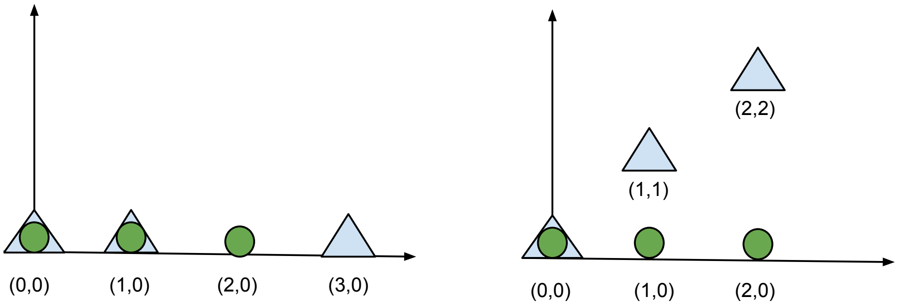


Figure 4.4: Staircase representatives and non-staircase representatives. The set with green circle points is a staircase and the set with blue triangle points is not a staircase.

However if a representative has maximum distance among all representatives, it does not necessarily have the most number of leading term ideals, as the next example illustrates.

Example 4.4 $S_1 = \{(0,0,0),(1,1,1),(2,2,2)\} \subseteq \mathbb{Z}_3^3$ has maximum distance and has 3 leading term ideals. Yet there exist sets like $S_2 = \{(0,0,0),(0,1,1),(1,0,2)\}$ with $D(S_1,0) > D(S_2,0)$ in $\mathbb{Z}_3^3$ with 3 points.

$$GB_1 = \{x_3^3 - x_3, x_3^2 + x_2 + x_3, x_3^2 + x_1 - x_3\}$$

$$GB_2 = \{x_3^2 + x_2 + x_3, x_2x_3 - x_2, x_2^2 - x_2, x_1 - x_2 + x_3\}$$

$$GB_3 = \{x_1 - x_2 + x_3, x_2^2 - x_2, x_1x_2, x_1^2 - x_1\}$$

$$GB_4 = \{x_3^2 + x_1 - x_3, x_2 - x_1 - x_3, x_1x_3 + x_1, x_1^2 - x_1\}$$

Here, S_2 has 4 reduced Gröbner bases and 4 leading term ideals. □

Chapter 5

DoEMS: A Website Linking Design of Experiments and Model Selection

The website is publicly available at https://s2.smu.edu/doems.

5.1. Introduction

For research on algebraic geometry, there are many online tools are available, e.g. GAP, Macaulay2 and Sage. Software systems such as Macaulay2 and Sage, have implemented advanced algorithms, e.g. FGLM, based on the development of algorithms for computing Gröbner bases. Researchers can use advanced computing tools to calculate model bases given data sets. However, calculating a Gröbner basis is typically a very time-consuming process for large polynomial systems. Till now no website can help researchers directly query Gröbner bases based on data alone.

As an implementation of our research results described in Chapter 3 and Chapter 4, we designed and developed a computational website DoEMS [50], to enable researchers to efficiently query GRN discrete data sets and their associated standard monomial bases, leading term ideals and Gröbner bases. From Chapter 4, we know the properties of the linear shift, ECs and representatives defined. DoEMS will also provide extra information about these properties. Meanwhile, we created a Python3 package and implemented a database that contains information about ECs and representatives in many n, m, p cases. By querying with different entries, DoEMS will return various models and data summary reports.

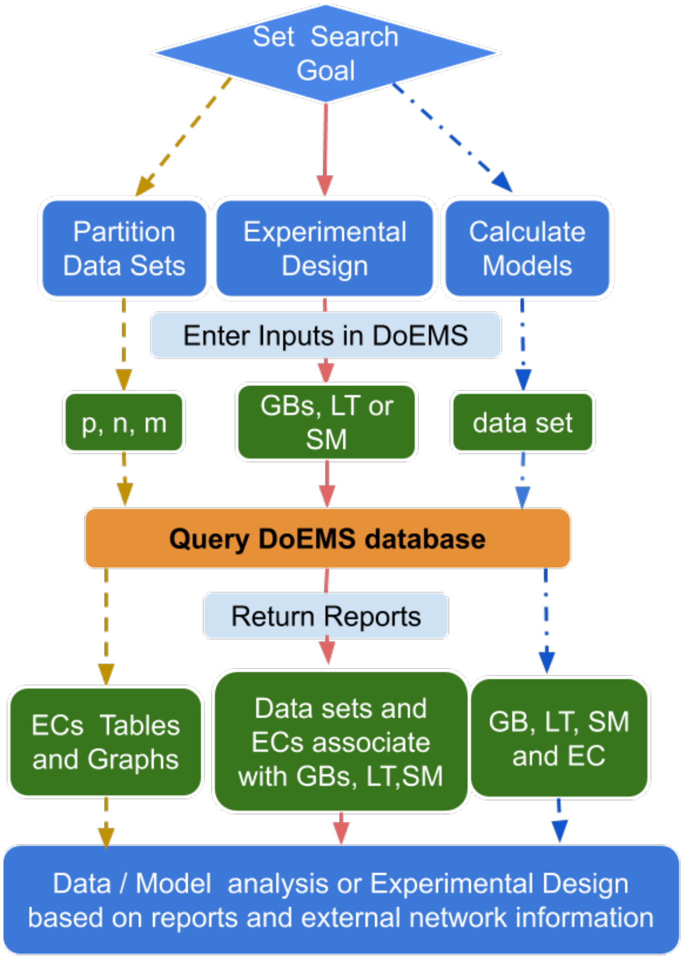


Figure 5.1: Flow chart of computational paths with DoEMS.

As the above flow chart shows, researchers can use the database as a fast method to get information on type of models and number of models given input data. More specifically, with different inputs, scientists are able get information about ECs and their representatives.

1. As the left path in the flow chart shows, Figure 5.1, some researchers may want to discover the distribution of data sets with given n, m, p. Then from our database, we can easily get partition (ECs) with different data sets associate with same SM and LT. For example, given $p = 3, n = 2$ and $m = 3$, Table 5.1 provides a summary of data set partitions. Table 5.1 is a glance of the number of GBs in the results report, which shows that there are 6 ECs.

# datasets	# classes	max(# model basis)	min(# model basis)
84	6	2	1

Table 5.1: Statistical Summary Table.

With model bases appear in Table 5.2, researchers can continue search on DoEMS to find more models' information.

# model basis	# datasets	model basis	classlabel
1	3	$\{\{1, x_2, x_2^2\}\}$	p3_n2_m3_4
1	3	$\{\{1, x_1, x_1^2\}\}$	p3_n2_m3_5
1	36	$\{\{1, x_2, x_1\}\}$	p3_n2_m3_3
2	6	$\{\{1, x_2, x_2^2\}, \{1, x_1, x_1^2\}\}$	p3_n2_m3_2
2	18	$\{\{1, x_2, x_2^2\}, \{1, x_2, x_1\}\}$	p3_n2_m3_0
2	18	$\{\{1, x_2, x_1\}, \{1, x_1, x_1^2\}\}$	p3_n2_m3_1

Table 5.2: Equivalence Classes Summary Table.

2. To make experimental design (the middle flow in Figure 5.1), usually researchers have ideal LT, model bases or GB for network model and want to recovering the data sets associate to these models. DoEMS is powerful in recovering the data sets or ECs with given model bases. From Theorem 4.1, we know the same leading term of GB models may return data sets that cannot be linear shifted to each other. So the benefit of DoEMS database is that it contains the representatives of each model basis. By fast comparison of representatives' leading term ideals, researchers can quickly get all ECs that contain all data sets which have the same leading term ideal and the same model bases. A better experimental design will contain representatives associate to given model bases. For example, with $n = 3, m = 3$ and $p = 2$, DoEMS generates the representative of each equivalence class in Table 5.3.

# datasets	representative dataset	model basis
8	$\{(0,0,0),(0,0,1),(1,0,0)\}$	$\{\{1,x_3,x_1\}\}$
8	$\{(0,0,0),(0,1,0),(1,0,1)\}$	$\{\{1,x_3,x_2\},\{1,x_2,x_1\}\}$
8	$\{(0,0,0),(1,0,1),(1,1,0)\}$	$\{\{1,x_3,x_2\},\{1,x_3,x_1\},\{1,x_2,x_1\}\}$
8	$\{(0,0,0),(0,1,1),(1,0,0)\}$	$\{\{1,x_3,x_1\},\{1,x_2,x_1\}\}$
8	$\{(0,0,0),(0,1,0),(1,0,0)\}$	$\{\{1,x_2,x_1\}\}$
8	$\{(0,0,0),(0,0,1),(1,1,0)\}$	$\{\{1,x_3,x_2\},\{1,x_3,x_1\}\}$
8	$\{(0,0,0),(0,0,1),(0,1,0)\}$	$\{\{1,x_3,x_2\}\}$

Table 5.3: Representatives Summary Table.

3. DoEMS is a very useful tool to search the model basis given data sets. Given m point data sets in $\mathbb{F}^n$, it is a time-consuming work to get all models. But by Theorem 4.1, we know data sets in the same EC will have the same standard monomial sets and the same leading term ideals. Researchers can quickly find the model basis of a given data set by checking if the data set is a linear shift of representatives in Table 5.3 in the database, which makes the computation cost as low as $O(m)$.

5.3. Data Partition and Equivalence Classes

First, to construct the SQL database, we need to make data partition based on the linear shift properties in Section 2.3. With the linear shift relationship, data sets in the same equivalence class are guaranteed to be associated with the same standard monomials.

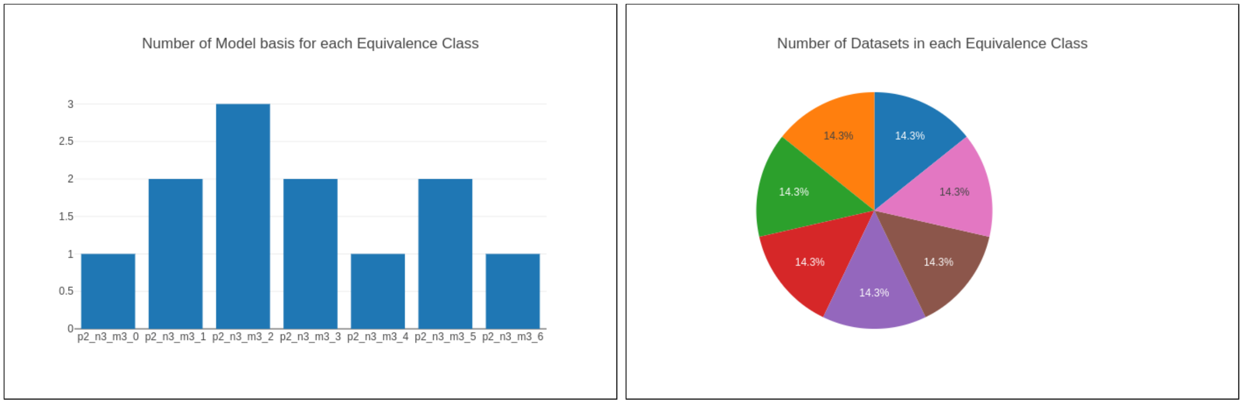
Algorithm 5.1 (Generate All ECs)

Description: generate all equivalence classes of a specific finite field $\mathbb{F}^n$ with m points.
Input: number of states: p, number of coordinates: n and number of points: m.
Output: ECs in the given case.

```
Generate all possible data sets $S = \binom{\mathbb{F}^n}{m}$ with given $p$, $n$ and $m$.
Select one data set $P_1 \in S$.
Initialize set $E = [P_1]$ and set $S_{rest} = S \setminus P_1$.
Initialize a list $L = [\ ]$ to store all mapping functions for one coordinate.

for $a$ in $[1, \dots, p-1]$
    for $b$ in $[0, \dots, p-1]$
        append function parameters list $[a, b]$ to $L$.
$LS = [[[a_1, b_1], \dots, [a_n, b_n]]$ for $0 < a_i < p$ and $0 \leq b_i < p$, $i = 1, \dots, n]$.
# Generate list of all combination of function parameters for $n$ coordinates.
while $P_1 \neq \emptyset$
    for $f_i$ in $LS = [f_1, \dots, f_k]$.
        apply $f_i$ to $P_1$ to generate new data set $P^*$.
        append $P^*$ to equivalence class list $E$ and remove $P^*$ from $S_{rest}$.
    output $E$ to file.
    set $P_1$ to first element in $S_{rest}$ and set $E = [P_1]$.
```

Example 5.1 In Figure 5.2, the Boolean network with three coordinates and 3 points will be partitioned to 7 equally sized equivalence classes (all have 8 data sets). These ECs have a different number of GBs.

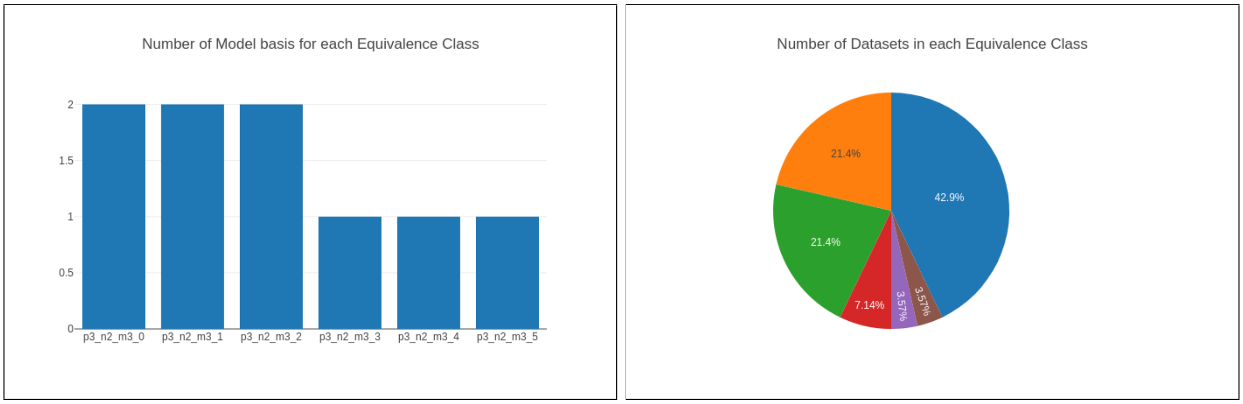


$p = 2$, $n = 3$, $m = 3$.

$p = 2$, $n = 3$, $m = 3$.

Figure 5.2: Graphs of quivalence classes with $p = 2$, $n = 3$ and $m = 3$.

In Figure 5.3, the 3-state network with 2 coordinates and 3-point data sets will be partitioned to 6 ECs. From the right plot in 5.3, we know almost half of the data sets are from the equivalence class represented by $\{(0,0), (1,0), (2,0)\}$. □



$p = 3,\ n = 2,\ m = 3.$ $p = 3,\ n = 2,\ m = 3.$

Figure 5.3: Equivalence classes graphs with $p = 2$, $n = 2$ and $m = 3$.

5.4. Check Linear Shift

Example 5.2 Considering the data set $S_2 = \{(1,0),(1,1)\}$, S_2 is the linear shift of the data set $S_1 = \{(0,0),(0,1)\}$ with mapping function set $\phi_1 = x+1, \phi_2 = x$. The specific matrix L with mapping function parameters, input matrix,denoted A_1, and output matrix, denoted A_2.

$$L \cdot A_1 = \begin{bmatrix} a_1 & 0 & b_1 \\ 0 & a_2 & b_2 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 0 \\ 0 & 1 \\ 1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 0 \\ 0 & 1 \\ 1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 0 & 1 \\ 1 & 1 \end{bmatrix} = A_2$$

Example 5.3 With known data set $S_1 = \{(0,0),(1,0),(2,1)\}$ and $S_2 = \{(0,0),(0,2),(0,1)\}$, we want to check the linear shift between these two data sets. Instead of calculating all possible linear shift functions for two variables, For $n = 2, m = 3 = n+1$, we get the invertible input matrix A_1 and output matrix A_2:

$$A_1 = \begin{bmatrix} 1 & 2 & 0 \\ 0 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix} A_2 = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 2 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

As $L \cdot A_1 = A_2$, calculate linear shift matrix:

$$L = A_2 \cdot A_1^{-1} = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 2 & 0 \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} 1 & 2 & 0 \\ 0 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix}^{-1} = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 2 & 0 \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} 1 & -2 & 0 \\ 0 & 1 & 0 \\ -1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

No matter what order of points arrangement we give to the output matrix, due to the first row containing are only zero values, we will always get first diagonal element of linear shift matrix L to be 0. which is not allowed by our definition of linear shift. It means there is no linear shift between S_1 and S_2. □

Definition 5.1 *Let S_1 and S_2 be two data sets in $\binom{\mathbb{F}^n}{m}$ and we can construct input matrix A_1 and output matrix A_2. The matrix L is a* linear shift matrix *if:*

$$L \cdot A_1 = A_2$$

To calculate linear shift matrix with $L = A_2 \cdot A_1^{-1}$, the input matrix A_1 need to be a square matrix to compute the inverse. Then input matrix A_1 and output matrix A_2 should be reconsidered with different cases of m and n: □

- $m = n + 1$: the input data sets matrix is a square matrix. The linear shift matrix L applied to input matrix A_1:

$$L \cdot A_1 = \begin{bmatrix} a_1 & \dots & 0 & b_1 \\ \vdots & \ddots & \vdots & \vdots \\ 0 & \dots & a_n & b_n \\ 0 & \dots & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{11} & \dots & x_{1m} \\ \vdots & \ddots & \vdots \\ x_{n1} & \dots & x_{nm} \\ 1 & \dots & 1 \end{bmatrix} = \begin{bmatrix} y_{11} & \dots & y_{1m} \\ \vdots & \ddots & \vdots \\ y_{n1} & \dots & y_{nm} \\ 1 & \dots & 1 \end{bmatrix} = A_2$$

If the input data sets matrix is a non-singular matrix, we can invert the matrix and get the linear shift matrix as follows:

$$L = A_2 \cdot A_1^{-1} = \begin{bmatrix} y_{11} & \dots & y_{1m} \\ \vdots & \ddots & \vdots \\ y_{n1} & \dots & y_{nm} \\ 1 & \dots & 1 \end{bmatrix} \begin{bmatrix} x_{11} & \dots & x_{1m} \\ \vdots & \ddots & \vdots \\ x_{n1} & \dots & x_{nm} \\ 1 & \dots & 1 \end{bmatrix}^{-1}_{m,n+1}$$

L is the matrix with fixed structure, satisfying the following rules:

1. All diagonal elements are non-zero.
2. The last diagonal element is 1.
3. Elements except diagonal elements and last column elements are 0.

The matrix L can be written as the sum of a diagonal matrix with parameters a_i and a matrix only with parameters b_i in the last column.

$$L = D + B = \begin{bmatrix} a_1 & \dots & 0 & 0 \\ \vdots & \ddots & \vdots & \vdots \\ 0 & \dots & a_n & 0 \\ 0 & \dots & 0 & 1 \end{bmatrix} + \begin{bmatrix} 0 & \dots & 0 & b_1 \\ \vdots & \ddots & \vdots & \vdots \\ 0 & \dots & 0 & b_n \\ 0 & \dots & 0 & 0 \end{bmatrix}$$

Here, the value of the diagonal elements is the vector $(a_1, a_2, \dots, a_n, 1)$ and the value of the last column elements is $(b_1, b_2, \dots, b_n, 1)$. It is time consuming to check all possible a_i and b_i for n variables. Suppose the input data sets produced a singular matrix, that is:

$$det(A_1) = det \begin{bmatrix} x_{11} & \dots & x_{1m} \\ \vdots & \ddots & \vdots \\ x_{n1} & \dots & x_{nm} \\ 1 & \dots & 1 \end{bmatrix} = 0$$

There is no inverse matrix of input data sets for any order of input data sets. We can't inverse the input data sets matrix but we can obtain a property based on this case.

- $m > n + 1$: the linear shift matrix and the input matrix are reconstructed as:

$$L \cdot A_1 = \begin{bmatrix} a_1 & \dots & 0 & 0 & \dots & 0 & b_1 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & \dots & a_n & 0 & \dots & 0 & b_n \\ 0 & \dots & 0 & 0 & \dots & 0 & 1 \end{bmatrix}_{n+1,m} \begin{bmatrix} x_{11} & \dots & x_{1n} & x_{1n+1} & \dots & x_{1m} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ x_{n1} & \dots & x_{nn} & x_{nn+1} & \dots & x_{nm} \\ 0 & \dots & 0 & 1 & \dots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 1 & \dots & 1 & 1 & \dots & 1 \end{bmatrix}_{m,m} = \begin{bmatrix} y_{11} & \dots & y_{1m} \\ \vdots & \ddots & \vdots \\ y_{n1} & \dots & y_{nm} \\ 1 & \dots & 1 \end{bmatrix}$$

Then the input data sets matrix becomes square matrix and if we can inverse new A_1, we will get information of linear shift matrix L.

$$L = \begin{bmatrix} a_1 & \dots & 0 & 0 \dots & 0 & b_1 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \; \vdots \\ 0 & \dots & a_n & 0 \dots & 0 & b_n \\ 0 & \dots & 0 & 0 \dots & 0 & 1 \end{bmatrix}_{n+1,m} = \begin{bmatrix} y_{11} & \dots & y_{1m} \\ \vdots & \ddots & \vdots \\ y_{n1} & \dots & y_{nm} \\ 1 & \dots & 1 \end{bmatrix} \begin{bmatrix} x_{11} & \dots & x_{1n} & \dots & x_{1m} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ x_{n1} & \dots & x_{nn} & \dots & x_{nm} \\ 0 & \dots & 1 & \dots & 0 \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ 1 & \dots & 1 & \dots & 1 \end{bmatrix}^{-1}$$

Then if first n rows and n columns in linear shift matrix L is not diagonal, there is no linear shift between input and output data sets.

- when $m < n + 1$ the output matrix and the input matrix were reconstructed with adding $n + 1 - m$ columns to the original matrix A_1 and A_2 in the case $n + 1 = m$. Then linear shift matrix L is

$$\begin{bmatrix} a_1 & \dots & 0 & b_1 \\ \vdots & \ddots & \vdots & \vdots \\ 0 & \dots & a_n & b_n \\ 0 & \dots & 0 & 1 \end{bmatrix} = \begin{bmatrix} y_{11} \dots & y_{1m} & & 0 & \dots & 0 \; b_1 \\ \vdots \; \ddots & \vdots & & \vdots & \ddots & \vdots \; \vdots \\ y_{m1} \; \dots & y_{mm} & & 0 & \dots & 0 \;\; b_m \\ y_{m+11} \; \dots & y_{m+1m} & & a_{m+1} & \dots & 0 \;\; b_{m+1} \\ \vdots \; \ddots & \vdots & & \vdots & \ddots & \vdots \; \vdots \\ y_{n1} \; \dots & y_{nm} & & 0 & \dots & a_n \; b_n \\ 1 \; \dots & 1 & & 0 & \dots & 0 \;\; 1 \end{bmatrix} \begin{bmatrix} x_{11} \; \dots & x_{1m} & 0 \dots & 0 \\ \vdots \; \ddots & \vdots & \vdots \; \ddots & \vdots \\ x_{m1} \; \dots & x_{mm} & 0 \; \dots & 0 \\ x_{m+11} \dots & x_{m+1m} & 1 \; \dots & 0 \\ \vdots \; \ddots & \vdots & \vdots \; \ddots & \vdots \\ 1 \; \dots & 1 & 0 \; \dots & 1 \end{bmatrix}^{-1}$$

Based on the above discussion, we know:

when $m = n + 1$, let the input matrix $A_1 = \begin{bmatrix} x_{11} & \dots & x_{1m} \\ \vdots & \ddots & \vdots \\ x_{n1} & \dots & x_{nm} \\ 1 & \dots & 1 \end{bmatrix}$, and when $m > n + 1$,

$$A_1 = \begin{bmatrix} x_{11} & \dots & x_{1n} & x_{1n+1} & \dots & x_{1m} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ x_{n1} & \dots & x_{nn} & x_{nn+1} & \dots & x_{nm} \\ 0 & \dots & 0 & 1 & \dots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 1 & \dots & 1 & 1 & \dots & 1 \end{bmatrix}, \text{ and when } m < n + 1,\ A_1 = \begin{bmatrix} x_{11} & \dots & x_{1n} & \dots & x_{1m} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ x_{n1} & \dots & x_{nn} & \dots & x_{nm} \\ 0 & \dots & 1 & \dots & 0 \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ 1 & \dots & 1 & \dots & 1 \end{bmatrix}.$$

Based on the discussion above, we know from the linear shift matrix, we can quickly check the linear shift relationship between two data sets. By the above construction of the input matrix and output matrix, we can calculate the linear shift matrix in the case of $det(A_1) \neq 0$ and $det(A_2) \neq 0$.

Algorithm 5.2 (Existence of Linear Shift)

Description: check if one data set is a linear shift of the other data set.
Input: two different data sets S_1, $S_2 \in \binom{\mathbb{F}^n}{m}$.
Output: return Yes if S_1 is a linear shift of S_2, else return No.

```
Initialize matrix $A_1$ from input $S_1$.
Initialize matrix $A_2$ from input $S_2$.
if $det(A_1) \neq 0$
  if $det(A_2) \neq 0$
    compute linear shift matrix $L = A_2 \cdot A_1^{-1}$.
    if first $n$ rows and $n$ columns of $L$ is a diagonal matrix
      return YES
```

```
        else: return NO
    else: return Unknown
else: return Unknown
```

However, the linear shift matrix cannot return all possible linear shift relationship. To implement our theorems of relationships, we created a Python package based on Algorithm 5.3 to find all mapping functions between two data sets.

Algorithm 5.3 (Generate Linear Shift Mapping Functions)

Description: return all possible linear shift mapping function for two data sets.
Input: two different data sets S_1 and S_2.
Output: return mapping functions if S_1 is a linear shift of S_2, else return "No".

```
Pick first point $P_0$ in $S_2$.
Initialize the mapping function list $F$.
Initialize result list $R$.
Initialize number of states $p$.

for point $P$ in $S_1$
    if $P = P_0$: Continue.
    else: for coordinate $i$ in $[1, \ldots, n]$
            for $a$ in $[1, \ldots, p-1]$
                $b = (P_i - a * P_{0i})\%p$
                append $[a, b]$ to $F_i$
                # append all possible mapping function parameters tuple to $f_i$
        $LS = \{(f_1, f_2, ..., f_n)$ for $f_1$ in $F_1$ ... for $f_n$ in $F_n\}$
        # generate all possible mapping function combinations for $n$ coordinates.
        for $f$ in $LS$
            $S_3 = f(S_2)$
            # generate $S_3$ from $S_2$ using mapping functions in $f$.
            if $S_3 = S_1$
                add function list $f$ to $R$.
return "No" if length(R) = 0, else return R.
```

Example 5.4 For $S_1 = \{(0,0),(1,0)\}$ and $S_2 = \{(0,1),(1,1)\}$, using the Python package with Algorithm 5.3, we can get linear shift parameters for all possible functions: $[[[a_{11}, b_{11}], [a_{12}, b_{12}]], [[a_{21}, b_{21}], [a_{22}, b_{22}]]] = [[[1,0],[1,1]],[[1,1],[1,1]]]$, which means there are two possible mapping functions sets: $\Phi_1 = (x, x+1), \Phi_2 = (x+1, x+1)$. □

5.5. Extract Equivalence Class Information

After generating all equivalence classes by Algorithm 5.1, we can make a model selection by data sets with more meaningful structures such as the representative of each EC, which has the smallest set distance. If an EC contains a staircase, it indicates that all data sets in the EC associated with a unique GB by Theorem 2.4 and this staircase is the representative by Theorem 4.7.

Algorithm 5.4 (Find Representatives)

Description: pick a representative from an equivalence class.
Input: an equivalence class E.
output: a representative of E.

```
Initialize representatives list: $rep = [\ ]$.
Initialize $rep = dataset[0]$; $D = +\infty$ ; $count = 0$.
for data set $S$ in equivalence class $E$
    Initialize $D_{new}$ as distance to the origin $D(S, 0)$.
    if $D_{new} < D$
            $D = D_{new}$
            $rep = S$
return rep,distance
```

Algorithm 5.5 (Find Staircase)

Description: Check if a data set is a staircase.
Input: a set of input points S
Output: return True if input data set S is a staircase, else return False.

#Find boundary points B of data set S

```
Initialize $B = \{\}$.
for $i$ from $0$ to $|S| - 1$ do
    set $P_{max} = S_i$
    for $j$ from $0$ to $|S| - 1$ do
        if $S_i \neq S_j$ and all coordinates of $S_j \geq S_i$
          then $P_{max} = S_j$
    $B = B \cup \{P_{max}\}$
#Create staircase points $S_{stair}$ with boundary points $B$.
Initialize $S_{stair} = \{B[0]\}$
for each point $P$ in $B$ do
    # calculate all points below $P$.
    $S_{new} = \{(x_1, x_2, ..., x_n)$ for $x_1$ in $[0, \ldots, P_1 - 1]$ ...for $x_n$ in $[0, \ldots, P_n - 1]\}$
    $S_{stair} = S_{stair} \cup S_{new}$
return True if $S = S_{stair}$, else return False.
```

Algorithm 5.6 (Find Standard Monomial)

Description: Check if a point is in the standard monomial data set with given leading terms.
Input: a point P and leading terms LT.
Output: return True if input point is in standard monomial data set, else return False.

```
Initialize $e$ as the exponents of $P$.
Initialize $E$ as the exponents of $LT$.
Initialize $flag_{less} = True$.
For $i$ from $0$ to $|E| - 1$ # for loop of each leading term exponent in $LT$.
    Initialize $flag_{small} = 0$ # $flag_{small}$ records number of variables less than leading term $E_i$
    for $j$ from 0 to $|e| - 1$ # for loop of each coordinate exponent of $P$.
        if $e_j < E_{ij}$
          $flag_{small} = flag_{small} + 1$
    If $flag_{small} = 0$: $flag_{less} = False$
    # $P$ is on the leading term boundary or out of the LT points boundary.
return True if $flag_{less} = True$, else return False.
```

Chapter 6

Applications

The results about the model analysis in this chapter are published in [27].

6.1. Reverse Engineering of Gene Regulatory Networks

The first GRN is considered well understood is *lac* operon. However, only a few GRNs are fully understood and studied. Therefore, instead of constructing models and fit data, researchers are more focused on developing modelling methods that will generate models directly from input data sets. These data-driven modelling methods are called *reverse engineering*. In other words, reverse engineering of gene regulatory networks is a process that recovers the regulatory relationships between genes in the system. Reverse engineering is based on input data sets such as gene activity level or expression level. Many methods have been proposed in the last few decades leading to a wide range of mathematical approaches [29].

There are numerous modeling techniques used in reverse engineering, including machine learning, Bayesian networks, Boolean networks, differential equations, information theory, PetriNets, neural networks, and genetic algorithms [19]. Each method has its advantages and disadvantages. With the increase in the number of genes, the computational cost is enormous. For example, for 30 genes, there are $2.71{\cdot}10^{158}$ probable network variants using Bayesian networks [28].

In [9], the authors developed a comparative study of five reverse engineering methods: 1. Boolean methods are used to infer GRNs by applying Boolean logic to the discretized gene states (0 or 1 states), which indicated the expression level of mRNA. 2. Bayesian methods use the Baye's rule to reverse engineer GRNs by using conditional probability distributions

[18]. 3. regulation matrix methods that use the linear system $y = b_i + \sum_{j=1}^{n} a_{ij}x_j$ around a steady state. Moreover, steady-states are measures before/after a specific perturbation such as temperature. 4. Both gene expression profiles and biological information should be considered to avoid the fundamental limits of methods related to high dimension or computational complexity. Alternatively, eliminate the uncertainty of data. 5. Machine learning approaches, which mostly focuse on genetic algorithms, neural networks or fuzzy logic. Technologies are developed for inference algorithms [48] and gene expression data clustering [37].

In the last decades, more and more novel methods have been developed in many gene networks [12]. Scientists pay more attention to data relationship or data manipulation in the process of reverse engineering modelling. For example, the missing input data sets measurement has been proposed to deal with the high-dimensional time-series data [34]. Alternatively, a novel unsupervised machine learning methodology has been developed to analyze a data set and determine the most probable number of states [45]. More methods have been applied to GRN problems and implemented to a more accurate and applicable analysis of GRN.

In the above reverse engineering methods, discrete models of GRN have gained popularity in computational systems biology. In this field, the algebraic geometry of data sets and model selection methods based on given input data sets are studied. The problems in GRN model selection are challenging considering the enormous computational costs for all possible models and the qualitative and quantitative analyses for different models before making a final decision. Based on previous research on discrete models of GRN [15], to best recover the specific gene regulatory networks, scientists can choose Gröbner bases, which can reflect right and concise regulatory relationships in systems.

6.2. Associating Models with Gröbner Bases

We consider input data sets with 4, 5, 7 points as examples to show how we analyze models based on Gröbner bases. Examples 6.1, 6.2 and 6.3 show a work flow of model analysis highlighting the relationship between data sets and their Gröbner bases, standard monomial sets, and number of PDS models as we discussed in Chapters 3 and 4. Recall that 2 GBs can result in the same model. So here we start from a data set associated with the most different GBs and show the relationship between each GB and associated SM and model. Last, we will present the experimental design of reducing the number of GBs by adding extra points.

Example 6.1 For 4 points $S_1 = \{(0,0,1,1),(0,1,0,0),(1,0,0,1),(1,0,1,0)\}$, the model computation process is

1. Extract Gröbner bases and standard monomials of the equation f_{x_1} based on the output point $[0,1,0,1]^T$ and 4 input points S_1. Here, there are six different Gröbner basis and associate standard monomials in Appendix B.

2. Using the evaluation matrix, calculate 6 models related to 6 different GBs. For example, with

$$GB_1 = \{x_4^2 + x_4, x_3^2 + x_3, x_3x_4 + x_2 + x_3 + x_4 + 1, x_1 + x_3 + x_4\}$$

   and $SM_1 = \{1, x_4, x_3, x_4x_3\}$, we get the linear system problem:

$$\begin{array}{|cccc|}\hline 1 & x_4 & x_3 & x_4x_3 \\ \hline 1 & 1 & 1 & 1 \\ \hline 1 & 0 & 0 & 0 \\ \hline 1 & 0 & 1 & 0 \\ \hline 1 & 1 & 0 & 0 \\ \hline \end{array} \cdot \begin{array}{|c|}\hline a \\ \hline b \\ \hline c \\ \hline d \\ \hline \end{array} = \begin{array}{|c|}\hline \text{output} \\ \hline 0 \\ \hline 1 \\ \hline 0 \\ \hline 1 \\ \hline \end{array}$$

   Solving this linear system we get $a = 1, b = 0, c = 1, d = 0$, so

$$f_{x_1} = a \cdot 1 + b \cdot x_4 + c \cdot x_3 + d \cdot x_3x_4 = 1 + x_4$$

Based on the same process which results in GB_1 to GB_6, we conclude that there are 2 different models for gene x_1:

$$\textit{4 GBs} \Longrightarrow Model_1 : f_{x_1} = 1 + x_4$$

$$\textit{2 GBs} \Longrightarrow Model_2 : f_{x_1} = 1 + x_1 + x_3$$

Then we analyze different model results. Here, $Model_1$ is related to 4 GBs and $Model_2$ is related to 2 GBs. So 6 GBs yields 2 normal forms.

3. A further step can help to get a unique Gröbenr basis, after adding at least 4 points $\{(0,0,0,0),(0,0,0,1),(0,0,1,0),(1,0,0,0)\}$ to original data set S, we got a staircase data set: $\{(0,0,0,0),(0,0,0,1),(0,0,1,0),(0,0,1,1),(0,1,0,0),(1,0,0,0),(1,0,0,1),$ $(1,0,1,0)\}$, which has a unique GB model:

$$GB = \{x_4^2 + x_4, x_3^2 + x_3, x_2x_4, x_2x_3, x_2^2 + x_2, x_1x_3x_4, x_1x_2, x_1^2 + x_1\}$$

$SM = \{1, x_4, x_3, x_3x_4, x_2, x_1, x_1x_4, x_1x_3\}$ and the model for gene x_1 is:

$$Model : f_{x_1} = x_2 + x_3 + x_3x_4$$

With output data $[0,0,1,0,1,0,0,1]^T$, the expression model is fixed and unique.

□

Example 6.2 For 5 points $S_2 = \{(0,0,1,0),(0,1,0,1),(1,0,0,1),(1,1,0,0),(1,1,1,1)\}$ researchers can at most get 13 different GBs. Different models are generated from 13 GBs with output data $[1,0,0,1,0]^T$ for gene x_1. For example, based on SM_1 and SM_3, different models are generated:

$$\begin{aligned} SM_1 : \{1, x_4, x_3, x_3x_4, x_2\} \qquad & SM_3 : \{1, x_4, x_2, x_2x_4, x_1\} \\ f_{x_1} = 1 + x_4 \qquad & f_{x_1} = 1 + x_4 \\ f_{x_2} = 1 + x_2 + x_3 + x_4 \qquad & f_{x_2} = x_1 + x_2 + x_4 + x_2x_4 \\ f_{x_3} = x_3 \qquad & f_{x_3} = 1 + x_1 + x_2x_4 \\ f_{x_4} = x_4 \qquad & f_{x_4} = x_4 \end{aligned}$$

Here, SM_1 and SM_3 have same model function for x_1. Then we summarized 4 different models generated from same data set and $Model_1$ has the largest frequency among all models:

$$\begin{aligned} \textit{10 GBs} \Longrightarrow Model_1 : f_{x_1} &= 1 + x_4 \\ GB_5 \Longrightarrow Model_2 : f_{x_1} &= 1 + x_1 + x_2 + x_2x_3 \\ GB_{11} \Longrightarrow Model_3 : f_{x_1} &= 1 + x_1 + x_2 + x_1x_3 \\ GB_{13} \Longrightarrow Model_4 : f_{x_1} &= x_3 + x_1x_2 \end{aligned}$$

We now analyze different model results. Here, $Model_1$ is related to 10 GBs and $Model_2$, $Model_3$ and $Model_4$ are related to a unique GB. So 13 GBs yields 4 normal forms.

With further calculation, we need add at least 6 more points to get a unique GB generated from 11 points: $\{(0,0,0,0), (0,0,0,1), (0,0,1,0), (0,0,1,1), (0,1,0,1), (0,1,1,1), (1,0,0,0), (1,0,0,1), (1,1,0,0), (1,1,0,1), (1,1,1,1)\}$ with fixed one SM: $\{1, x_4, x_3, x_3x_4, x_2, x_2x_4, x_2x_3, x_1, x_1x_4, x_1x_3, x_1x_2\}$ and unique GB.

$$Model : f_{x_1} = x_2 + x_3 + x_1x_2 + x_1x_3 + x_2x_4 + x_3x_4$$

□

Example 6.3 For 7 points $S_3 = \{(0,0,0,0), (0,0,0,1), (0,0,1,0), (0,1,0,0), (1,0,1,1), (1,1,0,1), (1,1,1,0)\}$ we get 2 different SMs and resulting models as below:

$$Model_1 : f_{x_1} = x_2 + x_3 + x_2x_4 + x_3x_4$$

$$Model_2 : f_{x_1} = x_1 + x_2 + x_3$$

After adding 4 more points: $\{(0,0,0,0),(0,0,0,1),(0,0,1,0),(0,1,0,0),(1,0,0,0),(1,0,0,1),$ $(1,0,1,0),(1,0,1,1),(1,1,0,0),(1,1,0,1),(1,1,1,0)\}$, we will get a unique GB model:

$$Model : f_{x_1} = x_2 + x_3 + x_1x_2 + x_1x_3 + x_2x_3.$$

□

Example 6.4 Consider data sets in $\mathbb{Z}_2^4$. Let S_{max} be a data set whose ideal of points has the maximum number of Gröbner bases. Define $S_{unique} = S_{max} \cup S_{add}$ where S_{add} is a collection of points such that the augmented data set S_{unique} has an ideal of points with a unique GB. With the Python package for searching least number of adding points in getting unique GB, the summary in Table 6.1 for different cases is

$\lvert S_{max}\rvert$	2	3	4	5	6	7	8	9	10	11	12	13
$\lvert S_{unique}\rvert$	5	5	8	11	11	11	11	12	15	15	15	15
$\lvert S_{add}\rvert$	3	2	4	6	5	4	3	3	5	4	3	2
$\#GB_{S_{max}}$	4	5	6	13	12	13	9	13	12	13	6	5
% reduce	0.25	0.40	0.21	0.15	0.18	0.23	0.30	0.31	0.18	0.23	0.27	0.40

Table 6.1: Adding the fewest number of points to data sets with the maximum number of GBs to create data sets with unique GBs.

The table summarizes the fewest points which must be added to guarantee a unique GB from a data set associated with the maximum number of GBs for different sized data sets. Here, $\%reduce = \frac{\#GB_{S_{max}}-1}{\#GB_{S_{max}} \cdot |S_{add}|}$ is the percentage of the number of GB reduced with adding one point. From the table, an average of 20% of the maximum number of GBs will be reduced by adding one extra point in each case. □

6.3. *Lac* Operon Network

The *lac* operon is a system of genes which control the transport and metabolism of lactose in many bacteria including *E. coli*. While there are numerous models for the *lac* operon (see, for example, [23, 36, 40, 49]), we consider a Boolean model proposed in [46]. There the authors reduced the system to a core sub-network consisting of the following four variables: M representing *lac* mRNA, L inter-cellular lactose, L_e extracellular lactose, and G_e extracellular glucose. The Boolean model for this sub-network is given by the following Boolean functions, where extraneous variables are introduced to capture intermediate values (L_m, L_{em}) of lactose inside and outside of the cell respectively: see Section 4.2.2 in [46] for a full description of the model.

$$\begin{aligned} f_M &= \neg G_e \wedge (L \vee L_m) & f_{L_e} &= L_e \\ f_L &= M \wedge L_e \wedge \neg G_e & f_{G_e} &= G_e \\ f_{L_m} &= ((L_{em} \wedge M) \vee L_e) \wedge \neg G_e & f_{L_{em}} &= L_{em} \end{aligned}$$

For the sake of illustrating the utility of the above results, we reduce this model to only include the four essential variables. To this end, we replace L_{em} with L_e and L_m with L, and remove all instances of L_{em} and L_m via substitution. Doing so produces

$$f_{L_m} = ((L_e \wedge M) \vee L_e) \wedge \neg G_e = L_e \wedge \neg G_e \tag{6.1}$$

which we substitute into the function f_M:

$$f_M = \neg G_e \wedge (L \vee (L_e \wedge \neg G_e)) = \neg G_e \wedge (L \vee L_e). \tag{6.2}$$

This results in the following Boolean network on four variables, with wiring diagram depicted in Figure 6.1:

$$\begin{aligned} f_M &= \neg G_e \wedge (L \vee L_e) \\ f_L &= M \wedge L_e \wedge \neg G_e \\ f_{L_e} &= L_e \\ f_{G_e} &= G_e. \end{aligned}$$

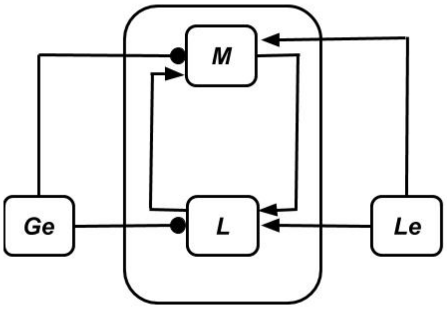


Figure 6.1: Wiring diagram for a simplified Boolean model of the *lac* operon in *E. coli.* Directed edges with pointed ends indicate positive regulation, while directed edges with round ends indicate negative regulation. The variables G_e and L_e regulate the operon from outside the cell, represented by a rectangle around M and L.

Boolean functions can be rewritten as polynomial functions over $\mathbb{Z}_2$ using the following translations: the Boolean expression $x \vee y$ can be represented as the polynomial $x + y + xy$, $x \wedge y$ as xy, and $\neg x$ as $x + 1$. Applying these rules to the above functions yields the finite dynamical system $f : \mathbb{Z}_2^4 \to \mathbb{Z}_2^4$; where, $f = (f_{x_1}, f_{x_2}, f_{x_3}, f_{x_4})$ and each f_{x_i} is a polynomial in the variables $x_1 := M$, $x_2 := L$, $x_3 := L_e$ and $x_4 := G_e$.

$$\begin{aligned} f_{x_1} &= x_2x_3x_4 + x_2x_3 + x_2x_4 + x_3x_4 + x_2 + x_3 \\ f_{x_2} &= x_1x_3x_4 + x_1x_3 \\ f_{x_3} &= x_3 \\ f_{x_4} &= x_4 \end{aligned}$$

The standard monomials set for this model is

$$SM = \{x_2x_3x_4, x_1x_3x_4, x_2x_3, x_1x_4, x_2x_4, x_1x_3, x_3x_4, x_1, x_2, x_3, x_4, 1\}.$$

Based on our database of linear shift, the equivalence class that has the same standard monomials will satisfy the reduced *lac* operon model. For the same standard monomials, based on our linear shift properties, we get the equivalence class with 4 different GBs:

$$GB_1 = \{x_4^2 + x_4, x_3^2 + x_3, x_2^2 + x_2, x_1x_2, x_1^2 + x_1\},$$

$$GB_2 = \{x_4^2 + x_4, x_3^2 + x_3, x_2^2 + x_2, x_1x_2 + x_1, x_1^2 + x_1\},$$

$$GB_3 = \{x_4^2 + x_4, x_3^2 + x_3, x_2^2 + x_2, x_1x_2 + x_2, x_1^2 + x_1\},$$

$$GB_4 = \{x_4^2 + x_4, x_3^2 + x_3, x_2^2 + x_2, x_1x_2 + x_1 + x_2 + 1, x_1^2 + x_1\}$$

and each of them corresponds to data sets S_1, S_2, S_3, S_4, respectively in Appendix B.

From the study of equivalence classes, we know the associated data set: $\{S_1, S_2, S_3, S_4\}$ of $\{GB_1, GB_2, GB_3, GB_4\}$ is a equivalence class with the same SM_s, $D(S_1, 0) \leq D(S, 0), S \in \{S_1, S_2, S_3, S_4\}$. Here, the S_1 is associated with GB_1, $S_1 = S^R$ is the representative data set of reduced model and also a staircase data set. By selecting the representative, it will associate with the most simplified GB and functions for the reduced model. The expression functions of different genes can be reconstructed from any data sets above with the evaluation matrix.

Same as reduced model in Figure 6.1, we get polynomials of advanced models with same strategy. Reduced model without inducer inclusion (left in Figure 6.2):

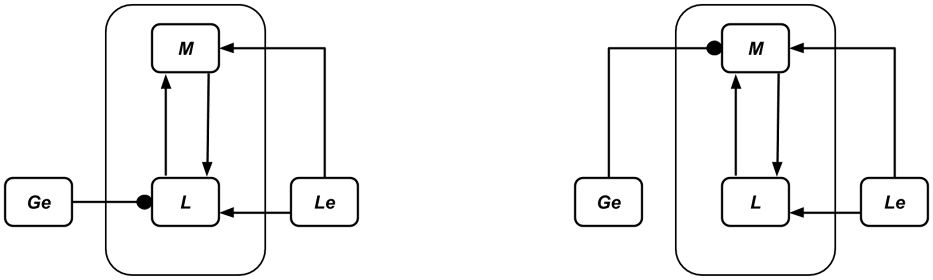


Figure 6.2: Two advanced models: without inducer exclusion (left), and without catabolic repression (right).

$$\begin{aligned} f_M &= \neg G_e \wedge (L \vee L_m) \\ f_L &= M \wedge L_e \\ f_{L_m} &= (L_{em} \wedge M) \vee L_e \end{aligned} \quad \Longrightarrow \quad \begin{aligned} f_{x_1} &= x_2x_3x_4 + x_2x_3 + x_2x_4 + x_3x_4 + x_2 + x_3 \\ f_{x_2} &= x_1x_3 \\ f_{x_3} &= x_3 \\ f_{x_4} &= x_4 \end{aligned}$$

The standard monomials for the model without inducer exclusion (left in Figure 6.2) is $SM_1 = \{x_2x_3x_4, x_1x_3, x_2x_3, x_2x_4, x_3x_4, x_3, x_2, x_4, x_1, 1\}$. For the same standard monomial set SM_1, based on linear shift properties, we get 8 different GBs from 8 different data sets.

Then the reduced model without catabolite repression is (right in Figure 6.2):

$$\begin{aligned} f_M &= L \vee L_m \\ f_L &= M \wedge L_e \wedge \neg G_e \\ f_{L_m} &= ((L_{em} \wedge M) \vee L_e) \wedge \neg G_e \end{aligned} \quad \Longrightarrow \quad \begin{aligned} f_{x_1} &= x_2x_3x_4 + x_2x_3 + x_3x_4 + x_2 + x_3 \\ f_{x_2} &= x_1x_3x_4 + x_1x_3 \\ f_{x_3} &= x_3 \\ f_{x_4} &= x_4 \end{aligned}$$

The standarad monomials set for this model is $SM_2 = \{x_2x_3x_4, x_1x_3x_4, x_2x_3, x_1x_4, x_2x_4, x_1x_3, x_3x_4, x_1, x_2, x_3, x_4, 1\}$. The resulting GBs and data sets are the same as for the original reduced model.

The data sets with a unique model are the data set that can only recreate given expression functions with fixed SMs. No more than one expression SMs set means the data set doesn't contain redundant information. This "filter" step is very important considering the reduction of computation cost. For example, the model without inclusion (left in Figure 6.2) to all possible 10 points data sets, the linear shift methods decrease candidates number from $|\binom{\mathbb{F}^4}{10}| = 8008$ to 8. At last, we make the model selection with a representative data set, which will fully recover the model information with simplified Gröbner bases.

6.3.1. Experimental Design of *Lac* Operon

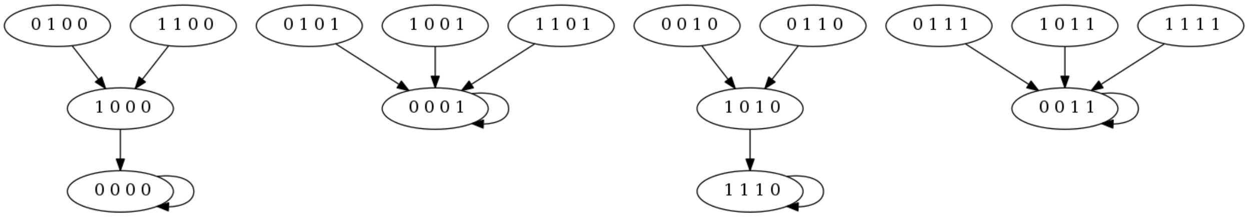


Figure 6.3: State space graph for the 4-dimensional finite dynamical system. Each node is a state (M, L, L_e, G_e) of the network and a directed edge from state a to state b indicates that $f(a) = b$.

Consider the first component of the state space of f in Figure 6.3: $C_1 = \{(0,0,0,0), (0,1,0,0), (1,0,0,0), (1,1,0,0)\}$. Note that the data points in C_1 form a staircase. By Corollary 2.3, the ideal $I(C_1)$ has a unique reduced Gröbner basis, namely

$$GB_1 = \{x_1^2 + x_1, x_2^2 + x_2, x_3, x_4\}.$$

In particular the data set C_1 has the unique leading term ideal $L = \langle x_1^2, x_2^2, x_3, x_4\rangle$ and standard monomial basis $S = \{1, x_1, x_2, x_1x_2\}$ for any monomial order. If we label the other components similarly, $C_2 = \{(0,0,0,1), (0,1,0,1), (1,0,0,1), (1,1,0,1)\}$, $C_3 = \{(0,0,1,0),$

$(0,1,1,0),(1,0,1,0),(1,1,1,0)\}, C_4 = \{(0,0,1,1),(0,1,1,1),(1,0,1,1),(1,1,1,1)\}$, we find that they are linear shifts of C_1, that is, $C_1 \overset{\phi_{12}}{\sim} C_2$, $C_1 \overset{\phi_{13}}{\sim} C_3$, and $C_1 \overset{\phi_{14}}{\sim} C_4$, where

$$\phi_{12} = (x_1, x_2, x_3, x_4+1), \phi_{13} = (x_1, x_2, x_3+1, x_4), \phi_{14} = (x_1, x_2, x_3+1, x_4+1).$$

According to Theorem 3.5, the data sets C_2, C_3, and C_4 have the same leading term ideal and standard monomial basis as C_1. So each of C_2, C_3, and C_4 also has a unique reduced Gröbner basis; however all four data sets have *different* unique reduced Gröbner bases.

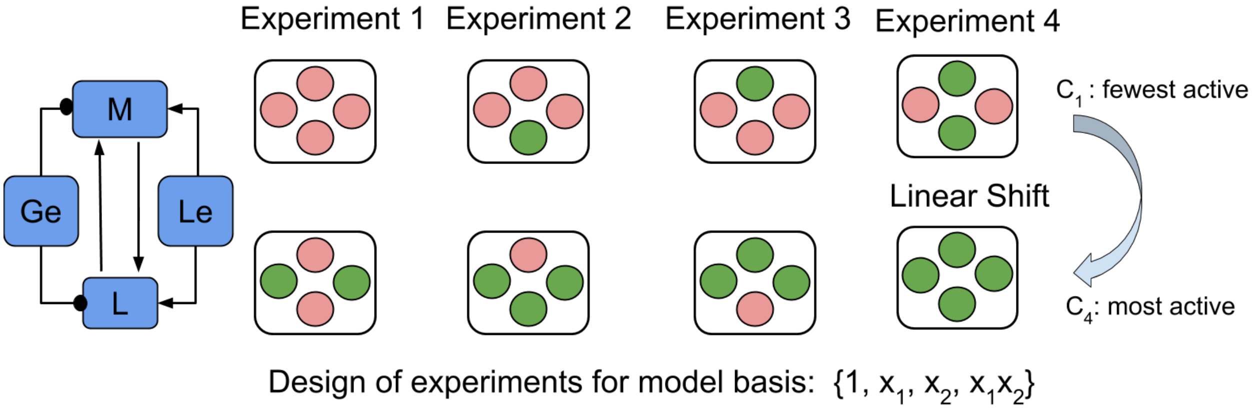


Figure 6.4: Experimental design for a Boolean network of the *lac* operon. The top row contains data sets with the fewest active nodes. The bottom row contains data sets with the most active nodes. Green represents 1 (active) and red represents 0 (inactive).

6.3.2. Efficient Way to Compute Gröbner Bases

Furthermore, we can directly apply the linear shift functions to produce the generators of the other reduced Gröbner bases explicitly, rather than computing them from the respective ideals:

$$GB_2 = GB(I(C_2)) = \{x_1^2 + x_1, x_2^2 + x_2, x_3, x_4 + 1\},$$

$$GB_3 = GB(I(C_3)) = \{x_1^2 + x_1, x_2^2 + x_2, x_3 + 1, x_4\},$$

$$GB_4 = GB(I(C_4)) = \{x_1^2 + x_1, x_2^2 + x_2, x_3 + 1, x_4 + 1\}.$$

While algorithms (and their corresponding complexities) related to the above theoretical results are not in the scope of the presented work, we close with a note about its potential to reduce significantly the time to compute Gröbner fans of zero-dimensional ideals. The worst-case complexity of computing one Gröbner basis of a zero-dimensional ideal in a general setting is quadratic in the number of variables n and cubic in the number of points m, that is $\mathcal{O}(nm^3+n^2m^2)$ [1], with various improvements in specialized settings. Computing a Gröbner fan for a zero-dimensional ideal from a given Gröbner basis is proved to be "a polynomial-time algorithm in the size of the output" [20]. In settings where data sets yield Gröbner fans with distinct cones, we can take advantage of linear shifts. From one data set and its calculated fan (set of reduced Gröbner bases), we can use the linear shift functions to produce the reduced Gröbner bases for the ideals of the linearly shifted points.

6.4. EGFR Inhibition Model for Tumor Growth

Discrete dynamic models including PDSs can be used in systems pharmacology. In the research of EGFR in [42], the authors translated a previously constructed pharmacodynamic model of growth factor receptor (EGFR) signaling to discrete models, a Boolean model and a three-state model. Also, they showed how the effects of an EGFR inhibitor could suppress tumor growth. In their discussion, the results of the prediction of EGFR inhibitor effects on tumor growth are mostly truth tables and based on Boolean rules.

However, there are no certain polynomials or GRN equations for the network of EGFR to quantify the effects of EGFR inhibitor effects on tumor growth. In this section, we want to construct the PDS of the network in Figure 6.5. After constructing Boolean PDS model and non-Boolean PDS model, we can find monomial bases. Then apply our linear shift theorems to find ECs associated with the same model basis.

For the Boolean network in Figure 6.5, based on the theorems in Chapter 4, we consider model selection.

$$\begin{aligned} Rkip^* &= \neg EGFR \\ Kras^* &= EGFR \vee (\neg Rasgap) \\ Raf1^* &= Kras \vee (\neg Rkip) \\ Proliferation^* &= Raf1 \wedge miR221 \end{aligned}$$

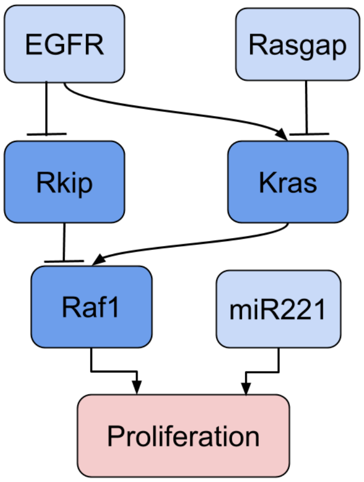


Figure 6.5: EFGR Model

$$f_{x_1} = E + 1 \tag{6.3}$$

$$f_{x_2} = E + (R + 1) + E(R + 1) \tag{6.4}$$

$$f_{x_3} = x_2 + (x_1 + 1) + x_2(x_1 + 1) \tag{6.5}$$

$$f_{x_4} = x_3M \tag{6.6}$$

The function (6.3) for x_1 and function (6.4) for x_2 are fixed by the parameters E and R. Notice that the function for x_4 is only related to the variable x_3 and the parameter M. So to recover the functions for the network, we will focus on reverse engineering the functions for x_3 and x_4.

From the model functions (6.5) and (6.6), we know that to recover this network the standard monomials should at least contain $\{x_2x_1, x_2, x_1, x_3, 1\}$, since this is the smallest factor-closed set. First, we can get the stable states for the 16 possible points in the case $p = 2$ and $n = 4$ in Appendix B. By searching the equivalence class associated with SM $= \{1, x_1, x_2, x_3, x_1x_2\}$, we found the equivalence class that can recover the network in the following table.

|Bases|	|Data sets|	Standard monomials
1	8	$\{\{1, x_3, x_2, x_1, x_1x_2\}\}$
2	8	$\{\{1, x_3, x_2, x_2x_3, x_1\}, \{1, x_3, x_2, x_1, x_1x_2\}\}$
2	8	$\{\{1, x_3, x_2, x_1, x_1x_3\}, \{1, x_3, x_2, x_1, x_1x_2\}\}$
3	8	$\{\{1, x_3, x_2, x_2x_3, x_1\}, \{1, x_3, x_2, x_1, x_1x_3\}, \{1, x_3, x_2, x_1, x_1x_2\}\}$

Table 6.2: All equivalence classes associated with standard basis: $\{1, x_1, x_2, x_3, x_1x_2\}$

To recover the full EFGR network, suppose we start with the data set

$$\{(0, 0, 0), (0, 1, 0), (1, 0, 0), (1, 1, 0)\} \to \{1, x_1, x_2, x_1x_2\}$$

which is associated with the standard monomial basis $\{1, x_1, x_2, x_1x_2\}$. What are all the possible points that need to be added to recover the basis $\{1, x_1, x_2, x_1x_2, x_3\}$? Based on Table 6.3, we know there are 4 candidates to recover the network with standard monomials $\{1, x_1, x_2, x_1x_2, x_3\}$ and one data set is closest to the origin. Each of the closest data sets can linear shift to the other three data sets in the other rows. The experimental design in Figure 6.6 contains two data sets that can recover the desired standard monomials $\{1, x_1, x_2, x_1x_2, x_3\}$. The first row is the data set with the fewest active nodes and the second row is the data set with the most active nodes.

|Bases |	Data set	Smallest distance
1	$\{\{0, 0, 0\}, \{0, 0, 1\}, \{0, 1, 0\}, \{1, 0, 0\}, \{1, 1, 0\}\}$	yes
1	$\{\{0, 0, 0\}, \{0, 1, 0\}, \{0, 1, 1\}, \{1, 0, 0\}, \{1, 1, 0\}\}$	no
1	$\{\{0, 0, 0\}, \{0, 1, 0\}, \{1, 0, 0\}, \{1, 0, 1\}, \{1, 1, 0\}\}$	no
1	$\{\{0, 0, 0\}, \{0, 1, 0\}, \{1, 0, 0\}, \{1, 1, 0\}, \{1, 1, 1\}\}$	no

Table 6.3: Recover EFGR model basis: $\{1, x_1, x_2, x_1x_2, x_3\}$ set by adding extra point.

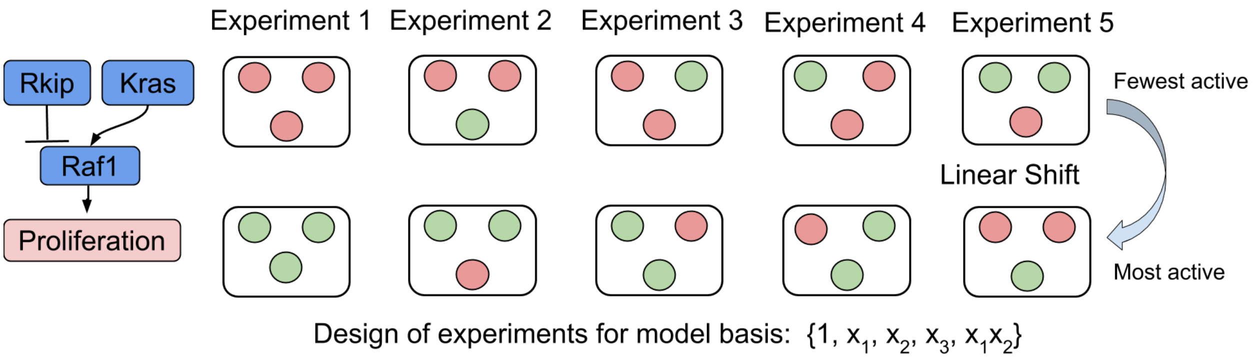


Figure 6.6: Experimental design of EFGR Boolean network with smallest distance data set. Green represents 1 (active). Red represents 0 (inactive).

Further discussion is suggested in [42] with multi-states for Rasgap, Kras, Rkip and miR221. Next, we implement the analysis with the same process in a non-Boolean network.

Based on the non-Boolean modeling data in [42], we construct a new EGFR network model.

$$f_{x_1} = -E \tag{6.7}$$

$$f_{x_2} = 2R + E + 2RE + ER^2 \tag{6.8}$$

$$f_{x_3} = 1 + 2x_2 + x_2^2 + 2x_1x_2 + 2x_1x_2^2 \tag{6.9}$$

$$f_{x_4} = 2 + 2x_3 + x_3M + +2x_3^2 + x_3^2M + M^2x_3 + x_3^2M^2 \tag{6.10}$$

In the non-Boolean EGFR network, we should focus on equations of x_3 and x_4 as x_1 and x_2 are fixed with input parameters E and R. Same as the process for Boolean networks, from the model function (6.9) and (6.10), we can find the smallest factor-closed set to recover this network. The standard monomials should at least include $\{x_1x_2^2, x_2^2, x_3^2, x_2x_1, x_2, x_1, x_3, 1\}$. The computation of ECs and representatives associated with the standard monomials is future work in the next month.

Chapter 7

CONCLUSION

7.1. Main Results

In the previous chapters, our work is mainly focused on four areas under the setting of a finite field $\mathbb{F}$:

1. Geometric characteristics of discrete input data sets for gene regulatory networks with a discussion of their structure associated with unique and non-unique GBs. A linked position conjecture was proposed, which indicates a method to reduce the number of GBs.

2. Model estimation from the properties, such as the number of models and the upper bound of the number of model bases, is considered. Specifically, we discussed a formula for the number of GBs with 2 points and 3 points in the finite fields. Furthermore, we proposed an upper bound, which provided a significant improvement compared to an existing upper bound.

3. Model selection based on the relationship between data sets, such as the linear shift relationship or model selection based on data sets structures, such as staircases or representatives. Instead of calculating all candidate data sets, model selection can be made only by checking linear shift relationship with a representative in each EC.

4. Analysis by querying the extensive database in DoEMS. By comparison in the same equivalence class or different equivalence classes, researchers can maintain specific standard monomials by manipulating the data sets in the reports generated by DoEMS.

7.2. Future Work

There are still questions that need more attention in the future. The plan of further work is

- Generate formula for the number of Gröbner bases for three or more points data sets, similar to the case with 2 points. However, the upper bounds for cases with three or more points were found. The specific formula will be more meaningful, considering the process of model selection.

- Create a larger database which will enable researchers to do data analysis with fewer limits. Now the database for the website only provides simple cases such as $p = 2, p = 3$. If larger data sets with $p = 5$, $p = 7$ are included with more points, the website can help to analyze more application problems.

- Apply the analysis methodology of ECs and linear shifts to more applied problems as the non-Boolean EGFR network in Section 6.4 or any other applicable polynomial dynamical systems.

Appendix A

Upper bound for the Number of GBs

Below we provide tables summarizing the comparison of the maximum number of distinct reduced Gröbner bases to the predictions made by the original bound (third column) listed in Theorem 3.4 and the modified bound (last column) listed in Equation 3.3. The second column shows the actual maximum number as computed for all sets in $\mathbb{Z}_p^n$ of size given in the first column.

# of points	max # of GBs	original bound	modified bound
1	1	1	1
2	2	2.520	2.520
3	1	4.327	1
4	1	6.350	1

Table A.1: $p = 2, n = 2$

# of points	max # of GBs	original bound	modified bound
1	1	1	1
2	3	8	8
3	3	27	11.180
4	3	64	22.627
5	3	125	11.180
6	3	216	8
7	1	343	1
8	1	512	1

Table A.2: $p = 2, n = 3$

# of points	max # of GBs	original bound	modified bound
1	1	1	1
2	4	27.86	27.86
3	5	195.07	47.59
4	6	776.05	147.03
5	13	2264.94	195.07
6	12	5434.08	389.08
7	13	11388.61	471.48
8	9	21618.82	389.08

Table A.3: $p = 2, n = 4$

# of points	max # of GBs	original bound	modified bound
1	1	1	1
2	2	2.520	2.520
3	2	4.327	4.327
4	2	6.350	4.642
5	2	8.550	4.642
6	2	10.903	4.327
7	2	13.391	2.520
8	1	16	1
9	1	18.721	1

Table A.4: $p = 3, n = 2$

# of coordinates	max # of GBs	original bound	modified bound
2	1	6.350	1
3	3	64	22.627
4	6	776.047	147.033
5	8	10321.270	1024

Table A.5: $m = 4$ points and $p = 2$

Appendix B

Computational Results

B.1. Computational Results for Example 6.1

$GB_2 = \{x_4^2 + x_4, x_3x_4 + x_2 + x_3 + x_4 + 1, x_3^2 + x_3, x_2x_4, x_2x_3, x_2^2 + x_2, x_1 + x_3 + x_4\},$

$SM_2 = \{1, x_4, x_3, x_2\} \Longrightarrow f_{x_1} = 1 + x_4$

$GB_3 = \{x_4^2 + x_4, x_1 + x_3 + x_4, x_2x_4, x_2^2 + x_2, x_1x_4 + x_1 + x_2 + x_4 + 1, x_1x_2, x_1^2 + x_1\},$

$SM_3 = \{1, x_4, x_2, x_1\} \Longrightarrow f_{x_1} = 1 + x_4$

$GB_4 = x_1 + x_3 + x_4, x_3^2 + x_3, x_2x_3, x_2^2 + x_2, x_1x_3 + x_1 + x_2 + x_3 + 1, x_1x_2, x_1^2 + x_1,$

$SM_4 = \{1, x_3, x_2, x_1\} \Longrightarrow f_{x_1} = 1 + x_1 + x_3$

$GB_5 = x_1 + x_3 + x_4, x_3^2 + x_3, x_1x_3 + x_1 + x_2 + x_3 + 1, x_1^2 + x_1,$

$SM_5 = \{1, x_3, x_1, x_1x_3\} \Longrightarrow f_{x_1} = 1 + x_1 + x_3$

$GB_6 = x_4^2 + x_4, x_1 + x_3 + x_4, x_1x_4 + x_1 + x_2 + x_4 + 1, x_1^2 + x_1$

$SM_6 = \{1, x_4, x_1, x_1x_4\} \Longrightarrow f_{x_1} = 1 + x_4$

B.2. Computational Results for Example 6.2

$SM_2 : \{1, x_4, x_3, x_2, x_2x_4\}$

$f_{x_1} = 1 + x_4$

$f_{x_2} = 1 + x_2 + x_3 + x_4$

$f_{x_3} = x_3$

$f_{x_4} = x_4$

$SM_3 : \{1, x_4, x_2, x_2x_4, x_1\} \Longrightarrow f_{x_1} = 1 + x_4$

$f_{x_2} = x_1 + x_2 + x_4 + x_2x_4$

$f_{x_3} = 1 + x_1 + x_2x_4$

$f_{x_4} = x_4$

$SM_4 : \{1, x_4, x_3, x_2, x_2x_3\} \Longrightarrow f_{x_1} = 1 + x_4$

$f_{x_2} = 1 + x_2 + x_3 + x_4$

$f_{x_3} = x_3$

$f_{x_4} = x_4$

$SM_5 : \{1, x_3, x_2, x_2x_3, x_1\} \Longrightarrow f_{x_1} = 1 + x_1 + x_2 + x_2x_3$

$f_{x_2} = 1 + x_1 + x_3 + x_2x_3$

$f_{x_3} = x_3$

$f_{x_4} = x_1 + x_2 + x_2x_3$

$SM_6 : \{1, x_4, x_3, x_3x_4, x_1\} \Longrightarrow f_{x_1} = 1 + x_4$

$f_{x_2} = 1 + x_1 + x_3 + x_3x_4$

$f_{x_3} = x_3$

$f_{x_4} = x_4$

$SM_7 : \{1, x_4, x_3, x_1, x_1x_4\} \Longrightarrow f_{x_1} = 1 + x_4$

$f_{x_2} = x_4 + x_1x_4$

$f_{x_3} = x_3$

$f_{x_4} = x_4$

$SM_8 : \{1, x_4, x_3, x_1, x_1x_3\} \Longrightarrow f_{x_1} = 1 + x_4$

$f_{x_2} = 1 + x_1 + x_3 + x_1x_3$

$f_{x_3} = x_3$

$f_{x_4} = x_4$

$SM_9 : \{1, x_4, x_3, x_2, x_1\} \Longrightarrow f_{x_1} = 1 + x_4$

$f_{x_2} = 1 + x_2 + x_3 + x_4$

$f_{x_3} = x_3$

$f_{x_4} = x_4$

$SM_{10} : \{1, x_4, x_2, x_1, x_1x_4\} \Longrightarrow f_{x_1} = 1 + x_4$

$f_{x_2} = x_4 + x_1x_4$

$f_{x_3} = 1 + x_2 + x_4$

$f_{x_4} = x_4$

$SM_{11} : \{1, x_3, x_2, x_1, x_1x_3\} \Longrightarrow f_{x_1} = 1 + x_1 + x_2 + x_1x_3$

$f_{x_2} = 1 + x_1 + x_3 + x_1x_3$

$f_{x_3} = x_3$

$f_{x_4} = x_1 + x_2 + x_1x_3$

$SM_{12} : \{1, x_4, x_2, x_1, x_1x_2\} \Longrightarrow f_{x_1} = 1 + x_4$

$f_{x_2} = x_2 + x_1x_2$

$f_{x_3} = 1 + x_2 + x_4$

$f_{x_4} = x_4$

$SM_{13} : \{1, x_3, x_2, x_1, x_1x_2\} \Longrightarrow f_{x_1} = x_3 + x_1x_2$

$f_{x_2} = x_2 + x_1x_2$

$f_{x_3} = x_3$

$f_{x_4} = 1 + x_3 + x_1x_2$

B.3. Database for Small Gene Networks

$S_1 = \{(0,0,0,0), (0,0,0,1), (0,0,1,0), (0,0,1,1), (0,1,0,0), (0,1,0,1), (0,1,1,0), (0,1,1,1), (1,0,0,0), (1,0,0,1), (1,0,1,0), (1,0,1,1)\}$,

$S_2 = \{(0,0,0,0), (0,0,0,1), (0,0,1,0), (0,0,1,1), (0,1,0,0), (0,1,0,1), (0,1,1,0), (0,1,1,1), (1,1,0,0), (1,1,0,1), (1,1,1,0), (1,1,1,1)\}$

$S_3 = \{(0,0,0,0), (0,0,0,1), (0,0,1,0), (0,0,1,1), (1,0,0,0), (1,0,0,1), (1,0,1,0), (1,0,1,1), (1,1,0,0), (1,1,0,1), (1,1,1,0), (1,1,1,1)\}$

$S_4 = \{\{0,1,0,0\}, \{0,1,0,1\}, \{0,1,1,0\}, \{0,1,1,1\}, \{1,0,0,0\}, \{1,0,0,1\}, \{1,0,1,0\}, \{1,0,1,1\}, \{1,1,0,0\}, \{1,1,0,1\}, \{1,1,1,0\}, \{1,1,1,1\}\}$

B.4. Graph of Stable Steady States

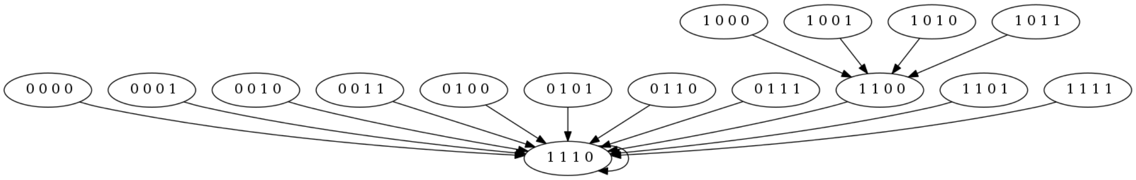


Figure B.1: EFGR Model Steady States with E=0, R=0 and M = 0

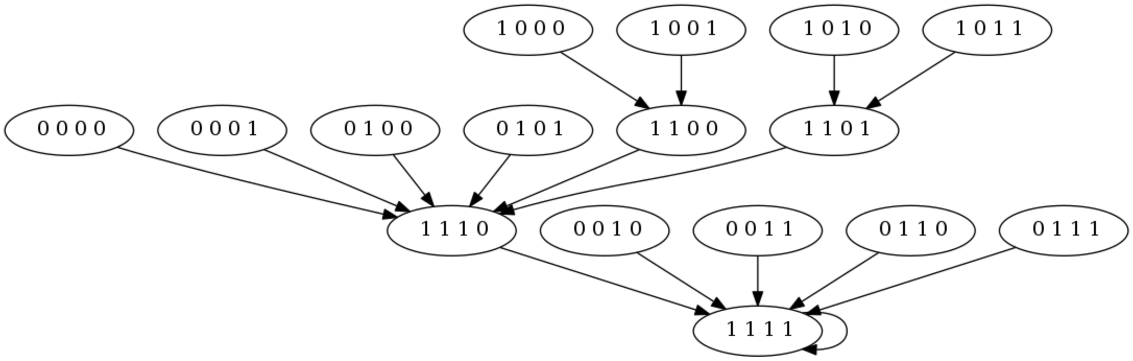


Figure B.2: EFGR Model Steady States with E=0, R=0 and M = 1

Appendix C

Python Code

```
# algorithm of fast linear shift check of two datasets
def LS_check(d1, d2, p):
    if sorted(d1) == sorted(d2):
        return ’Same␣datasets!’
    if len(d1)!= len(d2):
        return False
    pt = d2[0]
    funct= [[] for _ in range(len(d1[0]))]
    shiftfunct = []
    for d in d1:
        if d == pt:
            continue
        else:
            for i in range(len(d)):
                for k in range(1, p):
                    funct[i].append([k, (d[i] - k*pt[i])%p])
            allfunctions = list(itertools.product(*funct))
            for func in allfunctions:
                newdata = []
                for point in d2:
                    newpoint = [0]*len(d1[0])
                    for i in range(len(point)):
                        newpoint[i] = (func[i][0]*point[i] + func[i][1])%p
                    newdata.append(newpoint)
                if sorted(newdata) == sorted(d1):
                    L = len(d2[0])
                    newfunc = [[func[i][0], func[i][1]] for i in range(L)]
                    if newfunc not in shiftfunct:
                        shiftfunct.append(newfunc)
    if len(shiftfunct) == 0:
        return False
    else:
        print('Found␣liner␣shift␣from␣dataset:',d2,'to␣dataset:',d1)
        print('␣with␣linear␣shift␣functions:', shiftfunct)



class Linearshift_class:

    def __init__(self,n,m,p):
        self.n=n
        self.m=m
        self.p=p

    def subset_m(self):
        #input: number of variables(n),
        #number of state(p), number of points(m). All numbers are integer.
        #output: whole n variables p state m points datasets.
        #output is list of tuples.

        def product(*args, repeat=1):
            # product('ABCD', 'xy') --> Ax Ay Bx By Cx Cy Dx Dy
            # product(range(2), repeat=3) --> 000 001 010 011 100 101 110 111
            # input: range(p), repeat number(number of variables)
            # output: whole list of possible points with state: p and
            #number of variables: n
            pools = [tuple(pool) for pool in args] * repeat
            result = [[]]
            for pool in pools:
                result = [x+[y] for x in result for y in pool]
            for prod in result:
                yield tuple(prod)

        def powerset(iterable):
            import itertools
            from itertools import chain, combinations
            # input: whole list of possible points with state: p
            # and number of variables: n
            # output: subsets with points=m
            xs = list(iterable)
            comb = [combinations(xs,n) for n in range(len(xs)+1)]
            return chain.from_iterable(comb)
        #generate all subsets that from m=1 to m=p^n
        pro = list(product(range(self.p), repeat=self.n))
        all_subset=list(powerset(set(pro)))
        subset=list([])

        # filter out points=m datasets
        for dataset in all_subset:
            if len(dataset)==self.m:
                subset.insert(len(subset),dataset)

        return subset

    def f1(self,x1,a1,b1):
        import numpy as np
        #input: value of variable: x1, parameter of x1: a1,
        #intercept: b1. type: integer
        #output: value of shifted variable x2 type: integer
        #definition:  https://tigerprints.clemson.edu/all_dissertations/1730/
        #a linear shift function apply to value x1: x2=(a1*x1+b1)mod(p)
        x2=np.mod(a1*x1+b1,self.p)
        return int(x2)

    def check_dataset_equal(self,X1,X2):
        import numpy as np
        #input: dataset 1: x1, dataset 2: x2. type: List of List of list
        #output: if two datasets are equal. type: True and False
        a1=sorted(X1)
        a2=sorted(X2)
        return np.array_equal(a1,a2)

    def change_type(self,X):
        #input: dataset: X. type: List of Tuples
        #output: dataset: a. type: List of List of list
        t=len(X)
        m=len(X[0])
        n=len(X[0][0])
        a = list([[[0 for i in range(n)] for j in range(m)]for q in range(t)])
        for q in range(t):
            for i in range(m):
```

```
            for j in range(n):
                a[q][i][j]=int(X[q][i][j])
    return a

def get_distance_origin(self,set1):
    #input:  one shifted points set:set1. type: list of list
    #output: distance of points to origin. type: float
    #formula of distance for points (x1,x2,x3) to origin
    #is sqrt((x1-0)^2+(x2-0)^2+(x3-0)^2)
    #and sum of set's distance to origin is the sum of all points to origin.
    import numpy as np
    distancetotal=0
    distancesum=0
    for i in range(len(set1)):
        for k in range(len(set1[i])):
            distancesum+=set1[i][k]*set1[i][k]
        distancetotal+=np.sqrt(distancesum)
        distancesum=0
    return distancetotal

def findrep(self,dataset):
    #input: dataset of n variable,m number of points,
    #p number of state. type:list of list of list
    #output: rep(list of list): representative,
    #distance(float): distance to origin of each points set,
    #count(integer): number of duplicate representatives.
    #rule: calculate all points
    #and pick the set with minimun distance to origin.
    #if there are more than one minimun distance sets,
    #print there are duplicated representatives and count number.
    rep=[None];
    oldset=dataset[0]
    distance=1000
    count=0
    for set1 in dataset:
        distance_new=self.get_distance_origin(set1)
        if distance_new<distance:
            distance=distance_new
            rep=set1
    for set2 in dataset:
        if self.get_distance_origin(set2)==distance and set2!=rep:
            print('there␣are␣duplicated␣representatives',set2,'and',rep)
            count+=1

    return rep,distance,count
    #remove duplicate dataset
def remove_duplicate(self,X):
    #input: whole shifted dataset type:list of list of list
    #output: new whole shifted dataset type:list of list of list
    flag=list([0 for i in range(len(X))])
    for i in range(len(X)):
        for j in range(i+1,len(X)):
            if sorted(X[i])==sorted(X[j]):
                flag[i]=1
    X_new=list([])
    for i in range(len(X)):
        if flag[i]==0:
            X_new.append(X[i])
    return X_new

def generate_points_new(self,X):
    import itertools
    from itertools import chain, combinations
    #input: number of variables(integer):n, number of state(integer): p,
    #origin set of points(list of list): X.
    #output: all possible shifted datasets.(list of list of list)

    #generate number of points
    #m=len(X)
    #initialize p*(p-1) [all possible functions] columns and
    #n [number of variables] rows matrix
    values_each_variable=list([[0 for i in range(self.p*(self.p-1))]for
    j in range(self.n)])

    # Assign all possible shifted points to matrix
    for n1 in range(self.n):
        t=0
        for a1 in range(1,self.p):
            for b1 in range(0,self.p):
                a=list([0 for j in range(self.m)])
                for i in range(self.m):
                    a[i]=self.f1(X[i][n1],a1,b1)
                values_each_variable[n1][t]=a
                t=t+1
    # generate all permuation of pick one set from each row(each variable)
    comb=list(itertools.product(*values_each_variable))
    finalD=list([[[] for i in range(self.m)]for j in range(len(comb))])
    M=len(comb)

    #  transpose matrix to final output dataset
    for j in range(M):
        for i in range(self.m):
            for k in range(self.n):
                finalD[j][i].append(comb[j][:][k][i])

    # remove the duplicate sets in final dataset
    final_dataset=self.remove_duplicate(finalD)

    return final_dataset

def findrepresentativepoints(self,X):
    #input: number of variables(integer):n, number of state(integer): p,
    #origin set of points(list of list): X
    #output: R(list of list): represnetative,
    #distance(float): distance of set of points to origin,
    #count(integer):number of duplicate represntatives,
    # equ_class(list of list of list): equivalence class
    equ_class=self.generate_points_new(X)
    R,distance,count=self.findrep(equ_class)
    return R,distance,count,equ_class

def removeclass(self,wholedataset1,equ_class):
    #input: wholedataset(list of list of list)
    #generated whole list of n variable p states m points sets,
    #equ_class(list of list of list): equivalence class
    #output: dataset(list of list of list):
    #remove equ_class from wholedataset
    list3=[1]*len(wholedataset1)
    for i in range(len(wholedataset1)):
        for j in range(len(equ_class)):
            if sorted(wholedataset1[i])==sorted(equ_class[j]):
                list3[i]=0
    M=sum(list3)
    dataset=[None]*M
    dataset=[wholedataset1[i] for i in range(len(list3)) if list3[i]==1]
    return dataset


def findallrepresentatives(self):
    import pandas as pd
    import itertools
    from itertools import chain, combinations
    import numpy as np
    #input: n: number of variables, p: number of state, m: number of points
    #output: count2: number of equivalence classes,
    #information_list[0:count2]: summary of information of each equ_class,
    #number_class(1D-list of number): number of sets in each EC

    wholedataset1=self.subset_m()
    # generate whole dataset of n variable, p state, m points
    wholedataset1=self.change_type(wholedataset1)
    # change dataset from list of tuple to list of list of list



    #step1: generate equ_class
    X=wholedataset1[0]
    # select first set in wholedataset as origin set of points to start.
    R,distance,count,equ_class1=self.findrepresentativepoints(X)
    # compute first equ_class and representative
    information_list = [[0 for x in range(4)] for y in range(1000)]
    #initialize information list
    information_list[0][0] =R
    # assign representative to first column,first row
    information_list[0][1] =distance
    # assign distance to second column, first row

    #step2: remove equ_class
    number_list=list([])
    number_list.append(len(wholedataset1))
```

```
# initialize remain sets after removing each equ_class and
#assign first element with len(wholedataset1)
dataset=self.removeclass(wholedataset1,equ_class1)
number_list.append(len(dataset))
# remove first equ_class from wholedataset1 and
#assign the length of removed dataset to number_list

#step3: equ_class output
count2=0 # initialize number of equ_class
#generate dataframe of equ_class , assign the equ_class and
#distance of each set, and save it to txt file.
equ_class=pd.DataFrame(equ_class1)
filename1 = ’equ_p’+str(self.p)+’_n’+str(self.n)+’_m’
filename2 = str(self.m) +’_’+ str(count2)+’.txt’
distance_data=[0 for i in range(len(equ_class))]
for i in range(len(equ_class)):
    distance_data[i]=self.get_distance_origin(equ_class.iloc[i,:])
equ_class[’distance’]=distance_data
equ_class = equ_class.sort_values([’distance’])
equ_class.to_csv(filename1 + filename2)

#redo the steps 1,2,3 above until all equivalence
#classes are removed from wholedataset1.
while len(dataset)>0:
    count2=count2+1
    X=dataset[0]
    R,distance,count1,equ_class=self.findrepresentativepoints(X)
    dataset=self.removeclass(dataset,equ_class)
    information_list[count2][0]=R
    information_list[count2][1]=distance
    number_list.append(len(dataset))
    equ_class=pd.DataFrame(equ_class)
    filename1 = ’equ_p’+str(self.p)+’_n’+str(self.n)+’_m’
    filename2 = str(self.m) +’_’+ str(count2)+’.txt’
    distance_data=[0 for i in range(len(equ_class))]
    for i in range(len(equ_class)):
        distance_data[i]=self.get_distance_origin(equ_class.iloc[i,:])
    equ_class[’distance’]=distance_data
    equ_class = equ_class.sort_values([’distance’])
    equ_class.to_csv(filename1+filename2)
```

```
        count2=count2+1
        number_class=[0]*count2
        for i in range(0,count2):
            number_class[i]=number_list[i]-number_list[i+1]
        for i in range(0,count2):
            information_list[i][2]=number_class[i]

        # output summary file for all equivalence classes
        information=pd.DataFrame(information_list[0:count2])
        filename1=’equ_p’+str(self.p)+’_n’+str(self.n)+’_m’
        +str(self.m) +’_summary.txt’
        information.to_csv(filename1)

        return count2,information_list[0:count2]



#adding fewest points to get unique GB.
def calculate_total(self, newcontent, m1_number):
    string2 = newcontent[i][1]
        string3 = string2.split(’},’)
        list1 = []
        if len(string3) == m1_number:
                for i in range(len(string3)):
                        newlist = []
                        for j in range(len(string3[0])):
                                if string3[i][j] in [’0’,’1’]:
                                        newlist.append(int(string3[i][j]))
                        list1.append(newlist)
                self.total.append(list1)

def checkm4add1pt(self, m4gbnumber, m5gbnumber, m1_number, m2_number):
        m4gbnumber = str(m4gbnumber)
        #print(m4gbnumber)
        m5gbnumber = str(m5gbnumber)
        #print(m5gbnumber)
        with open(’/Users/MacBook/Downloads/p2n4.txt’) as f:
            content = f.readlines()
        #print(content[:100])
        newcontent = []
```

```
for i in range(len(content)):
    newcontent.append(content[i].split(’;’)[:2])
#print(newcontent)
self.total = []
max1= 0
for i in range(len(newcontent)):
        if len(newcontent[i][1].split(’},’)) ==  m1_number:
                if int(newcontent[i][0]) > max1:
                        max1 = int(newcontent[i][0])
        string2 = ’’
        string3 = []
        if newcontent[i][0] == m4gbnumber:
                self.calculate_total(newcontent, m1_number)
m4gb6 = self.total
print(’Maximum␣number␣of␣GB␣models:’, max1)
if int(m4gbnumber) > max1:
        print("Error!␣larger␣than␣maximum␣number␣of␣GB")
if int(m4gbnumber) == max1:
        print("this␣is␣the␣largest␣number␣of␣GB␣in␣this␣case!")


newcontent = []
for i in range(len(content)):
    newcontent.append(content[i].split(’;’)[:2])
self.total = []
for i in range(len(newcontent)):
        string2 = ’’
        string3 = []
        if newcontent[i][0] == m5gbnumber:
                self.calculate_total(newcontent, m2_number)
m5gb1 = self.total

for set1 in m4gb6:
        #print(’set1’,set1)
        for set2 in m5gb1:
                #print(’set2’,set2)
                flag = 0
                for set11 in set1:
                        if set11 in set2:
                                flag += 1
                        if flag == m1_number:
                                self.path = True
                                self.res.append((set1, set2))
        #print(self.res)
        print('end␣of␣the␣test')

def checkaddone(self, m4gbnumber, m5gbnumber, m1_number, m2_number):
        self.path = False
        self.res = []
        self.checkm4add1pt(m4gbnumber, m5gbnumber, m1_number, m2_number)

        if not self.path :
                print(self.path)
                print('There␣is␣no␣add␣one␣point␣path␣for␣'
                + str(m4gbnumber) +'␣Gbs␣' + )
                print(str(m1_number) + '␣points␣to␣' + str(m5gbnumber)
                +'␣Gbs␣' +)
                print(str(m2_number) + '␣points' )
        else:# write result to file:
                print(self.path)
                print( str(m4gbnumber) + '␣Gbs␣for␣' + str(m1_number)
                +'␣points␣to␣' +)
                print(str(m5gbnumber) + '␣Gbs␣for␣' + str(m2_number)
                +'␣points')
                f1 = str(m4gbnumber)+'gb'+str(m1_number)
                f2 = 'pts'+str(m5gbnumber)+'gb'+str(m2_number)+'pts'
                file = open('/UniqueGBmodelresults/' + f1 + f2+'.txt','w')
                file.write(str(self.res))
                file.close()
                print('file␣has␣been␣created␣in␣your␣folder')
```

BIBLIOGRAPHY

[1] Abbott, J., Bigatti, A., Kreuzer, M., and Robbiano, L. Computing ideals of points. *Journal of Symbolic Computation 30*, 4 (2000), 341–356.

[2] Abłamowicz, R. Some applications of Gröbner Bases in Robotics and Engineering. *Geometric Algebra Computing, ISBN 978-1-84996-107-3. Springer-Verlag London Limited, 2010, p. 495* (05 2010), 495–.

[3] Adams, W. W., and Loustaunau, P. *An Introduction to Gröbner Bases.* American Mathematical Soc., 1994.

[4] Alakwaa, F. M. Modeling of gene regulatory networks: A literature review. In *Journal of Computational Systems Biology* (2015).

[5] Andrews, G. E. A lower bound for the volume of strictly convex bodies with many boundary lattice points. *Transactions of the American Mathematical Society 106*, 2 (02 1963), 270–279.

[6] Babson, E., Onn, S., and Thomas, R. The Hilbert zonotope and a polynomial time algorithm for universal Gröbner bases. *Advances in Applied Mathematics 30*, 3 (2003), 529–544.

[7] Berkeley School of Information. What is data science? https://datascience.berkeley.edu/about/what-is-data-science/. Accessed: 2019-11-18.

[8] Brickenstein, M., Dreyer, A., Greuel, G.-M., Wedler, M., and Wienand, O. New developments in the theory of Gröbner bases and applications to formal verification. *Journal of Pure and Applied Algebra 213* (08 2009), 1612–1635.

[9] Cho, K.-H., Choo, S.-M., Jung, S., Kim, J.-R., Choi, H.-S., and Kim, J. Reverse engineering of gene regulatory networks. *IET Systems Biology 1*, 3 (2007), 149–163.

[10] Cox, D., Little, J., and O'Shea, D. *Ideals, Varieties, and Algorithms.* Springer Verlag, New York, 1997.

[11] Cox, D. A., Little, J., and O'shea, D. *Using Algebraic Geometry*, vol. 185. Springer Science & Business Media, 2006.

[12] Crombach, A., Wotton, K. R., Cicin-Sain, D., Ashyraliyev, M., and Jaeger, J. Efficient reverse-engineering of a developmental gene regulatory network. *PLOS Computational Biology 8*, 7 (07 2012), 1–21.

[13] Dehghannasiri, R., Yoon, B.-J., and Dougherty, E. Optimal experimental design for gene regulatory networks in the presence of uncertainty. *IEEE/ACM Transactions on Computational Biology and Bioinformatics / IEEE, ACM 12* (12 2014).

[14] Dimitrova, E., He, Q., Robbiano, L., and Stigler, B. Small Gröbner fans of ideals of points. *Journal of Algebra and Its Applications* (2019).

[15] Dimitrova, E., Jarrah, A., Laubenbacher, R., and Stigler, B. A Gröbner fan method for biochemical network modeling. In *Proc. 2007 Internat. Symp. Symbolic Algebraic Computat.* (2007), ACM, pp. 122–126.

[16] Eder, C., and Faugère, J.-C. A survey on signature-based algorithms for computing Gröbner bases. *Journal of Symbolic Computation 80* (07 2016).

[17] Faugère, J. C. 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* (New York, NY, USA, 2002), ISSAC '02, ACM, pp. 75–83.

[18] Friedman, N., Linial, M., Nachman, I., and Pe'er, D. Using Bayesian networks to analyze expression data. *Journal of Computational Biology 7* (07 2004), 3–4.

[19] Frolova, A. Overview of methods of reverse engineering of gene regulatory networks: Boolean and Bayesian networks. *Biopolymers and Cell 28* (11 2011), 163–170.

[20] Fukuda, F., Jensen, A., and Thomas, R. Computing Gröbner fans. *Mathematics of Computation 76*, 260 (2007), 2189—-2212.

[21] Gao, S., Platzer, A., and Clarke, E. M. Quantifier elimination over finite fields using Gröbner bases. In *Algebraic Informatics* (Berlin, Heidelberg, 2011), F. Winkler, Ed., Springer Berlin Heidelberg, pp. 140–157.

[22] Gat-Viks, I., and Shamir, R. Chain functions and scoring functions in genetic networks. *Bioinformatics 19*, suppl 1 (2003), i108–i117.

[23] Goodwin, B. *Temporal Organization in Cells.* Academic Press, 1963.

[24] Goutsias, J., and Lee, N. Computational and experimental approaches for modeling gene regulatory networks. *Current Pharmaceutical Design 13*, 14 (2007), 1415–1436.

[25] Gustafsson, M., Hornquist, M., and Lombardi, A. Constructing and analyzing a large-scale gene-to-gene regulatory network-lasso-constrained inference and biological validation. *IEEE/ACM Trans. Comput. Biol. Bioinformatics 2*, 3 (July 2005), 254–261.

[26] He, Q. *Algebraic Geometry Arising from Discrete Models of Gene Regulatory Networks*. PhD thesis, Clemson University, 2016.

[27] He, Q., Dimitrova, E. S., Stigler, B., and Zhang, A. Geometric Characterization of Data Sets with Unique Reduced Gröbner Bases. *Bulletin of Mathematical Biology 81*, 7 (2019), 2691–2705.

[28] Karlebach, G., and Shamir, R. Modelling and analysis of gene regulatory networks. *Nature Reviews Molecular Cell Biology 9* (10 2008), 770–780.

[29] Laubenbacher, R., and Stigler, B. A computational algebra approach to the reverse engineering of gene regulatory networks. *J. Theor. Biol. 229*, 4 (2004), 523–537.

[30] Lin, Z., Xu, L., and Wu, Q. Applications of Gröbner bases to signal and image processing: A survey. *Linear Algebra and its Applications 391* (2004), 169–202.

[31] Makarim, R. H., and Stevens, M. M4GB: An efficient Gröbner-basis algorithm. In *ISSAC* (2017).

[32] Markham, A., and Trigoni, N. The automatic evolution of distributed controllers to configure sensor network operation. *The Computer Journal 54*, 3 (2011), 421–438.

[33] Mora, T., and Robbiano, L. The Gröbner fan of an ideal. *J. Symb. Comput. 6* (10 1988), 183–208.

[34] Ogundijo, OE, E. A., and Wang, X. Reverse engineering gene regulatory networks from measurement with missing values. *EURASIP J Bioinform Syst Biology 2* (01 2017).

[35] Onn, S., and Sturmfels, B. Cutting corners. *Advances in Applied Mathematics 23*, 1 (1999), 29–48.

[36] Ozbudak, E., Thattai, M., Lim, H., Shraiman, B., and van Oudenaarden, A. Multistability in the lactose utilization network of *Escherichia coli*. *Nature 427* (2004), 737–740.

[37] Petri, T., Mikko, K., Garry, W., and Eero, C. Analysis of gene expression data using self-organizing maps. *FEBS Letters 451* (05 1999), 142–146.

[38] Raeymaekers, L. Dynamics of Boolean networks controlled by biologically meaningful functions. *Journal of theoretical biology 218* (11 2002), 331–41.

[39] Robbiano, L. Gröbner bases and statistics. In *Gröbner Bases and Applications* (New York, 1998), B. Buchberger and F. Winkler, Eds., vol. 251 of *London Mathematical Society Lecture Notes Series*, Cambridge University Press, pp. 179–204.

[40] Santillán, M. Bistable behavior in a model of the *lac* operon in *Escherichia coli* with variable growth rate. *Biophysical Journal 94*, 6 (2008), 2065–2081.

[41] Schlitt, T., and Brazma, A. Current approaches to gene regulatory network modelling. *BMC Bioinformatics 8* (09 2007), 1471–2105.

[42] Steinway, S. N., Wang, R.-S., and Albert, R. Discrete Dynamic Modeling: A Network Approach for Systems Pharmacology. In *Systems Pharmacology and Pharmacodynamics* (2016), Springer International Publishing, pp. 81–103.

[43] Stigler, B. Polynomial Dynamical Systems in Systems Biology. In *Proceedings of Symposia in Applied Mathematics* (2007), vol. 64, p. 53.

[44] Stigler, B., and Zhang, A. The number of Gröbner bases in finite fields. *AWM Proceedings of 2019 Research Symposium* (2020).

[45] Tsakanikas, P., Manatakis, D., and Manolakos, E. S. Machine learning methods to reverse engineer dynamic gene regulatory networks governing cell state transitions. *bioRxiv* (2018).

[46] Veliz-Cuba, A., and Stigler, B. Boolean models can explain bistability in the *lac* operon. *J Comput Biol. 18*, 6 (2011), 783–794.

[47] Vera-Licona, P., Jarrah, A., Garcia-Puente, L. D., McGee, J., and Laubenbacher, R. An algebra-based method for inferring gene regulatory networks. *BMC Systems Biology 8*, 1 (2014), 37.

[48] Wahde, M., and Hertz, J. Coarse-grained reverse engineering of genetic regulatory networks. *Bio Systems 55 1-3* (2000), 129–36.

[49] Wong, P., Gladney, S., and Keasling, J. Mathematical model of the *lac* operon: Inducer exclusion, catabolite repression, and diauxic growth on glucose and lactose. *Biotechnology Progress 13*, 2 (1997), 132–143.

[50] Zhang, A., and Stigler, B. DoEMS: Linking Design of Experiments and Model Selection. Available at `https://s2.smu.edu/doems`.