5-2025

CONSTRUCTING BINARY ENCODING MATRICES FROM JOINED GRAPHS

Joshua Avalos
California State University - San Bernardino

Constructing binary encoding matrices from joined graphs

A Thesis

Presented to the

Faculty of

California State University,

San Bernardino

In Partial Fulfillment

of the Requirements for the Degree

Master of Arts

in

Mathematics

---

by

Joshua Avalos

May 2025

Constructing binary encoding matrices from joined graphs

A Thesis

Presented to the

Faculty of

California State University,

San Bernardino

---

by

Joshua Avalos

May 2025

Approved by:

Youngsu Kim, Committee Chair

Jeremy Aikin, Committee Member

Rolland Trapp, Committee Member

Dr. Madeleine Jetter, Chair, Department of Mathematics

Dr. Corey Dunn, Graduate Coordinator

Abstract

Codes and technology are part of our daily lives and allow the modern world to function, and for us to have conveniences in our lives such as smartphones that can be used to privately call people on the other side of the planet, and for secure access to the internet. In this thesis we will explore the construction of binary codes created by vertex-edge incidence matrices of planar graphs. The Hamming $(7, 4)$ code was an incredible code that allowed the detection and correction of errors after receiving them through a transmission. We will explore the possibility of the creation of generator matrices that can detect and possibly correct errors that occur due to interference caused between the message being sent and it being received. Here, we will provide a proof for tree and cycle graphs, as well as a basis for other graphs and future work.

Acknowledgements

I would like to express my sincere gratitude to my advisor Dr. Youngsu Kim for all of his time and patience during this process, and for not giving up on me. Without Dr. Kim's help and motivation, I would not have been able to stay in the masters program and complete this thesis.

I would also like to acknowledge my Committee advisors Dr. Rolland Trapp, and Dr. Jeremy Aikin for their support and help during my time at CSUSB and for always being open for questions and advice. As well as all of the members of the "alchemist's valorant server" discord server for their emotional and mental support during this entire process.

This Thesis was supported in part by the high performance computing resources provided by Information Technology Services and Academic Technologies & Innovation at California State University San Bernardino.

Last, but not least I would like to acknowledge the Mathematic Department staff for their open door policy and to the College of Extended Learning for helping me to afford to stay in the masters program and to finish what I started, Thank You!

Table of Contents

Abstract **iii**

Acknowledgements **iv**

List of Figures **vi**

1 Introduction **1**

2 Algebraic Coding Theory **4**

3 Graph Theory **13**

4 Codes Based On Incidence Matrices **18**
4.1 Tree Graphs: Over $\mathbb{F}_2$ 18
4.1.1 Trees 22
4.2 Cycle Graphs 25

5 Joining **29**
5.1 Parallel Connection 31
5.2 Direct Sum 47
5.2.1 Examples 51
5.2.2 Joining To End Leaf 54
5.2.3 Joining To Shared Vertex 56

6 Conclusion **61**
6.1 Possible Outcomes 64

7 Expository Work **66**
7.1 Possible Generator Matrices From Cycle Graphs 68

Bibliography **76**

List of Figures

3.1 Graph C_3 . . . 14
3.2 Graph C_3 With Loop And Parallel Edges . . . 15
3.3 Graph G . . . 15
3.4 Path Subgraph of G Examples . . . 16
3.5 Cycle Subgraph of G Examples . . . 16

4.1 P_2 Incidence Matrix And Row Reduced Echelon Form . . . 19
4.2 The Graph of P_2 . . . 19
4.3 The Graph of S_4 . . . 22
4.4 The Graph of C_3 . . . 25
4.5 Single And Double Error Transmissions . . . 28

5.1 Graph A . . . 30
5.2 Graph B . . . 30
5.3 C_3 Incidence Matrix . . . 33
5.4 C_3 Joining Matrix . . . 33
5.5 House Graph . . . 40
5.6 $C_{3,3}$ Direct Sum . . . 47
5.7 The Graph of $C_{4,4}$ . . . 52
5.8 The Graph of P_3 . . . 53
5.9 The Graph of $C_{4,4,4}$ . . . 54
5.10 The Graph of DS_4 . . . 56

6.1 Graph E . . . 62

Chapter 1

Introduction

Encoding is a part of nearly every interaction we have with technology. There are multiple reasons to encode data, but I believe the two main reasons are to have a high level of reliability, and to protect your information from errors caused by natural causes or some other form of interference during the transmission of the information. The topics discussed here will allow us to get a basic understanding of what codes are, how code generation works, as well as how effective the codes can be at detecting and correcting errors.

The main topic of this thesis shall focus on *algebraic coding theory*, which is the study of transmitting encoded messages across a noisy channel that can cause errors during transmission. Coding theory is a relatively new subject in mathematics that was created during the 20th century and became more broadly known after 1948 with the publication of the article "A Mathematical Theory of Communication" by Claude Shannon. One of the first applications of coding theory was with punch card reading machines that were sometimes used to count votes during elections. The codes used were able to count the punch cards and improved versions could detect errors during the counting in order to have more accurate counts without human error. Counting punch cards in this manner seems simple now, but when it was first introduced it allowed for votes to be counted faster and with fewer errors during critical election times. Although not all devices were the same, the overall concept was that when the voters punch cards were made, the machine would also have a reading portion that matched the voting cards hole punch pattern with electrical contacts. The voters cards would be placed on the

reading part of the machine, and the machine would determine which hole was punched through by closing an electrical circuit through any hole punches in the voters card. The machine would then use this grid of closed circuits and go through its process to output a complete summary and count of the vote. The hole punches in the cards can be seen as an array or matrix of $1s$ where a hole punch existed and the circuit closed, and $0s$ where a hole didn't exist and the circuit could not close.

In coding theory, a code encodes a *message word* m of length k in the form of a vector $m = [m_1 m_2 \dots m_k]$ into a *codeword* c of length n in the form $c = [c_1 c_2 \dots c_n]$. Using a $k \times n$ matrix called a *generator matrix*, message words can be encoded into codewords by using matrix multiplication. In the 21st century, Coding Theory is often used during the transmission of (binary) messages through many different types of channels, and occurs all around us without us ever even noticing. One instance where this occurs is when a computers Random Access Memory (RAM) modules interact with the Central Processing Unit (CPU) of the computer. Binary messages are sent back and forth between the RAM and CPU, and in order for the CPU to receive the correct messages from the RAM, the messages are encoded before transmission and decoded by the CPU using a code when they are received.

The ability to encode a message with redundant information allows an operator on the receiving end of the transmitted encoded message to determine if the received word is correct, or if an error may have occurred during transmission. Encoded messages are not fool proof however, there can be interference, or noise, in the channel used to send the message, such as electromagnetic interference or even interference cause by events out of human control such as solar flares and ionized particles flying through space. This noise can change components of the codeword from a zero into a one, or from a one into a zero. This could change the message portion, or the redundancy portion of the message sent depending on how the interference affects the word during transmission. Thus being able to encode a message into a form that can be used to detect the errors in itself after being transmitted, and that can have its error corrected, is the best scenario but can be difficult to do with reasonable message lengths and number of message words.

Codes that can detect if an error occurred but cannot determine the location of the error still have their uses, but only work well if they have an operator on the receiving end of the codewords that is monitoring the incoming words and decoding them them-

selves. If enough errors occur and the transmitted word is changed into another codeword that would not alert to errors, then the operator is necessary and must determine if the decoded message is the intended message. For cases such as these, the operator can determine if the message received makes sense for the situation, and if unsure they should request for the message to be retransmitted in order to compare the received codes and make a more informed decision.

In Chapter 2 we will see an introduction into algebraic coding theory. There we will see the components that make up a code from the message words that are created in the beginning to how message words are encoded into codewords. We will also go over the decoding methods that will be used at the end of a transmission process by the machine or operator on the receiving end.

In Chapter 3 we will see an introduction to graph theory. The graphs used in this Chapter are general tree graphs, special tree graphs, as well as cycle graphs. Using these graphs we will obtain their vertex-edge incidence matrices and row reduced them to their row reduced echelon form. These incidence matrices will become the base for the generator matrices of codes that are constructed in Chapter 4.

Using the incidence matrices that have been reduced into their row reduced echelon forms, in Chapter 4 we will learn about the types of codes that the Chapter 3 graphs incidence matrices can create, as well as the decoding processes that we will use for these codes. After going over the benefits and downsides of the different codes the graphs created, we will learn about the decoding process used after an encoded message is received and how decoding using the redundancy portion of the code can help locate the location of the error.

After seeing the types of codes the simple graphs create in Chapter 4, in Chapter 5 we will see what the type of codes the incidence matrices of graphs connected by a common edge or by a common vertex can construct. We will still use the graphs that were available to us in previous chapters, but we will be able to construct more complex graphs composed of many smaller subgraphs that will produce a code that offers better error detection and error correction than those from previous Sections.

Chapter 2

Algebraic Coding Theory

Algebraic coding theory is the study and creation of error-detecting and error-correcting codes for transmitting encoded messages across noisy channels that can cause errors during transmission.

Most computers and information that travels across electrical or some other transmission line is send in binary, which contains only two elements. Let $\mathbb{B}^n$ be the set of all binary words, denoted as row vectors, such that the closed field of order 2, $\mathbb{F}_2 = \mathbb{B} = \{0, 1\}$, and all binary words are made-up of n consecutive $0s$ or $1s$ [Fra67, P. 149-151]. Since $\mathbb{B}^n$ can contain all binary words, we must select an n to limit the length and number of words. By limiting the words length, we obtain a subset of $\mathbb{B}^n$ that is will form the set of vectors we transmit as messages. First we select what number of messages we require, and if we fix the length of the words containing our message information to length k, then the maximum number of words possible in the set of messages is 2^k. These length k vectors are called *message words* denoted by, $m_i = (w_1 \dots w_{k-1} w_k)$, such that the components that make it up are $w = 1$ or 0, and $\{m_i \in M \mid 1 \leq i \leq 2^k\}$ [Fra67, P. 151].

Now although we can transmit the messages across a noisy channel alone we will not have any way of knowing if the message was received without any errors. Unless the sender can communicate with the receiver in some other way, but that is assumed to not be possible[1]. In order to add redundancy of some form to the message words, we must encode the message words using a matrix of an appropriate size. A *standard generator*

[1] If it was, this redundancy would not be necessary, since the two operators could confirm messages and errors among themselves.

matrix is a size $k \times n$ matrix, such that k columns of the matrix are identity columns and the remaining $n-k$ columns are non-identity columns [Fra67, P. 153]. By left multiplying the length k message words with a generator matrix G, we obtain a length n vector known as a codeword. A *codeword* is denoted $\{c_i \in C \mid c_i = (u_1 \dots u_{k-1} u_k \dots u_{n-1} u_n)\}$, and the first k components of a codeword c_i are the message word m_i [Fra67, P. 151].

A *code* is a closed set C of codewords, $C = \{(c_1 c_2 \dots c_{n-1} c_n), (c_1^2 \dots c_{n-1}^2 c_n^2), \dots (c_1^n \dots c_{n-1}^n c_n^n)\}$, and are closed under the binary operation, word addition, such that the sum of any two codewords results in a codeword already in the set, and all binary codes of length n are subsets of $\mathbb{B}^n$. *Word addition* is the process of adding two or more words in a code C such that when taking the sum of the words c_1 and c_2, the i^{th} element of c_1 is added to the corresponding i^{th} element of c_2, and the resulting row vector is also a codeword contained in C [Fra67, P. 151]. Word addition is assumed to be done by adding, and then using the remainder after dividing by 2 resulting in either zero or one, *modulo* 2, unless otherwise specified.[2]

A code is composed of two main components: message-words, and generator matrices. Message-words will be in the form of row vectors, and their size is $1 \times k$, such that k is equal to the number of rows in the message-words corresponding generator matrix. The quantity of messages that can be sent with a single code will be determined by the length of the message words, as well as the codewords that are available after encoding since the codewords set must be closed. Generator matrices are denoted by G, and are used to encode message-words into codewords for transmission through noisy channels. A generator matrix G has size $k \times n$ and is composed of two parts: An identity portion with size $k \times k$ denoted by I_k which is how our message-word is sent, and a non-identity portion with size $k \times (n-k)$ that is the encoding portion of the code. The non-identity columns of the generator matrix are what is known as the *redundancy-portion* or *parity-check portion* of the codeword [Fra67, P. 151]. The parity-check portion of the matrix can be written as an equation of the parity column composed of the previous identity columns, such as the third column x_3 from Remark 1, the parity column $x_3 = x_1 + x_2$.

Example 2.1. Let G represent the generator matrix of a $(3, 2)$ code with length $k = 2$ message words, and length $n = 3$ codewords, and the message word we wish to transmit

[2]This is done in the same way vector addition is done such that the i^{th} element of one vector is added to the i^{th} element of another vector.

is $m_1 = (10)$.

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

Left multiplying, $m_1 \cdot G = c_1$

$$c_1 = \begin{bmatrix} 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 \end{bmatrix}.$$

The first two components of c_1 are equal to the message word m_1, and the third column of c_1 is the redundancy portion of the codeword.

Parity check equations are composed of elements from the message words. The equations are used to determine if a component from the message portion of the codeword had any error during transmission. If there was an error, the equations can be used to determine if the error can be located and corrected without retransmitting the codeword.

Example 2.2. Consider a 2×3 generator matrix

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

This code contains four possible message words in a set denoted

$$MW = \{(00), (10), (01), (11)\},$$

and the codewords that are created are

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

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

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

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

The closed set of codewords is

$$C = \{(000), (101), (011), (110)\}$$

and contains all of the codewords produced by the generator matrix G. We can see the codewords are closed under word addition

$$\begin{aligned}
(000) + (101) &= (101) \\
(000) + (110) &= (110) \\
(000) + (011) &= (011) \\
(000) + (000) &= (000) \\
(101) + (110) &= (011) \\
(101) + (011) &= (110) \\
(101) + (101) &= (000) \\
(110) + (011) &= (101) \\
(110) + (110) &= (000) \\
(011) + (011) &= (000).
\end{aligned}$$

Thus we can see above that all possible sum combinations of codewords in C results in a codeword in C, and the set C is closed under word addition.

Remark 1. From the generator matrix G above Example 2.2, we notice that the first two columns of G, lets call c_1 and c_2, make up the identity message portion of the generator matrix and the third columns, lets call c_3, makes up the redundancy or parity-check portion of the matrix. The third column, c_3, is written as an equation composed of the message columns, the first two columns c_1 and c_2. The single parity check column would form the parity-check equation $x_3 = x_1 + x_2$, where x in the equation represents the columns, and would be used to determine if an error occurred in one of the two message components during transmission.

Properties of codewords in a code include the *Hamming weight* of a codeword u, denoted $wt(u)$, which is the number of components that are 1 [Fra67, P. 151]. A second

property of codewords is the *distance* between a pair of words u and v, denoted by $d(u, v)$ [Fra67, P. 151]. The distance between two codewords is the number of components in which the entries of the two codewords u and v are different, such that one word has a 0 where the other has a 1.

The sets of codewords have properties of their own. The *minimum distance* of a code C is the smallest distance between any two distinct codewords u and v in a code C, denoted $d(C)$ [Fra67, P. 151]. This is useful in different areas of a code for different reasons. In terms of the entire code, two words having a higher weight signifies there is a lower chance of us confusing a word that underwent multiple errors with another unrelated codeword. If our message words are not yet labeled to mean something or represent an action, the words with the highest difference can be used as pair such as one meaning yes and the other no, for easier and faster decision making when decoding transmission with errors.

Let C be an (n, k) *linear code*. C is called an (n, k) *binary group code*[3] if it satisfies the following

- C satisfies the additive property and is closed.
- C satisfies the scalar multiplicative properties required for a subspace, where the only scalars are, 0 and 1.
- codewords are thought of in terms of row vectors with n components that we identify with the words in C

Example 2.3. Consider the $(3, 2)$ code in Example 2.2, and its four codewords contained in C. The code C is a $(3, 2)$ binary group code with order $2^k = 2^2 = 4$. The weight of the codewords that are composed with $1s$ is $wt(c) = 2$, and the last codeword is the codeword composed of all zeroes and has zero weight, $wt(c_0) = 0$.[4] The distance between any of the codewords in C is two, so any codewords u and v in C will have distance $d(u, v) = 2$. We can use the closed set of four codewords to double check that all of the codewords have $d(u, v) = 2$. Thus, the two words that were added must have had two components that did not match and resulted in a codeword with $wt(c_i) = 2$, or the two codewords

[3] A (n, k) *binary group code* is a subgroup C^* of $\mathbb{B}^n$ [Fra67, P. 151-152]. The order of C^* must be a divisor of the order 2^n of B^n, so the order of C^* must be 2^k for some integer k where $0 \leq k \leq n$.

[4] Since all three of the codewords have two $1s$ and a single 0 as components

being added were exactly the same and the components all added to zero and resulted in codewords with $wt(c_i) = 0$.

The error detection capabilities of this $(3, 2)$ code are very lacking. The code contains a single redundancy column that is composed of the first two message columns. The code has a minimum distance of 2, and does not contain enough parity check equations to correct any errors when decoding. The code is only capable of detecting if a single error occurs since any change in one of the first two components would result in a different output in the redundancy column of the codeword, and the received word with one error would not be a codeword. Even if a single error or odd number of errors occurred and we received a word that is not a codeword, we would not be able to determine whether the error occurred in one of the message column or in the redundancy column since the single parity check equation would not allow our decoding methods to determine the location of the error.

Since the code has a minimum distance of 2, any codeword that undergoes an even number of errors will result in another codeword, since it would be similar to adding codewords which change two arbitrary components. This is dangerous and inefficient since it can allow an operator to believe the incorrect words he received with errors, are correct. Thus, the $(3, 2)$ code can detect single errors, but is incapable of correcting errors caused during transmission.

Decoding a received word w by selecting a codeword at minimum distance from w is known as *nearest-neighbor decoding* [Fra67, P. 154]. In some cases, more than one codeword might be at the same minimum distance from w. When this occurs, it is up to the receiving operator to determine the best fitting message word of those available. If multiple message words are at the same distance and multiple codewords work for the situation, then the operator should request retransmission of the codeword to avoid incorrect assumptions.

Definition 2.4 ([Fra67, P. 154])**.** For an (n, k) code, the *parity-check matrix* H is a size $(n - k) \times n$ matrix whose rows are composed of the $(n - k)$ parity check equations. The i^{th} row of H contains the n coefficients of $x_1, x_2, \ldots,$ and, x_n in the i^{th} parity equation.[5]

Since nearest neighbor decoding can be very limiting and can result in multiple

[5] $1s$ are placed in columns where the parity equation contains the components, and components that are not contained in the parity equation contain $0s$ in their corresponding columns.

possible correct message words, we will use another error checking method to attempt to detect and correct errors which uses the parity check equations. Given a parity check matrix H, and a received word w, the steps of *parity-check matrix decoding* are

1. Compute Hw
2. If Hw is a zero vector, decode as the first k characters of w[6].
3. If Hw is the j^{th} column H then
   (a) If $j > k$, then decode as the first k characters of w.
   (b) Otherwise decode as the first k characters with the j^{th} character changed.
4. If neither item 2 nor item 3 occurs, then more than one error was made; ask for retransmission [Fra67, P. 156].

Parity check matrix decoding typically allows for the detection and correcting of a *single error*, which is when only one interchange of the characters 0 and 1 occurs during transmission of a codeword [Fra67, P. 150]. A *syndrome*, which is the product of a word vector[7] from $\mathbb{B}^n$ and its parity-check matrix H, allows for the detection of error patterns. Patterns such as repetitive errors cause by a constant noise that causes an error in the same element of multiple different codeword transmitted.

By using more than one parity-check equation we cannot only detect, but also correct a single-error transmissions of a codeword. During single-error transmissions, the distance between the codeword (before transmission) u, and the received word v will be one, $d(u, v) = 1$. If we can make the minimum distance of the code be three(3), then the correct codeword (before transmission) will be the unique codeword at distance one(1) from the received word; In the case of $(n, k) - codes$, there is the equation $d = 2t + 1$, where d is the minimum distance of the code and t represents the number of errors that can be corrected [Fra67, p. 156-158]. This method of encoding was developed by Richard Hamming in 1948 after becoming frustrated with an early model of a computer that could not correct error and would simply output the wrong transmission and move on to the next.

[6]Exercise 29 in the citation shows that w is a codeword if and only if Hw is the zero column vector
[7]Not a codeword, since not all length n words received after transmission are codewords.

Example 2.5. Consider the Hamming $(7, 4)$ code that has sixteen unique 4-component message words that are encoded into sixteen 7-component codewords by a generator matrix

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

Let x_1, x_2, x_3, and x_4 be the first four components (identity columns) of the generator matrix G. Let x_5, x_6, and x_7 be the remaining columns of the redundancy portion of the generator matrix such that they create parity check equations

$$x_5 = x_1 + x_2 + x_3,$$
$$x_6 = x_1 + x_3 + x_4,$$
$$x_7 = x_2 + x_3 + x_4.$$

In order to create the parity-check matrix H from this code, we will need the parity-check equations. The matrix will be composed of the parity-check equations such that each equation will form a row in the parity-check matrix with ones in the components in the equation and zeros in the components missing from the equation, for each equation. Consider the parity-check matrix H, and the transmitted word w[8]

$$H = \begin{bmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 \end{bmatrix}, w^T = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 1 & 1 \end{bmatrix}.$$

$$Hw = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} = \text{third column of } H \text{ by Definition 2.4.}$$

and thus we can decode the first four components of the received word and then change the third component of it. Thus the message that we were intended to receive should be

$$\begin{bmatrix} 1 & 1 & 0 & 0 \end{bmatrix}.$$

[8] w is written as it transpose due to formatting. w is a column vector

When decoding codewords that were received after transmission, there are a couple of options when deciding how to decode the received words. Since the generator matrix is what does the encoding portion of the code, it must be protected and be kept away from anyone who will not be encoding or decoding messages. Because of this however, it is also important for both the sender and the receiver of the codewords to have their own copy of the generator matrix, the parity-check equations, the parity-check matrix, or most preferably, a copy of all three.

Chapter 3

Graph Theory

Graph theory is the study of the properties and applications of graphs. A *graph* G is consists of a set $V(G)$ of *vertices* and $E(G)$ of *edges* [Oxl11, P. 3]. If $e \in E(G)$, and v_1 and v_2 are end vertices of edge e, then we say that v_1 and v_2 are *neighbors* or *adjacent*, and that edge e is *incident* with v_1 and v_2. Graph theory has many different applications. A few examples are mapping gene structures for genetic biology, calculating efficient work schedules for employees of companies, and constructing efficient routes for connecting network servers and homes together by internet service providers.

For example, if we wished to use the least amount of cables to interconnect a group of servers, first we can let the servers represent vertices, and the cables represent the edges. Then we would be able to model the construction of the servers and by constructing different models, and removing unnecessary or redundant edges we can reduce the model (graph) to be as small as possible while still maintaining the necessary connections between servers.

The number of edges that are incident to a vertex, is equal to the *degree of a vertex*, and vertices who are incident to only one edge have a degree of one, and are called *leaves* [Lev21, p. 242-243]. An edge that is incident to only one vertex, and contains the same edge as both of its end vertices is called a *loop*[1]. Consider the distinct edges e_1 and e_2, and distinct vertices v_1 and v_2 in the arbitrary graph G. If vertices v_1 and v_2 make up the two end vertices of e_1, as well as the two end vertices of e_2, then edges e_1 and e_2 are called *parallel edges* [Oxl11, P. 3]. [2]

[1] For this thesis, loops will not be allowed in any graph or work done

[2] Parallel edges are also not allowed in any graphs in the thesis.

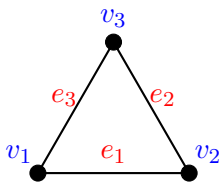


Figure 3.1: Graph C_3

Example 3.1. Consider the cycle graph in Definition 3.2 with three vertices and three edges C_3. Here Graph C_3 has the set of edges $E = \{e_1, e_2, e_3\}$ and the set of vertices $V = \{v_1, v_2, v_3\}$.

A *walk* in a graph is a sequence $v_1e_1v_2e_2 \cdots v_{k-1}e_kv_k$ such that $v_0, v_1, \ldots, v_k$ are vertices, $e_1, e_2, \ldots, e_k$ are edges, and each vertex or edge in the sequence, except v_k, is incident with its successor in the sequence [Oxl11, P. 3-5]. Suppose that the vertices, $v_1, v_2, \ldots,$ and v_k are distinct. Then the edges, $e_1, e_2, \ldots, e_k$ are also distinct and the walk is a *path*, also called a *path graph* P. A walk allows for vertices to be repeated in the sequence while a path does not, and all edges and vertices in a path can only be used once in the sequence.

All graphs used in this thesis are connected graphs, and a graph is *connected* if each pair of distinct vertices is joined by a path [Oxl11, P. 5]. A *simple graph*, is a connected graph that contains no loops or parallel edges [Oxl11, P. 3].

Definition 3.2 ([Oxl11, P. 5])**.** If P is a (u, v) path in a graph G, and e is an edge of G that connects vertices u and v but is not in P, then the *subgraph* of G whose vertex set is $V(P)$ and whose edge set in $E(P) \cup \{e\}$ is called a *cycle* and the subgraph which contains an equal number of edges and vertices, and a single cycle is called a *cycle graph*.

A graph such that any two vertices are connected by exactly one edge and contains no cycles is called a *tree graph*, and is denoted by T_n for a tree graph with n vertices [Lev21, p. 243]. There exist special cases of tree graphs that are shown here. One is a *spoke graph* which is a tree graph such that there exists a fixed vertex, v_1, that is connected all other vertices, $v_2, v_3, \ldots, v_k$, in the graph by unique edges $e_1, e_2, \ldots, e_{k-1}$ such that no loops, cycles, or parallel edges exits.

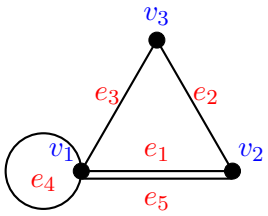


Figure 3.2: Graph C_3 With Loop And Parallel Edges

Example 3.3. By adding the edge e_4 that is incident to vertex v_1 and vertex C alone, there now exists a loop in the new graph. Edge e_4 in Chapter 3 is a loop. Looking at Chapter 3, the distinct edges e_1 and e_5 share the same end vertices, v_1 and v_2. Thus e_1 and e_5 are parallel edges.

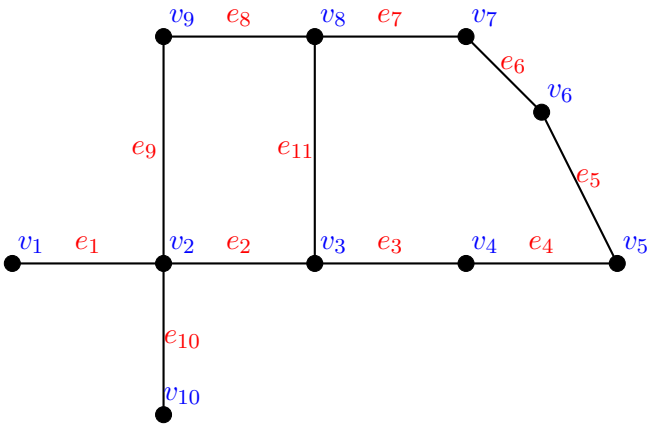


Figure 3.3: Graph G

Example 3.4. Consider the graph G above with the vertex set $V(G) = \{v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8, v_9, v_{10}\}$ colored in blue, and edge set $E(G) = \{e_1, e_2, e_3, e_4, e_5, e_6, e_7, e_8, e_9, e_{10}, e_{11}\}$ colored in red. There are multiple different subgraphs of G that form paths, and thus many different subsets of edges that form paths. Some examples of that are the subset $E_1(G) = \{e_1, e_2, e_3, e_4, e_5, e_6, e_7, e_8\}$ with its vertex subset $V_1(G) = \{v_1, v_2, v_3, v_4, v_5, v_6,$

$v_7, v_8, v_9\}$ that form a path P_1. The edge subset $E_2(G) = \{e_1, e_3, e_4, e_5, e_6, e_8, e_9, e_{11}\}$ and its corresponding vertices $V_2(G) = \{v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8, v_9\}$ form another unique path P_2.

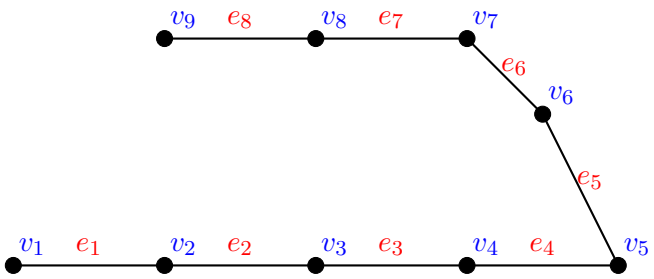


Figure 3.4: Path Subgraph of G Examples

The of subset of edges $E_3(G) = \{e_2, e_8, e_9, e_{11}\}$, and vertices $V_3 = \{v_2, v_3, v_8, v_9\}$ form a cycle graph C_4. The edge set $E_4(G) = \{e_3, e_4, e_5, e_6, e_7, e_{11}\}$, and vertex set $V_4 = \{v_3, v_4, v_5, v_6, v_7, v_8\}$ form the cycle graph C_6. Another cycle subgraph contained in G comes from the edge subset $E_5(G) = \{e_2, e_3, e_4, e_5, e_6, e_7, e_8, e_9\}$ and vertex set $V_5 = \{v_2, v_3, v_4, v_5, v_6, v_7, v_8, v_9\}$. We can see that each of these subsets contains a single cycle and form simple cycle graphs.

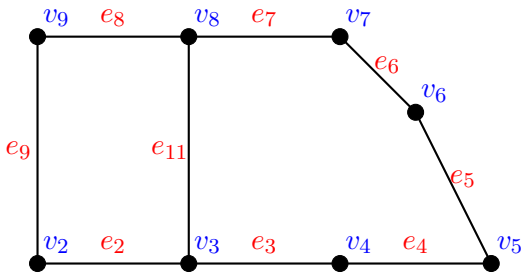


Figure 3.5: Cycle Subgraph of G Examples

Alternatively if we take the subset of edges $E_6 = \{e_1, e_2, e_3, e_4, e_7, e_8, e_9, e_{10}\}$ and its corresponding vertex set $V_6 = \{v_1, v_2, v_3, v_4, v_5, v_7, v_8, v_9, v_{10}\}$ from Chapter 3 form a tree graph.

Definition 3.5 ([GM12, Def. 4.16])**.** A *vertex-edge incidence matrix* is a binary matrix

with rows that represent the vertices of a graph, columns that represent the edges, and have either a 1 placed where an edge a vertex interact or a 0 where an edge and vertex do not interact.

A matrix is considered in its *row-reduced echelon form* if

- The first, leading, input of a row is 1, and the leading 1 component of the next row below is to the right of the leading entry above it.
- Each column containing a leading 1, contains zeros for all other entries of the column, and all zero rows are at the bottom of the matrix.

After row operations, a matrix in its row reduced form can contain multiple consecutive ones or zeros in a row or column. Let a bold one, $\mathbf{1}$, in a matrix column, represents an arbitrary number of consecutive ones in the column, and a bold zero, $\mathbf{0}$, represents an arbitrary number of consecutive zeroes from that bold zero in the column and below, and a bold zero with an overline, $\overline{\mathbf{0}}$, in a matrix represents a row of arbitrary length composed of only zeros.

Chapter 4

Codes Based On Incidence Matrices

4.1 Tree Graphs: Over $\mathbb{F}_2$

In this Chapter we will explore using graphs to obtain their incidence matrices. From these matrices we will create generator matrices and construct codes, now let's begin with tree graphs.

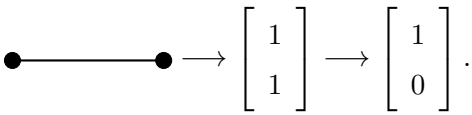
Definition 4.1 ([GM12, P. 20])**.** A *submatrix* of a matrix is another matrix obtained by forming the intersection of certain rows and columns, or equivalently by deleting certain rows and columns.

A tree graph can be drawn and labeled many different ways. Tree graphs contain special cases within them, and two that we will be looking at are spoke graphs and path graphs.

Paths

A path graph has been defined previously but a path cannot contain any cycles, nor can it contain any loops. Such a graph can be represented as a line of consecutive connected vertices and edges. Consider the path graph P_2 and its incidence matrix in Section 4.1

The path graph P_2 contains a single edge, and ordered pair of vertices that make up that edge. By simplifying the matrix of P_2, we can obtain its row reduced echelon



$$\longrightarrow \begin{bmatrix} 1 \\ 1 \end{bmatrix} \longrightarrow \begin{bmatrix} 1 \\ 0 \end{bmatrix}.$$

Figure 4.1: P_2 Incidence Matrix And Row Reduced Echelon Form

form and use this reduced form as a generator matrix to transmit codes. Unfortunately, the generator matrix produced from the graph P_2 can only transmit a single component as a message with no redundancy portion or parity-check equations.

Example 4.2. Let P_n such that $n = 2$ represent a path graph with two vertices.

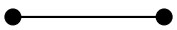


Figure 4.2: The Graph of P_2

$$\begin{bmatrix} 1 \\ 1 \end{bmatrix} \longrightarrow \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

The incidence matrix constructed from P_2 results in the first matrix on the left. In this case adding the first row to the second row is the only step needed to row reduce the matrix. Now the incidence matrix of P_2 is in its row reduced echelon form and is of the form

$$\begin{bmatrix} 1 \\ 0 \end{bmatrix}.$$

Theorem 4.3. *The row reduced echelon form of the incidence matrix of an arbitrary path graph P_n is*

$$\left[\begin{array}{c} I_{n-1} \\ \hline \overline{\mathbf{0}} \end{array}\right].$$

Proof. We induct on the number of vertices, n.

Base Case: Let $n = 2$, the path graph P_2 has incidence the matrix

$$\begin{bmatrix} 1 \\ 1 \end{bmatrix}.$$

After performing row operations, its Row-Reduced Echelon Form is

$$\begin{bmatrix} 1 \\ 0 \end{bmatrix}.$$

Induction Hypothesis: Assume $n = k$ such that the row reduced echelon form of the incidence matrix of P_k is

$$\left[\begin{array}{c} I_{k-1} \\ \hline \overline{\mathbf{0}} \end{array}\right].$$

Induction Step: We want to show that the row reduced echelon form of the incidence matrix of P_{k+1} is

$$\left[\begin{array}{c} I_k \\ \hline \overline{\mathbf{0}} \end{array}\right].$$

The incidence matrix of P_{k+1} is

$$\left[\begin{array}{c:c} 1 & 0 \\ \hdashline 1 & 1 \\ \hdashline 0 & 1 \end{array}\right] \longrightarrow \left[\begin{array}{c:c} 1 & 0 \\ \hdashline 0 & 1 \\ \hdashline 0 & 1 \end{array}\right].$$

The submatrix in the top left of the incidence matrix for P_{k+1} of size $k \times k-1$, separated by the dashed lines, is the incidence matrix of P_k. By adding the first row to the second row we see the first column is now in its reduced form. Now if we delete the first column and first row, we obtain a submatrix in P_{k+1} that is the incidence matrix of P_k. By the induction hypothesis, we know that the incidence matrix of P_k is

$$\left[\begin{array}{c} I_{k-1} \\ \hline \overline{\mathbf{0}} \end{array}\right],$$

and thus the incidence matrix of P_{k+1} is equivalent to

$$\left[\begin{array}{c|c} 1 & 0 \\ \hline 0 & I_{k-1} \\ 0 & \overline{\mathbf{0}} \end{array}\right].$$

Including the previously deleted row and column, we can see that the matrix is already in its row reduced echelon form and in the expected pattern

$$\begin{bmatrix} I_k \\ \overline{\mathbf{0}} \end{bmatrix}.$$

□

Question 4.4. Can a path graph produce a generator matrix with a minimum difference of three or more?

From Theorem 4.3, we can see that the incidence matrices produced from path graphs will always reduce to be identity matrices. Thus we can conclude that regardless of the size, a path graph cannot produce a generator matrix with a minimum difference of three or more.

Spoke

A *spoke graph* is a specific type of tree graph in which we will have a fixed vertex and every other vertex in the graph will be joined to the fixed vertex and no other vertex by an edge. Let vertex A be fixed in space, and let n number of vertices join A with n number of edges. All vertices are joined to A, and only A, in the shape of a spoked wheel, thus such a graph is called a *Spoke Graph*.

The incidence matrix of graph S_4 is of the form

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & 1 \end{bmatrix} \longrightarrow \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}.$$

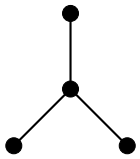


Figure 4.3: The Graph of S_4

Different from the path graph incidence matrix, the spoke graph incidence matrix is nearly in its identity form already. We must simply add all row before the last row[1], to the last row. The incidence matrix of S_4 is equivalent to a (4×3) identity matrix that contains a trivial zero row. Thus by row operations, a spoke graph will result in an incidence matrix who's row reduced echelon form is equivalent to an $(n-1)$ identity matrix which only transmits the message word as the codeword[2]

$$\begin{bmatrix} I_{n-1} \\ \overline{\mathbf{0}} \end{bmatrix}.$$

Due to the resulting matrix being the identity matrix, this generator matrix simply transmits the message word as the codeword, and does not encode or add any redundancy portion to the transmitted codewords. Thus, the incidence matrices of path and spoke graphs, will reduce to identity matrices and cannot be used to create generator matrices and (n, k) codes that can both detect and correct errors caused during transmission.

4.1.1 Trees

Theorem 4.5. *The row reduced echelon form of the incidence matrix of a trees graphs T_n is*

$$\left[\begin{array}{c} I_{n-1} \\ \hline \overline{\mathbf{0}} \end{array}\right].$$

[1]The last row being the row that represents the only vertex that does not have a degree of one in the graph.

[2]This is verified by the Section 4.1.1.

Proof. We induct on the number of vertices, n.

Base Case: Consider an arbitrary tree graph with three vertices T_3, and its incidence matrix can be of the form

$$\begin{bmatrix} 1 & 0 \\ 1 & 1 \\ 0 & 1 \end{bmatrix}.$$

Its row reduced echelon form is

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix}.$$

Induction Hypothesis: Let $n = k$, such that the incidence matrix of a tree graph T_k has a row reduced echelon form equivalent to

$$\begin{bmatrix} I_{k-1} \\ \overline{\mathbf{0}} \end{bmatrix}.$$

Induction Step: Want to show that the incidence matrix for T_{k+1} has a row reduced echelon form equivalent to

$$\begin{bmatrix} I_k \\ \overline{\mathbf{0}} \end{bmatrix}.$$

Let T_{k+1} have an incidence matrix of the form[3]

$$\left[\begin{array}{c|cc} 1 & 0 & 0 \\ \hline 1 & 1 & * \\ \mathbf{0} & 1 & * \end{array}\right] \longrightarrow \left[\begin{array}{c|cc} 1 & 0 & 0 \\ \hline 0 & 1 & * \\ \mathbf{0} & 1 & * \end{array}\right].$$

By adding he first row to the second row, the first column/row is now in its identity form[4]. Now if we add the first row and column, the submatrix produced is an incidence

[3]This is possible because we already understand the matrix of T_k, and the matrix of T_{k+1} is simply the matrix of T_k with a single additional row and column to represent the additional vector, and the additional edge necessary to connect the k^{th} with the $k + 1^{th}$ vector.

[4]We know the first row will contain only a single nonzero component since it is the initial point of the tree graph and the initial vertex of tree graphs in this paper will have a degree of 1.

matrix for a tree graph with k vertices, which we know has a row reduced echelon form in our desired form from the induction hypothesis. Thus the incidence matrix of T_{k+1} is now in the form

$$\left[\begin{array}{c:c} 1 & \overline{\mathbf{0}} \\ \hdashline \mathbf{0} & \mathbf{I_{k-1}} \\ & \overline{\mathbf{0}} \end{array}\right] \longrightarrow \left[\begin{array}{c} \mathbf{I_k} \\ \overline{\mathbf{0}} \end{array}\right].$$

Thus the row reduced echelon form of T_{k+1} is equivalent to our expected matrix pattern

$$\left[\begin{array}{c} I_k \\ \overline{\mathbf{0}} \end{array}\right].$$

□

This generator matrix can send 2^k message words, of length k. The message words are identical to the codewords since there is no redundancy portion to the generator matrix. Thus the row reduced echelon form of a single tree graphs incidence matrix cannot be turned into a generator matrix since it does not contain any redundancy portion after row reducing.

The row reduced echelon form of the incidence matrix is nearly in the form of a code generator matrix with identity columns and redundancy columns. In the case of tree graphs, we can see in Section 4.1.1 that their row reduced echelon form is an identity matrix with an additional zero row. If we delete the last row, the zero row, we obtain a square size k identity matrix that can be used as a generator matrix that transmits length k message words as length k codewords with no redundancy portion encoded to it. A code without any redundancy portion is not useful in most modern applications since it does not encode the message word. This type of generator matrix will not be capable of detecting errors, nor correcting them.

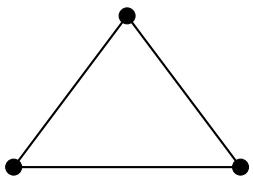
4.2 Cycle Graphs

Theorem 4.6. *The row reduced echelon form of a cycle graph with n number of vertices, C_n, is*[5]

$$\left[\begin{array}{c|c} I_{n-1} & \mathbf{1} \\ 0 & 0 \end{array}\right].$$

Proof. Inducting on the number of vertices, n.

Base Case: Let $n = 3$. The incidence matrix of the cycle graph with three vertices C_3 is



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

Figure 4.4: The Graph of C_3

The row reduced echelon form of the incidence matrix of C_3 is

$$\begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 0 \end{bmatrix}.$$

Induction Hypothesis: Assume $n = k$ such that the row reduced echelon form of the incidence matrix of C_k is

$$\begin{bmatrix} I_{k-1} & \mathbf{1} \\ 0 & 0 \end{bmatrix}.$$

Induction Step: We want to show that the row reduced echelon form of the incidence

[5]The bold one, **1**, represents a parity-check column. In cycle graphs these columns are $k-1$ nonzero components long with the final components being zero.

matrix of C_{k+1} is

$$\begin{bmatrix} I_k & \mathbf{1} \\ \overline{\mathbf{0}} & 0 \end{bmatrix}.$$

The incidence matrix of C_{k+1} is of the form

$$\begin{bmatrix} 1 & 0 & 0 & \dots & 0 & 0 & 1 \\ 1 & 1 & 0 & \dots & 0 & 0 & 0 \\ 0 & 1 & 1 & \dots & 0 & 0 & 0 \\ 0 & 0 & 1 & \dots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & 1 & 1 & 0 \\ 0 & 0 & 0 & \dots & 0 & 1 & 1 \end{bmatrix}.$$

By adding the first row to the second row, the first column is in its identity form, and we obtain a matrix of the form

$$\left[\begin{array}{c|cccccc} 1 & 0 & 0 & \dots & 0 & 0 & 1 \\ \hline 0 & 1 & 0 & \dots & 0 & 0 & 1 \\ 0 & 1 & 1 & \dots & 0 & 0 & 0 \\ 0 & 0 & 1 & \dots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & 1 & 1 & 0 \\ 0 & 0 & 0 & \dots & 0 & 1 & 1 \end{array}\right].$$

By deleting the first column and deleting the first row of the matrix, we obtain a submatrix that is equivalent to the incidence matrix of C_k given in the induction step. By the induction hypothesis, the row reduced echelon form of C_k is

$$\left[\begin{array}{c|c} I_{k-1} & \mathbf{1} \\ \overline{\mathbf{0}} & 0 \end{array}\right]$$

and the incidence matrix of C_{k+1} is equivalent to

$$\left[\begin{array}{c|c|c} 1 & 0 & 1 \\ \hline 0 & I_{k-1} & \mathbf{1} \\ 0 & \overline{\mathbf{0}} & 0 \end{array}\right].$$

This matrix is now in its row reduced echelon form and is equivalent to

$$\begin{bmatrix} I_k & \mathbf{1} \\ \overline{\mathbf{0}} & 0 \end{bmatrix}.$$

□

Example 4.7. Consider the matrix, with columns labeled c_1, c_2, and c_3 from left to right, produced by the graph C_3

$$\begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 0 \end{bmatrix}.$$

By deleting the zero row, we obtain the generator matrix

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

The code produced by G begins with the closed set of message words $M = \{(00), (01), (10), (11)\}$, and are encoded into the closed set of codewords $C = \{(000), (011), (101), (110)\}$. The parity-check equation of this generator matrix is $x_3 = x_1 + x_2$, and all components of the message words are used in the parity check equations.

During transmissions where a single error occurs, it is not possible for us to determine whether the error occurred in a component of the message or if the error occurred in the parity check component using nearest-neighbor decoding since two the received word has two codewords that are equally likely to be the correct codeword. This code can still find use in small applications where the users require a very simple set of messages they need to transmit, and more than one error occurring during transmission

$$
\begin{aligned}
Single-error & \\
(000) &= (100), (010), (001) \\
(101) &= (001), (111), (100) \\
(011) &= (111), (001), (010) \\
(110) &= (010), (100), (101) \\
two-errors & \\
(000) &= (110), (011), (101) \\
(101) &= (011), (110), (000) \\
(011) &= (101), (000), (110) \\
(110) &= (000), (101), (011).
\end{aligned}
$$

Figure 4.5: Single And Double Error Transmissions

is highly unlikely. Anytime a single error occurs during transmission, we can see in Figure 4.5 that the received word will not match any of the codewords in our set C, and we can be certain that an error occurred.

In the case of two errors occurring on two unique components, the word received from transmission would be a codeword, but it would be incorrect and different from the originally sent codeword since the codewords only contain two message components and the code has a minimum distance of 2. Thus if two error occur during transmission, the entire message potion of the word can change, and since a two components changed in this binary code, the parity check components of the codeword remains unchanged which negates its purpose.

We can see that the row reduced echelon form of a cycle graph produces a generator matrix that contains a redundancy portion. Although the single parity check column is not enough to correct errors, it is an improvement for error detection.

Chapter 5

Joining

When joining graphs together to form a single connected graph, the new graph also constructs a vertex-edge incidence matrix. There are two ways we will join two separate connected graphs as a single connected graph, and that is "parallel connection" which use an edge and its end vertices, and "direct sum" which uses only vertices to join two graphs. Both of these methods do not join by removing edges and vertices, or by adding edges or vertices, but by merging vertices, or an edge from each graph as a single edge or vertex.

Definition 5.1 ([Oxl11, P. 251])**.** Let G_1 and G_2 be two graphs, and let e_i and c_i be be edges in G_1 and G_2 respectively. Let the edge e_i have end vertices v_1 and v_2, and the edge c_i have end vertices p_1 and p_2. The *parallel connection* of edges e_i and c_i results in a new graph G with the following properties:

- The vertices v_1 and p_1 are deleted and replaced by a single vertex a.
- The vertices v_2 and p_2 are deleted and replaced by a single vertex b.
- Both edges e_i and c_i, that previously connect vertices v_1 and v_2, and p_1 and p_2, are deleted and are replaced by a single edge d, with end vertices a and b.[1]

Definition 5.2. Let G_1 and G_2 be two arbitrary graphs, and let $v_1 \in G_1$ and $v_2 \in G_2$ be vertices in these graphs. The *direct sum* of vertices v_1 and v_2 results in a new graph G with the following properties:

[1]Both graphs G_1 and G_2 are subgraphs of the graph G, share the common edge d with end vertices a and b.

- The vertex set V is the union of the vertex set of G_1, V_1, and the vertex set of G_2, V_2, and v_1 and v_2 are identified as a single vertex v:

$$V = (V_1 \cup V_2) \setminus \{v_1, v_2\} \cup v.$$

- The edge set E is the union of the edges from both graphs, $E = E_1 \cup E_2$, where any edge incident to v_1 or v_2 is now incident to the new vertex v.

Example 5.3. Consider the cycle graph C_3 and its incidence matrix as seen in Section 4.2. Let graph A be two C_3 cycle graphs joined by "direct sum", and graph B be two C_3 graphs joined by "parallel connection". The two C_3 graphs are joined as a single graph in graph A and graph B, and graph C_3 is contained as a subgraph twice in graphs A and B each.

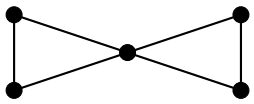


Figure 5.1: Graph A

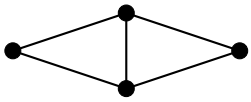


Figure 5.2: Graph B

The incidence matrix of graph A is

$$M(A) = \left[\begin{array}{ccc:ccc} 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ \hdashline 0 & 1 & 1 & 1 & 1 & 0 \\ \hdashline 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 \end{array}\right].$$

In $M(A)$, we focus on two sub-matrices. One in the upper left 3×3 submatrix of the incidence matrix, and one 3×3 submatrix in the bottom right of $M(A)$ are the incidence matrix of C_3 from Section 4.2 respectively[2].

[2]The sub-matrices are separated in the incidence matrices $M(A)$ and $M(B)$ by dashed lines.

The incidence matrix of graph B is

$$M(B) = \left[\begin{array}{ccccc} 1 & 1 & 0 & 0 & 0 \\ \hdashline 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \\ \hdashline 0 & 0 & 0 & 1 & 1 \end{array}\right].$$

In $M(B)$, we see that there also exists sub-matrices. Similar to the $M(A)$, there is one in the upper left 3×3 submatrix of the incidence matrix, and one 3×3 submatrix in the bottom right of $M(B)$ equivalent to the incidence matrix of C_3.

The incidence matrix $M(A)$, contains the incidence matrix of cycle graph C_3 as sub-matrices as a result of it being contained as a subgraph of graph A. Similarly, the incidence matrices of C_3 are seen in $M(B)$ as sub-matrices. The graph A is joined by sharing a common vertex, and the third row(vertex) of the incidence matrix is shared between the two sub-matrices of the incidence matrix of C_3. Since the graph B is joined by sharing an edge between the two cycle graphs, and every edge contains two end vertices, the incidence matrices of C_3 in $M(B)$ share the third column(edge), as well as the second and third row.

5.1 Parallel Connection

The vertex-edge incidence matrix of the parallel connection of two graphs exists, and can be expressed as a block matrix containing the incidence matrices of the individual graphs. In order to join two graphs using a common edge, *parallel connection*, or by a common vertex, *direct sum*, we require the final column of the graphs incidence matrices to have the same form as the first column of the incidence matrix, which is two consecutive nonzero rows. This allows the matrix to be merged using its final column as the initial column of the joining graphs incidence matrix through parallel connection, as well as allowing the final row of the incidence matrix to end with a 1 and the first row to begin with a 1 which allows incidence matrices to be joined by direct sum. In order to accomplish this we must rearrange the columns of the matrix, and this can be done by simply relabeling the edges of the graphs which relabels the corresponding columns of the matrix.

Remark 2. In order for two cycle graphs to join and allow for additional graphs to join

by parallel connection, we make the first column of their incidence matrix will begin with two nonzero rows, and the final column will end with two nonzero rows.

Proof. In order for two cycle graphs to join by a common edge, they must join not only the edge, but its two vertices that form the its ordered pair as well. For this, there must exist a column with two adjacent nonzero rows in the incidence matrix. In order to join two incidence matrices, the columns needed must be the first and last column of the incidence matrix in order to keep the individual matrices used as sub matrices of the joined graphs incidence matrix. By using the incidence matrix given by Theorem 4.6, we know that a incidence matrix for a graph with n vertices, is of the form

$$\begin{bmatrix} 1 & 0 & 0 & \dots & 0 & 0 & 1 \\ 1 & 1 & 0 & \dots & 0 & 0 & 0 \\ 0 & 1 & 1 & \dots & 0 & 0 & 0 \\ 0 & 0 & 1 & \dots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & 1 & 1 & 0 \\ 0 & 0 & 0 & \dots & 0 & 1 & 1 \end{bmatrix}.$$

In order to join graphs by parallel connection, the final column of the graphs incidence matrices must be different from its current form. The column needed as the final n^{th} column is the original second to last column whose final two rows are nonzero. This is achieved by moving the final column into the second columns position and shifting over all columns to its right to the right by one position. [3] Now the incidence matrix that can be joined to other incidence matrices is of the form

$$\begin{bmatrix} 1 & 1 & 0 & \dots & 0 & 0 & 0 \\ 1 & 0 & 1 & \dots & 0 & 0 & 0 \\ 0 & 0 & 1 & \dots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & 1 & 1 & 0 \\ 0 & 0 & 0 & \dots & 0 & 1 & 1 \\ 0 & 1 & 0 & \dots & 0 & 0 & 1 \end{bmatrix}.$$

[3] the original second columns becomes the third, the third became the forth, and so on and so forth until the original second-to-last column becomes the final n^{th} column.

Now the incidence matrices of any cycle graphs that are going to be joined through parallel connection must have a first column with the two rows being non zero and a final column where its last two rows are nonzero. The incidence matrix of the graphs being joined will be in the same form as the matrix found in Theorem 4.6. □

Example 5.4. The graph in Example 3.1, would result in the incidence matrix $M(C_3)$ below

$$M(C_3) = \begin{bmatrix} 1 & 0 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \end{bmatrix}.$$

Figure 5.3: C_3 Incidence Matrix

Except for the column connecting the first row to the last row, each column contain two consecutive rows that are nonzero. Let the columns, from left to right, be labeled c_1, c_2, c_3 respectively and c_2 be the parity-check column of the generator matrix as seen in the in Example 4.7. By rearranging the columns such that the column without consecutive ones becomes the second column, and the columns ending in two consecutive $1s$ becomes the last column in the matrix

$$M(C_3) = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix}.$$

Figure 5.4: C_3 Joining Matrix

Remark 3. When using the SageMath code in Chapter 7 to join two cycle graphs, there is a restriction such that the initial graph C_n, must be the same size or greater than C_m, the second graph that will be joined to C_n. In other words, If $n \geq m$, then the sagemath code will work. The cause for this restriction in sagemath is still unknown to me.

Proposition 5.5. The incidence matrix of the parallel connection of two cycle graphs C_n and C_m, with n and m number of vertices, respectively, is the size $(n+m-2)\times(n+m-1)$

is of the form[4]

$$\left[\begin{array}{ccc:c:ccc:ccc} 1 & 1 & \ddots & 0 & 0 & \ddots & 0 & 0 & 0 \\ \hdashline 1 & 0 & \ddots & 0 & 0 & \ddots & 0 & 0 & 0 \\ 0 & 0 & \ddots & 1 & 1 & \ddots & 0 & 0 & 0 \\ 0 & 1 & \ddots & 1 & 0 & \ddots & 1 & 0 & 0 \\ \hdashline \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \\ 0 & 0 & \ddots & 0 & 0 & \ddots & 0 & 1 & 1 \\ 0 & 0 & \ddots & 0 & 1 & \ddots & 0 & 0 & 1 \end{array}\right].$$

and its row reduced echelon form is

$$\left[\begin{array}{ccc:c:ccc:c} 1 & 0 & \ddots & \mathbf{1} & 0 & \ddots & 0 & 0 & 1 \\ 0 & 1 & \ddots & 1 & 0 & \ddots & 0 & 0 & 1 \\ 0 & 0 & \ddots & 1 & 0 & \ddots & 0 & 0 & 1 \\ & \overline{\mathbf{0}} & \ddots & 0 & 1 & \ddots & 0 & 0 & 1 \\ \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & 1 \\ 0 & 0 & \ddots & 0 & 0 & \ddots & 0 & 1 & 1 \\ \overline{\mathbf{0}} & & \dots & \overline{\mathbf{0}} & & \dots & \overline{\mathbf{0}} & & 0 \end{array}\right]$$

containing two non-identity columns, and $n + m - 3$ identity columns.

Proof. Base Case: We induct on the number of vertices n. Let $n = 3$, and C_m be an arbitrary cycle graph with m-number of vertices. The incidence matrix of the parallel connection of C_3 and C_m is

$$\left[\begin{array}{cc:c:ccc:ccc} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hdashline 1 & 0 & 1 & 1 & \ddots & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & \ddots & 1 & 0 & 0 \\ \hdashline \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \\ 0 & 0 & 0 & 0 & \ddots & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & \ddots & 0 & 0 & 1 \end{array}\right].$$

[4]The bottom horizontal and right vertical dashed lines form the boundary of the incidence matrix of C_n in the top left of the matrix, and the left vertical and top horizontal dashed line form the boundary of the incidence matrix for C_m in the bottom right of the matrix.

We know the submatrix in the bottom right of this matrix beginning on the third column from the left, and the second row from the top has size $m \times m$. Thus the size of the parallel connection must have size $(m+1) \times (m+2)$. This is the claimed size when $n = 3$, $(3 + m - 2) \times (3 + m - 1)$. By first adding the first two rows to the final row of the submatrix of C_3, we obtain the zero row for final row of the submatrix. By adding all rows of the submatrix of graph C_m including the first row itself[5], to the first row of C_m, we obtain zeroes for the first row of the submatrix of C_m. By continuing the reduction pattern[6], $R_2 + R_3 + \ldots R_m \longrightarrow R_2^*$, $\ldots R_{m-1} + R_m \longrightarrow R_{m-1}^*, R_m + R_m \longrightarrow R_m^*$, we obtain the zero row in the final row of the parallel connection incidence matrix's row reduced echelon form, G_1, of the form

$$G_1 = \left[\begin{array}{cc:c:ccccc:c} 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & \ddots & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & \ddots & 0 & 1 \\ \hdashline \vdots & \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & 1 \\ 0 & 0 & 0 & 0 & \ddots & 0 & 1 & 1 \\ 0 & 0 & \overline{\mathbf{0}} & 0 & \ddots & 0 & \overline{\mathbf{0}} & 0 \end{array}\right].$$

Induction Hypothesis: Let $n = k$ such that the matrix produced by the parallel connection

[5] to counter the addition on the first row and second row that occurred when adding all rows in the submatrix of C_3

[6] R^* refers to the new row being formed after the operation, and by adding the all rows of the submatrix C_m to the first row of C_m, we obtain zeros for all components of the row that was shared with the submatrix C_3. This allows us to then complete additional row operations in the submatrix of C_m without affecting components of the submatrix of C_3 since adding by zero is trivial and does not change the components of the matrix.

of C_k and C_m has a size of $(k+m-2)\times(k+m-1)$, and is of the form

$$\left[\begin{array}{ccc:c:ccc:ccc} 1 & 1 & \ddots & 0 & 0 & \ddots & 0 & 0 & 0 \\ 1 & 0 & \ddots & 0 & 0 & \ddots & 0 & 0 & 0 \\ \dots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ \hdashline 0 & 0 & \ddots & 1 & 1 & \ddots & 0 & 0 & 0 \\ 0 & 1 & \ddots & 1 & 0 & \ddots & 1 & 0 & 0 \\ \hdashline \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \\ 0 & 0 & \ddots & 0 & 0 & \ddots & 0 & 1 & 1 \\ 0 & 0 & \ddots & 0 & 1 & \ddots & 0 & 0 & 1 \end{array}\right].$$

With a row reduced echelon form of

$$\left[\begin{array}{ccc:c:ccc:ccc} 1 & 0 & \ddots & 1 & 0 & \ddots & 0 & 0 & \mathbf{1} \\ 0 & 1 & \ddots & \mathbf{1} & 0 & \ddots & 0 & 0 & \mathbf{1} \\ \dots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ \hdashline 0 & 0 & \ddots & 1 & 0 & \ddots & 0 & 0 & \mathbf{1} \\ 0 & 0 & \ddots & 0 & 1 & \ddots & 0 & 0 & \vdots \\ \hdashline \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots & \vdots \\ 0 & 0 & \ddots & 0 & 0 & \ddots & 0 & 1 & 1 \\ 0 & 0 & \ddots & 0 & 0 & \ddots & 0 & 0 & 0 \end{array}\right].$$

Induction Step: Want to show the parallel connection of C_{k+1} produces a matrix of size $(k+m-1)\times(k+m)$ and with a row reduced echelon form containing two non identity columns and $k+m-3$ identity columns. Let $n=k+1$, and C_m be cycle graphs. The incidence matrices of C_{k+1} and C_m are

$$\left[\begin{array}{cccccc} 1 & 1 & 0 & \dots & 0 & 0 \\ 1 & 0 & 1 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & 1 & 1 \\ 0 & 1 & 0 & \dots & 0 & 1 \end{array}\right] \text{ and } \left[\begin{array}{ccccccc} 1 & 1 & 0 & \dots & 0 & 0 & 0 \\ 1 & 0 & 1 & \dots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & 0 & 1 & 1 \\ 0 & 1 & 0 & \dots & 0 & 0 & 1 \end{array}\right] \text{ respectively.}$$

The parallel connection of C_{k+1} and C_m results in the incidence matrix

$$\left[\begin{array}{ccc:c:ccc:ccc} 1 & 1 & \ddots & 0 & 0 & \ddots & 0 & 0 & 0 \\ 1 & 0 & \ddots & 0 & 0 & \ddots & 0 & 0 & 0 \\ \ldots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ \hdashline 0 & 0 & \ddots & 1 & 1 & \ddots & 0 & 0 & 0 \\ 0 & 1 & \ddots & 1 & 0 & \ddots & 1 & 0 & 0 \\ \hdashline \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \\ 0 & 0 & \ddots & 0 & 0 & \ddots & 0 & 1 & 1 \\ 0 & 0 & \ddots & 0 & 1 & \ddots & 0 & 0 & 1 \end{array}\right].$$

After performing the row reduction pattern in the base case, the result is the submatrix of size $k-1$ in the top left of the parallel connections incidence matrix, formed by the deleting the left most column and first top row. It is the same size and is equivalent to the submatrix in the induction step of size k with $k-1$ identity columns, one non-identity column, and a zero row in the submatrix.

$$\left[\begin{array}{c|c:c:ccc:ccc} 1 & & 1 & 0 & \ldots & 0 & 0 & 0 \\ \hline 0 & \mathbf{I_{k-1}} & 1 & 0 & \ldots & 0 & 0 & 0 \\ \vdots & & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & & 1 & 0 & \ldots & 0 & 0 & 0 \\ 0 & \overline{\mathbf{0}} & 0 & 1 & \ldots & 0 & 0 & 0 \\ \hdashline \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \ldots & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & \ldots & 0 & 0 & 1 \end{array}\right].$$

Now if we looks at the submatrix of the cycle graph formed by C_m, in the bottom right of the matrix, beginning at the $k+1$ row and column. Following the same row operation pattern as the first submatrix with the $k+1$ row being considered first row R_1, the $k+2$ row is R_2 and so on and so forth. Adding the deleted column and row of C_{k+1} back into the submatrix, the parallel connections incidence matrix is of the form

$$\left[\begin{array}{c:c|c:c} \mathbf{I_k} & \mathbf{1} & \mathbf{0} & 1 \\ \overline{\mathbf{0}} & 0 & \mathbf{I_{m-2}} & \mathbf{1} \\ \mathbf{0} & \mathbf{0} & \overline{\mathbf{0}} & 0 \end{array}\right].$$

Thus, the final matrix size is $(k+(m-2)+1)\times(k+1+(m-1)+1)$ which is equivalent to $(n+m-2)\times(n+m-1)$, and our row reduced echelon form contains $n+m-3$ identity columns, two non-identity columns, and a trivial row composed of zeros. □

Theorem 5.6. *The parallel connection of any two tree graphs T_a and T_b, with a and b number of vertices each, respectively, has a matrix of size $(a+b-2)\times(a+b-3)$, and by deleting the zero row from its row reduced echelon form, the row reduced echelon form is a square identity matrix of size $a+b-3$.*

Proof. Base Case: We induct on the number of vertices a.[7] Let T_3 be a connected graph with $a=3$ vertices, and T_b be an arbitrary connected graph with b number of vertices. The matrix produces by the parallel connection of T_3 and T_b is

$$\left[\begin{array}{c:c:ccccc} 1 & 0 & 0 & 0 & \ddots & 0 & 0 \\ \hdashline 1 & 1 & 0 & 0 & \ddots & 0 & 0 \\ 0 & 1 & 1 & 0 & \ddots & 0 & 0 \\ \hdashline 0 & 0 & 1 & 1 & \ddots & 0 & 0 \\ \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots \\ 0 & 0 & 0 & 0 & \ddots & 1 & 1 \\ 0 & 0 & 0 & 0 & \ddots & 0 & 1 \end{array}\right] \longrightarrow \left[\begin{array}{cc:c:ccccc} 1 & 0 & 0 & 0 & \ddots & 0 & 0 \\ 0 & 1 & 0 & 0 & \ddots & 0 & 0 \\ \hdashline 0 & 0 & 1 & 0 & \ddots & 0 & 0 \\ \hdashline 0 & 0 & 0 & 1 & \ddots & 0 & 0 \\ \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots \\ 0 & 0 & 0 & 0 & \ddots & 1 & 0 \\ 0 & 0 & 0 & 0 & \ddots & 0 & 0 \end{array}\right].$$

The submatrix in the bottom right which forms the incidence matrix of T_b has size $(b)\times(b-1)$. Thus the size of the whole matrix is $b+1\times b$ which is equivalent to the expected size from the equation in our statement, $(3+b-2)\times(3+b-3)$.

Induction Hypothesis: Let $a=k$, such that the incidence matrix of T_k is an identity matrix with size $k\times(k-1)$.

Induction Step: Let $a=k+1$, and T_b be an arbitrary connected graph with b vertices. The incidence matrix of the parallel connection of T_{k+1} and T_b is of the form

[7] Each vertex added to the initial connected graph increases the number of edges by one as well. Since no loops are allowed, each additional vertex adds an additional edge and vice versa. Thus we cannot simply increase the number of vertices without increasing the number of edges as well.

$$\left[\begin{array}{cccc:c:ccc:cc} 1 & 0 & \ddots & 0 & 0 & 0 & \ddots & 0 & 0 \\ 1 & 1 & \ddots & 0 & 0 & 0 & \ddots & 0 & 0 \\ \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \\ \hdashline 0 & 0 & \ddots & 1 & 1 & 0 & \ddots & 0 & 0 \\ 0 & 0 & \ddots & 0 & 1 & 1 & \ddots & 0 & 0 \\ \hdashline \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \\ 0 & 0 & \ddots & 0 & 0 & 0 & \ddots & 1 & 1 \\ 0 & 0 & \ddots & 0 & 0 & 0 & \ddots & 0 & 1 \end{array}\right] \longrightarrow \left[\begin{array}{c:c:ccc} \mathbf{I_{k-1}} & \mathbf{0} & \dots & 0 & 0 \\ \hdashline 0 & 1 & \vdots & 0 & 0 \\ \overline{\mathbf{0}} & 1 & 1 & 0 & 0 \\ \hdashline \vdots & \vdots & \vdots & \vdots & \vdots \\ \mathbf{0} & \mathbf{0} & \ddots & 1 & 0 \\ \vdots & \vdots & \ddots & 1 & 1 \\ \mathbf{0} & \mathbf{0} & \ddots & 0 & 1 \end{array}\right].$$

We can see how the submatrix of T_k in the top left of incidence matrix reduced to an identity matrix with size $k-1$, without including the shared column. The submatrix in the bottom right of the parallel connection matrix, including the shared column, represents the matrix of T_b and by Section 4.1.1 it must simplify to an identity matrix as well. Thus we obtain the matrix of the form

$$\left[\begin{array}{c:c} \mathbf{I_{k-1}} & \overline{\mathbf{0}} \\ \hdashline \mathbf{0} & \mathbf{I_{b-2}} \\ \hline \overline{\mathbf{0}} & \overline{\mathbf{0}} \end{array}\right] = \left[\begin{array}{c} \mathbf{I_{k+b-3}} \\ \hline \overline{\mathbf{0}} \end{array}\right].$$

The size of our matrix is $(k+1+b-2) \times (k+1+b-3) = (k+b-1) \times (k+b-2)$ which is the expected size following the equation from the initial statement. Thus the parallel connection of any two tree graphs T_a and T_b, has an incidence matrix of size $(a+b-2) \times (a+b-3)$ composed of a square size $k+b-3$ identity submatrix and a zero row.[8] □

Example 5.7. Consider the two cycle graphs C_3 and C_5. By joining them using a parallel connection it will form a house graph. The incidence matrix of the house graph can be obtained from the SageMath code in Chapter 7, or from following the steps in Definition 5.1.

[8]This includes spoke graphs and path graphs since although they are special cases, they are still tree graph

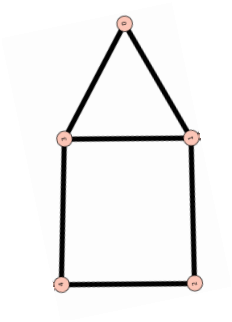


Figure 5.5: House Graph

The resulting incidence matrix is

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

Remark 4. By Definition 3.2, a cycle graph C_n contains n vertices and edges, and any two adjacent vertices are connected by a single edge. Since all columns in a cycle graphs incidence matrix contain two nonzero row elements, and row addition is done under $\mathbb{F}_2$, the sum of all rows in a arbitrary cycle graphs incidence matrix is 2 which is 0. The parallel connection of two cycle graphs does not introduce any additional edges or vertices, does not introduce parallel edges, and all edges are adjacent to two vertices. Thus similar to the rows of the incidence matrix of simple cycle graphs adding up to a zero row, the sum of all rows in the incidence matrix of the parallel connection of cycle graphs results in a zero row.

Example 5.8. Here we will go over steps to obtain the row reduced echelon form of the incidence matrix of the parallel connection of two cycle graphs without using sagemath, and then turn it into a generator matrix.

Now we can look at this incidence matrix, A, as a block matrix. We have the 4×4 matrix on the top left, and a 3×3 matrix on the bottom right. They over lap and share the fourth column from the left, on the third and fourth row. First step is to add all rows to the final sixth row, $R_1 + R_2 + R_3 + R_4 + R_5 \longrightarrow R_5$. This makes the last row a

zero row due to all columns containing two non zero inputs and our row addition is done under $\mathbb{F}_2$ and the sum of all rows will equal zero in each column,

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

The Second step is to focus on the first block in the matrix, and to add up the rows to the final non-zero row of A, in this case the fourth row. Thus $R_1+R_2+R_3+R_4 \longrightarrow R_4$. This cleans up the gap of zeros between the ones in the second column,

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

The next step would be to add our rows upwards beginning at the last nonzero row, which is R_4 in this case. The rows manipulation will be

$$\begin{aligned} R_1 &= R_1 + R_2 + R_3 + R_4 \\ R_2 &= R_2 + R_3 + R_4 \\ R_3 &= R_3 + R_4 \\ R_4 &= R_4 \\ R_5 &= \text{zero row.} \end{aligned}$$

This results in the matrix

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

By adding R_1 to R_2, then adding the new R_2 back to R_1, and lastly adding the new R_1 to R_2[9], we obtain

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

The linearly independent columns, c_1, c_2, c_3, c_5, make up the message portion of the codewords we can send. The fourth and sixth columns, make up the parity check equation portion of the code. In this case, we are able to send 4-bit message words using the first three columns and fifth column, as an encoded 6-bit codewords using the two parity check columns.

By rearranging the columns of the matrix to match a more row reduced echelon form, and removing the trivial zero row, we obtain the matrix

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

We are able to rearrange the columns of the reduced incidence matrix to then convert it into a generator matrix, since all columns of the matrix are unique and there are no two identical columns in any incidence matrix obtained in this paper.

Definition 5.9 (Invertible Matrix)**.** Let U be a square, $n \times n$, matrix. If there exists an $n \times n$ matrix, U^{-1}, such that $U \cdot U^{-1} = I_n = U^{-1} \cdot U$, then the square matrix U is *invertible* and U^{-1} is the *inverse* of U.

Remark 5. Columns in the row reduced echelon form of the matrix are identical if and only if they were identical to begin with.

Let the matrix A be the result of the parallel connection or direct sum of two cycle graphs, and R the row reduced echelon form of A. The series of finitely many elementary row operation that are performed on A can be written as a left multiplication by a invertible vector u_i, such that $\{u_1, ..., u_i\}$ make up the invertible matrix U.

[9]Swapping the position of the first and second rows.

Statement 5.10. There exists an invertible matrix U for every incidence matrix A constructed using our methods, such that $U \cdot A = R$, the row reduced echelon form of A.

Let A be a size $m \times n$ matrix, and row operations are written as invertible vectors denoted by $\{I_{r_i} \mid \ 1 \leq i \leq m\}$, which make up the square matrix U.

$$U = \begin{bmatrix} I_{r_1} \\ I_{r_2} \\ \vdots \\ I_{r_{m-1}} \\ I_{r_m} \end{bmatrix}, \text{ and } U \cdot A = R,$$

such that R is the row reduced echelon form of A.

Proof. Let C_1 and C_2 be columns in A, R_1 and R_2 be the respective corresponding columns in R, and I_r be an invertible vector in U such that $I_r \cdot A = R$.

Assume $C_1 = C_2$ ($\implies$)

$$\begin{aligned} C_1 &= C_2 \\ I_r \cdot C_1 &= I_r \cdot C_2 \\ R_1 &= R_2 \end{aligned}$$

Assume $R_1 = R_2$ ($\impliedby$)

$$\begin{aligned} R_1 &= R_2 \\ I_r \cdot C_1 &= I_r \cdot C_2 \\ I_r^{-1} \cdot I_r \cdot C_1 &= I_r^{-1} \cdot I_r \cdot C_2 \\ C_1 &= C_2. \end{aligned}$$

Thus columns are identical in the row reduced echelon form if and only if they were identical to begin with. □

Example 5.11. Consider the house graph in Example 5.8 composed of the parallel

connection of a C_4 and a C_3 cycle graph. Its incidence matrix is

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

and the invertible matrix composed of the row operations is

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

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

The matrix produced by $U \cdot A$ is in the form we expected, and $A \cdot U$ is not possible since the number of columns in A is not equal to the number of rows in U.

Want to show U is invertible, or compute the determinant of U and check that it is not zero.

$$U^{-1} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}.$$

Now

$$U \cdot U^{-1} = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

In this House Graph Example we can see how the columns in the incidence matrix and its row reduced echelon form are all unique and there are no identical columns. This allows for the relabeling of columns in order to rearrange them in the matrix for the final generator matrix shape, without having to worry about duplicate columns.[10]

Example 5.12. Consider the house graph and its resulting generator matrix

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

The set of message words is $M = \{0000, 0001, 0010, 0011, 0100, 0101, 0110, 0111, 1000, 1001, 1010, 1011, 1100, 1101, 1110, 1111\}$, but the close set of codewords we can transmit are $C = \{000000, 000110, 001011, 001101, 010011, 010101, 011000, 011110, 100011, 100101, 101000, 101110, 110000, 110110, 111011, 111101\}$. The closed set of codewords allow us to use matrix decoding to decode the received codewords into message words after transmission. The decoding parity check matrix composed of the parity check equations $x_5 = x_1 + x_2 + x_3 + x_4$ and $x_6 = x_1 + x_2 + x_3$ is

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

Using the decoding matrix as defined in Definition 2.4, it is possible to right hand vector multiply the received word with the matrix H to obtain the vector that points to the location of the error. However, we do not have enough parity check equations to

[10] A duplicate or identical columns would imply that there exist edges with the same ordered pair of vertices(parallel edges) in the graphs, but parallel edges are not possible in our graphs. Thus it is not possible to have identical columns in the row reduced echelon form of the incidence matrices since parallel edges are not possible.

have unique columns in matrix H. This is caused from the joining graphs in the house graph creating a parity check column that contains all components of the message word except x_4 which in turn made the fourth column of the parity check matrix to be the only distinct column in the first four columns. However, the fourth column is identical to the fifth column and thus, in this code, when decoding by using parity check matrix decoding we will not be able to determine if the error occurred in the parity check component or in the message component of the received word. For example, let $w = [111111]$ represent a word received that was created from the generator matrix above. By parity check matrix decoding, $H \cdot w$ should result in the column vector that matches one in H, and the next step depends on whether it matches on of the first m message components. In this case

$$Hw = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}.$$

This vector matches both the fourth and firth components of H. We are supposed to change the fourth components but since it matches the fifth components, we must assume no errors occurred. If either component is changed, it results in a codeword from C. Thus the code allows us to detect errors, and when an error is detected the code can narrow down which codewords may be the intended one for transmission by using parity check matrix decoding. If the operator receiving the transmitted word cannot determine what codeword was originally sent they may request for retransmission of the codeword.

Although this does not allow for the correction of all errors using just the matrices, it adds further improvement for error detection and correction using generator matrices produced from graphs incidence matrices.

5.2 Direct Sum

Example 5.13. Consider the cycle graph C_3 and its incidence matrix

$$M(C_3) = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix}.$$

In order to join two of these graphs by a vertex we must take the direct sum of the two graphs to form graph $C_{3,3}$, forming the new graph and incidence matrix

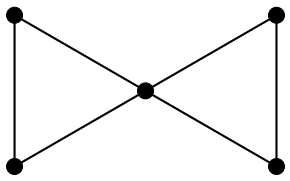


Figure 5.6: $C_{3,3}$ Direct Sum

$$M(C_{3,3}) = \left[\begin{array}{cccccc} 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ \hdashline 0 & 1 & 1 & 1 & 1 & 0 \\ \hdashline 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 \end{array}\right].$$

The middle row of the direct sums incidence matrix, separated by dashed line, we can see the row corresponding to the vertex that connects the two graphs. The single row has four nonzero components which correspond to the four edges are incident to the vertex. By relabeling the columns of the row reduced echelon form of the matrix and deleting the zero row, we obtain the generator matrix G

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

This generator matrix looks like it can send a 4-bit message word and contain two parity-check columns for those four components, but the two parity check-columns each contain

unique components of the message word and don't share any common message components. So the generator matrix send four bit messages, but the parity portion of the generator matrix cannot correct errors since there are no share message components. The matrix works similar to sending two individual 2-bit message words as 3-bit codewords, like the codewords produced from the generator matrix of a C_3 graph. Thus the direct sum of two C_3 cycle graphs produced a generator matrix that sends the codewords of a $C_{3,3}$ graphs generator matrix twice, as one codeword.

The direct sum of two cycle graphs C_n and C_m would produce an incidence matrix of size $(n+m-1)\times(n+m)$, due to the graphs not having a common edges, and sharing a single common vertex. The direct sum of two graphs results in the incidence matrix having the sum of the two graphs edges, and one less vertex than the sum of the two graphs vertices.

Proposition 5.14. The incidence matrix of the direct sum of cycle graphs C_n and C_m, with n and m number of vertices, has size $(n+m-1)\times(n+m)$, with a row reduced echelon form containing $n+m-2$ identity columns, two non-identity columns, and a zero row, but the two non-identity columns are independent of each other and share no components.

Proof. Base Case: We induct on the number of vertices n. Let $n=3$, and C_m be an arbitrary cycle graph with n-number of vertices. The incidence matrix of the direct sum of C_3 and C_m is

$$\left[\begin{array}{ccc:ccccc} 1 & 1 & 0 & & & \mathbf{0} & & \\ 1 & 0 & 1 & & & & & \\ \hdashline 0 & 1 & 1 & 1 & 1 & \ddots & 0 & 0 \\ \hdashline & & & \ddots & \ddots & \ddots & \ddots & \\ & \mathbf{0} & & 0 & \ddots & 0 & 1 & 1 \\ & & & 0 & \ddots & 0 & 0 & 1 \end{array}\right].$$

From Theorem 4.6, and following a similar reducing pattern to the one used in Example 5.8 above[11], the submatrix of C_3 in the top left of the direct sums incidence matrix

[11]The only changes in the reduction pattern are the row operations done on the C_m graphs submatrix. The first row of C_m does not need to be added an additional time since the reducing pattern done in C_3 no longer affects the rows of c_m since they no longer share the second row in C_3 and first row of C_m as a common row in the direct sums incidence matrix, and the sum of the rows of C_3 no longer affects the components in the first row of C_m since they are all zeros.

will row reduce to its row reduced echelon form without changing any of the components of the submatrix C_m in the bottom right of the direct sums incidence matrix

$$\left[\begin{array}{ccc:ccccc} 1 & 0 & 1 & & & \mathbf{0} & & \\ 0 & 1 & 1 & & & & & \\ \hdashline 0 & 0 & 0 & 1 & 1 & \ddots & 0 & 0 \\ & & & \ddots & \ddots & \ddots & \ddots & \\ & \mathbf{0} & & 0 & \ddots & 0 & 1 & 1 \\ & & & 0 & 1 & 0 & 0 & 1 \end{array}\right].$$

The submatrix in the bottom right of the matrix form the matrix of cycle graph C_m with size $m \times m$. Using the modified reduction pattern stated above, the submatrix C_m is reduced to its row reduced echelon form

$$\left[\begin{array}{ccc:cccc} 1 & 0 & 1 & & & \mathbf{0} & \\ 0 & 1 & 1 & & & & \\ 0 & 0 & 0 & 1 & 0 & \ddots & 0 & \mathbf{1} \\ & & & \ddots & \ddots & \ddots & \ddots & \vdots \\ & \mathbf{0} & & 0 & \ddots & 0 & 1 & \mathbf{1} \\ \hdashline & \overline{\mathbf{0}} & & & & \overline{\mathbf{0}} & & \end{array}\right] \longrightarrow \left[\begin{array}{cc:cc} \mathbf{I_2} & \mathbf{1} & \overline{\mathbf{0}} & 0 \\ 0 & \overline{\mathbf{0}} & \mathbf{I_{m-1}} & \mathbf{1} \\ \hdashline 0 & 0 & 0 & 0 \end{array}\right].$$

Thus the direct sums incidence matrix has size $(m+2) \times (m+3)$, and its row reduced echelon form contains $n+m-2$ identity columns, two non-identity columns, a zero row, and the two non-identity columns are independent of each other and share no components.

Induction Hypothesis: Let $n = k$ such that the matrix produced by the direct sum of C_k and C_m form an incidence matrix has a size of $(k+m-1) \times (k+m)$ with a row reduced echelon form containing $k+m-2$ identity columns, two non-identity columns that are independent of each other, and a zero row.

Induction Step: Want to show the direct sum of C_{k+1} and C_m produces an incidence matrix of size $(k+m) \times (k+m+1)$ with a row reduced echelon form containing $k+m-1$ identity columns, two non-identity columns that are independent of each other, and a zero row.

Let $n = k+1$, and C_{k+1} and C_m be arbitrary cycle graphs. The incidence

matrix of the direct sum of graphs C_{k+1} and C_m is

$$\left[\begin{array}{cccccc:cccccc}
1 & 1 & 0 & \ddots & 0 & 0 & 0 & 0 & \ddots & 0 & 0 \\
1 & 0 & 1 & \ddots & 0 & 0 & 0 & 0 & \ddots & 0 & 0 \\
\ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots \\
0 & 0 & 0 & \ddots & 1 & 1 & 0 & 0 & \ddots & 0 & 0 \\
\hdashline
0 & 1 & 0 & \ddots & 0 & 1 & 1 & 1 & \ddots & 0 & 0 \\
\hdashline
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & \ddots & 0 & 0 \\
\ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots \\
0 & 0 & 0 & \ddots & 0 & 0 & 0 & 0 & \ddots & 1 & 1 \\
0 & 0 & 0 & \ddots & 0 & 0 & 0 & 1 & \ddots & 0 & 1
\end{array}\right].$$

In the top left of the matrix, there is the incidence matrix of C_{k+1} as an submatrix. After row operation, $R_1 + R_2 \longrightarrow R_2^*$, the first column will be in its identity form. The submatrix of size k in the top left, formed by the deleting the first left column and first top row,there exists a submatrix of size k, that is the matrix in the induction step. The submatrix of C_k, as seen in the base case reduces to

$$\left[\begin{array}{c:c|ccc:c}
\mathbf{I_{k-1}} & \mathbf{1} & 0 & 0 & \ddots & 0 \\
\hdashline
\overline{\mathbf{0}} & 0 & 1 & 1 & \ddots & 0 \\
\hdashline
\mathbf{0} & 0 & 0 & \ddots & 0 & 0 \\
\vdots & \ddots & \ddots & \ddots & \ddots & \ddots \\
0 & 0 & 0 & \ddots & 1 & 1 \\
0 & 0 & 1 & \ddots & 0 & 1
\end{array}\right].$$

The bottom right of the matrix contains a submatrix of C_m, and from Theorem 4.6 we know row reduces to

$$\left[\begin{array}{c:c|c:c}
\mathbf{I_{k-1}} & \mathbf{1} & \mathbf{0} & \mathbf{0} \\
\hdashline
\overline{\mathbf{0}} & 0 & \mathbf{I_{m-1}} & \mathbf{1} \\
\hdashline
\overline{\mathbf{0}} & \overline{\mathbf{0}} & \overline{\mathbf{0}} & \overline{\mathbf{0}}
\end{array}\right].$$

Thus if we add the trivial row/column that were deleted we obtain a matrix of size

$(k+1+m-1) \times (k+1+m)$ which is $(k+m) \times (k+m+1)$

$$\left[\begin{array}{c|c:c:c} 1 & 0 & 0 & 0 \\ \hline 0 & \mathbf{I_{k-1}} & \mathbf{1} & \mathbf{0} \\ \hdashline 0 & \mathbf{\overline{0}} & 0 & \mathbf{I_{m-1}} \\ 0 & \mathbf{\overline{0}} & \mathbf{\overline{0}} & \mathbf{\overline{0}} \end{array}\right] \longrightarrow \left[\begin{array}{c:c:c:c} \mathbf{I_k} & \mathbf{1} & \mathbf{0} & \mathbf{0} \\ \mathbf{\overline{0}} & 0 & \mathbf{I_{m-1}} & \mathbf{1} \\ \mathbf{0} & \mathbf{0} & \mathbf{\overline{0}} & \mathbf{\overline{0}} \end{array}\right].$$

The row reduced echelon form of the direct sum incidence matrix contains $k+m-1$ identity columns, two non-identity columns that are independent of each other, and a zero row. □

One flaw with taking the direct sum of cycle graphs is the resulting incidence matrix. Since the incidence matrix row reduced into a matrix of size $n \times (n+1)$, and this results in a single parity column and equation for the code. While the addition of a parity column is great for error detection, a single equation does not allow us to correct any errors during transmission. If a graph were constructed and only used direct sum joined graphs, it would have to be constructed using a graph with the least number of columns, or else the incidence matrix will continue to add two or more message columns each time a graph is joined, but only add one parity columns for the entire matrix. This results in the increase of the number of components in a message word, but the parity check equations will not contain the original components of the message word. This adds complexity to the code without much benefit from the parity check equation.

Remark 6. When joining a single graph by direct sum to two graphs are already joined by parallel connection, the direct sum joining graph can attach to any vertex of the parallel connection attached graphs, but it will be difficult to use the incidence matrix pattern stated before. In order to use our pattern, the direct sum joining graph must attach to the last vertex of the parallel connection joined graphs. This will allow the graphs to maintain the original columns pattern and allow for the joining of more graphs by parallel connection or direct sum in the future.

5.2.1 Examples

The following examples are experiments for possible generator matrices constructed by the joining two graphs by direct sum different vertices of the direct sum

graphs[12].

Example 5.15. Let $C_{4,4}$ represent the graph of two C_4 graphs joined by a vertex.

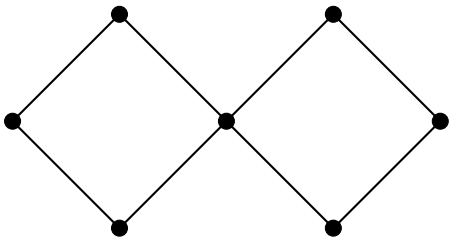
$C_{4,4} =$ 

Figure 5.7: The Graph of $C_{4,4}$

The graph $C_{4,4}$ was the direct sum of two C_4 graphs, and in the incidence matrix we can see the patter for joining in order to allow for future graphs to join. We use the same pattern as seen in Remark 2 for both C_4 graphs and join them such that the first vertex v_1 of the second graph joins with the last vertex v_4 of the first vertex resulting in the incidence matrix

$$M(C_{4,4}) = \left[\begin{array}{cccccccc} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ \hdashline 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ \hdashline 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \end{array}\right].$$

joining in this pattern allows for the final columns of the incidence matrix $M(C_{4,4})$ to end with two adjacent $1s$, which allows for another cycle or tree graphs incidence matrix to join to $M(C_{4,4})$. It also leaves the first column beginning with to adjacent ones, which allows for $M(C_{4,4})$ to be joined to another graphs incidence matrix as the second matrix.

Let P_3 represent the path graph with 3 vertices with a graph and incidence matrix in the form

[12]Such as the differences between joining one graph to an another graphs vertex that is adjacent to two edges, compared to a vertex that is already shared between two sub graphs and is adjacent to four edges.

Figure 5.8: The Graph of P_3

$$\begin{bmatrix} 1 & 0 \\ 1 & 1 \\ 0 & 1 \end{bmatrix}.$$

From Section 4.1.1 we know that the incidence matrix of tree graphs have all identity columns.

The direct sum of the end leafs of two path graphs will produce a new path graph, but if the graphs were joined by a vertex that was not an end leaf it will result in a tree graph that is no longer a path. For example, let $2P_3$ represent the direct sum of two P_3 graphs,

$$2P_3 = \left[\begin{array}{cc:cc} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{array}\right].$$

As we can see in the incidence matrix, the joining of two path graphs by their end leaf results in a larger path graph. The number of vertices in the new path graph is one less than the sum of its previous two graphs vertices. The rank of the direct sums matrix is the sum of the rank of the two graph that make it up.

5.2.2 Joining To End Leaf

Suppose that we join three C_4 graphs by their vertices in a path.

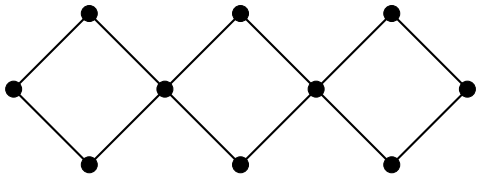


Figure 5.9: The Graph of $C_{4,4,4}$

The incidence matrix of graph $C_{4,4,4}$ will be of the form

$$M(C_{4,4,4}) = \left[\begin{array}{cccc:cccc:cccc} 1&0&1&0&0&0&0&0&0&0&0&0 \\ 1&1&0&0&0&0&0&0&0&0&0&0 \\ 0&1&0&1&0&0&0&0&0&0&0&0 \\ \hdashline 0&0&1&1&1&0&1&0&0&0&0&0 \\ \hdashline 0&0&0&0&1&1&0&0&0&0&0&0 \\ 0&0&0&0&0&1&0&1&0&0&0&0 \\ \hdashline 0&0&0&0&0&0&1&1&1&0&1&0 \\ \hdashline 0&0&0&0&0&0&0&0&1&1&0&0 \\ 0&0&0&0&0&0&0&0&0&1&0&1 \\ 0&0&0&0&0&0&0&0&0&0&1&1 \end{array}\right].$$

By using row reduction and following the pattern we found in Remark 4, the matrix above is equivalent to a generator matrix for codewords. For example, the matrix above can be simplified to its row reduced echelon form and by rearranging the non-identity columns of the matrix, we obtain the generator matrix

$$G(C_{4,4,4}) = \left[\begin{array}{ccccccccc:ccc} 1&0&0&0&0&0&0&0&0&1&0&0 \\ 0&1&0&0&0&0&0&0&0&1&0&0 \\ 0&0&1&0&0&0&0&0&0&1&0&0 \\ \hline 0&0&0&1&0&0&0&0&0&0&1&0 \\ 0&0&0&0&1&0&0&0&0&0&1&0 \\ 0&0&0&0&0&1&0&0&0&0&1&0 \\ \hline 0&0&0&0&0&0&1&0&0&0&0&1 \\ 0&0&0&0&0&0&0&1&0&0&0&1 \\ 0&0&0&0&0&0&0&0&1&0&0&1 \\ \hline 0&0&0&0&0&0&0&0&0&0&0&0 \end{array}\right].$$

This generator matrix denoted G can can encode length 9 messages into length 12 codewords.

Due to the nature of the parity-check columns being in groups of three and not interacting with each other, the detecting characteristics of this code are strange. The code can detect a single error in each of these groups, but cannot detect when two errors occur in a group. Although this allows us to detect more errors, there is the issue of the codeword now being broken into three different segments after transmission. If we detect an error in the first segment but don't detect an error in the other two, we cant be sure that they were transmitted correctly.

There could be some usefulness in this type of code. The code could be used to send three message that are each three bits long to fill the 9-bit message word. This would allow the parity-check columns to check each segment as an individual code. If the codeword cannot easily be re-transmitted, then the three 3-bit message words can be repeated as a 9-bit message word so that there is redundancy to the message being sent and redundancy with the error detection.

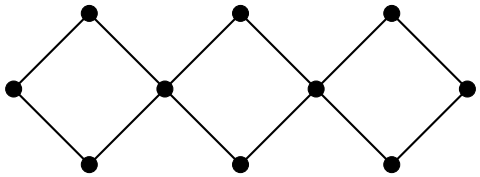
Example 5.16. For example, let's transmit the 3-bit message (001) using the generator matrix above, and the message word used is (001001001). The message word would be encoded as the codeword (001001001111).

The redundancy in the message being sent allows us to choose the message the was most common. If there was an error in one of the segments, but two other are identical, then we can assume the majority message to be the correct one. This also

shows the flaw with detecting more than one error. If two different message segments get the same error, we will receive two segments that are wrong and one correct one. This could make us decode the incorrect message when choosing from the majority of wrong transmission that were assumed to be correct.

5.2.3 Joining To Shared Vertex

Consider the graph of $C_{4,4,4}$ in Section 5.2.2



Question 5.17. What would happen if another C_4 graph was joined to a vertex[13] in the middle of the $C_{4,4,4}$ graph, instead of an end leaf?

Let DS_4 denote such a graph, and be of the form

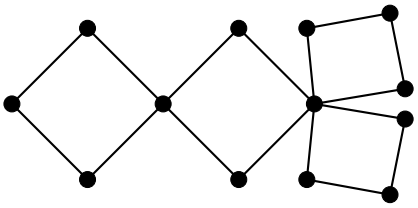


Figure 5.10: The Graph of DS_4

[13] A vertex with degree greater than two that us shared by the C_4 subgraphs.

The incidence matrix of such a graph will be of the form

$$M(DS_4) = \left[\begin{array}{cccc:cccc:cccc:cccc}
1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\hdashline
0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\hdashline
0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\hdashline
0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\
\hdashline
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\
\hdashline
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1
\end{array}\right].$$

When row reducing the matrix, the blocks in the matrix that correspond to the end leafs must be simplified first. In this case, the last three rows can be added to the seventh row to make the last three columns of the seventh row zeroes. We can then add $R_{12} + R_{11} \to R_{11}$. This simplifies that last four columns into their row reduced form. From then we can follow our pattern of viewing and working with the matrix by blocks. Each block corresponds to a cycle graph, and are equivalent to a 3×3 identity matrix with a fourth column of ones and the fourth row of each block is a zero row. This allows us to get the row reduce echelon form with a zero row in R_9. The row reduced echelon form of the matrix is creates the generator matrix

$$G(DS_4) = \left[\begin{array}{cccccccccccc:cccc}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
\hdashline
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
\hdashline
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
\hdashline
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1
\end{array}\right].$$

Here we can see how the first twelve columns from the left are in their identity form and the final four columns make up the parity check equation columns. Let the matrix columns be numbered one through sixteen starting from the left. The thirteenth through sixteenth columns can be written as the linear combination of columns one through twelve. The non-identity columns after the identity columns follow a similar pattern and create the parity check equations $R_{13} = R_1 + R_2 + R_3$, $R_{14} = R_4 + R_5 + R_6$, $R_{15} = R_7 + R_8 + R_9$, and $R_{16} = R_{10} + R_{11} + R_{12}$.

Since the parity check equations have no overlap in the columns that make them up, there is no way of using multiple parity check equations to check for errors. This code would simply allow us to check for single input errors or the rare cases where all three message inputs go through interference and are changed. The case of two changes in the message occurring can occur, but cannot be detected using a single parity check equation in this code because we are working under $\mathbb{F}_2$. Two changes in the inputs of the codeword would cancel each other out in the parity check equation and would result in the same input as a correct codeword would.

Such a code could be used as a repetitive code to send the same three-bit message multiple times and use parity check matrix decoding to determine if any of the messages had interference and errors. Let H be the parity check matrix of $G(DS_4)$ obtained from

its parity check equations

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

Assume we received the word[14]

$$w^T = \begin{bmatrix} 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix},$$

then

$$Hw = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 0 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}.$$

The vector would not be able to show us which specific component had an error, but we can tell that the error occurred in the first three components of the message. We can ignore that section of the received code and use another three-bit message portion of the code that was not affected to determine what the original transmitted code was.

[14] A received word is written as a column vector, and w^T is the transpose of w and is only written as a row vector here due to page layout constraints.

Although it is not possible to correct errors using the code and its parity check equations, this type of code offers another choice for codes that can detect errors, and can determine the original codeword if an operator is present.

Chapter 6

Conclusion

The codes generated from joining cycle graphs through direct sums provides a code that is useful for repetitive transmissions. This type of code allows for there to be a single parity check equation for each "cycle graph" that s in the matrix. This does not allow us to create complex parity check equations that can correct errors, but due to the repetitive nature of these codes, if the code is sent multiple times then the receiver will be able to check each cycles single parity check input and use the repeated codewords as a second form of checking the transmission for errors. This method is not very efficient for a single person to manually read the decoded messages, but it is simpler process for transmission of information that is not time sensitive, does not need encoding, and can be retransmitted if necessary.

At the beginning of the thesis I believed that a combination of both parallel connection and direct sums would produce a matrix with a large enough message portion, and enough redundancy portion to detect errors without being too large. By joining using the direct sum of a path or tree graph, it will help with increasing the number of identity columns without affecting the parity-check columns of the generator matrix. When joining using parallel connection with a cycle graph, we can increase the number of identity columns, as well as add a parity column to the resulting generator matrix. After seeing the generator matrices produced by the direct sum of tree graphs, it is unreasonable to use tree graphs to construct generator matrices since they do not produce any redundancy portion to their generator matrix. Although joining tree graphs can increase the length of the message words, simply increasing the message word without adding redundancy

columns causes problems by increasing the complexity of encoding and decoding as well as increasing the number of possible error locations. Cycle graphs joined by parallel connection allow us to increase our redundancy columns, but increase the number of message columns as well.

Cycle graphs joined by a combination of direct sum and parallel connection will produce interesting generator matrices with multiple forms of redundancy. Parallel connections can allow us to increase our redundancy columns in the generator matrix produced to a desired amount. By joining a cycle graph by direct sum to the incidence matrix of the parallel connections, we can think of the identity portion of the direct sums matrix as a clone of the parallel connection portion and the independent parity check column as a redundancy column separate to the parity columns produced from the parallel connection graphs. Although the message portion is increased similar to how the tree graphs do, the increase in the message portion of the cycle graphs direct sum is solely used as redundancy.

Example 6.1. Let E represent the graph of the parallel connection of three triangles, and the direct sum of the triangles with a pentagon. First, the incidence matrix of three

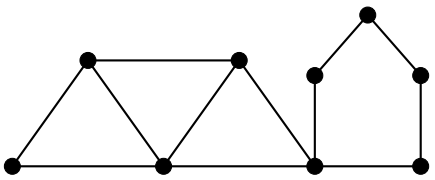


Figure 6.1: Graph E

C_3 cycle graphs and that of a pentagon are

$$\begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix} \text{ and } \begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 \end{bmatrix}.$$

The direct sum of these two graphs results in the incidence matrix

$$M(E) = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \end{bmatrix}.$$

This results in the generator matrix G

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

The generator matrix should be seen as sending the same length four message word twice, with two different redundancy portions. The first four message components have three parity check equations that allow for the detections and locating of errors in three different locations. The second half of four message components contain a single parity check equation that sends offers single error detection, as well as offers a redundant copy of the message word in the transmitted codeword. Although the increase in the lengths of message words typically comes with an increase in message words, since the same length 4 message word is sent twice, the number of possible message words and

codewords is 2^4, and not 2^8. The parity check decoding matrix H is

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

As we can see in the first four columns, the first two columns would not allow us to determine which of those two the error occurred in, but we would be able to correctly detect and correct errors in the third or fourth components of the message word and can use the redundant message to determine an error between the first two components of the message. The generator matrix G is not being used in the conventional form where all k message components form a single message word. Although the code is not capable of detecting and correcting errors without human input, the code has much more use cases with the additional redundancy and can be implemented into real applications where sixteen or less messages are required.

6.1 Possible Outcomes

By only using direct sum of cycle graphs, it's possible to construct codes that strictly rely on repetition as its main form or redundancy. The redundancy columns of the matrix are the first form of error detection and can be used to determine the correct message by simply voting by majority. The cycle graphs create repetitions of the message and parity columns. Thus if any of the repetitions contains errors, the operator on the receiving end of the word can simply count the like versions of the message that were received and determine the majority message as the correct one. For such a code, the more repetitions being sent, the less errors the majority count would output, but it also increases the length of the codeword[1] as well as increases the likelihood for human error during the decoding process. This type of code can be useful, but the creator and person

[1] An increase in message length typically only increases the complexity, but in this case each graph incidence submatrix becomes independent and are sent as copies of each other. The main message from such a code, composed of the direct sum of multiple C_k cycle graphs, has length $k-1$ and is sent as many times as there are graphs.

implementing would have to determine a balance between the amount of redundancy and the complexity and length of codewords.

The second type of code is one constructed using parallel connections of cycle graphs. From Proposition 5.5 we saw how each cycle graphs constructed a parity check equation that included every row except one of its own. This allows for the possible detection an correction of errors that occurred at that rows component of the codeword. By including more cycle graphs, we would create more components where errors can be detected and corrected, improving the code. Since each cycle graph must adds message components with each parity columns, it has drawbacks such as increasing generator matrix complexity and overall codeword length. By using a combination of parallel connection and direct sum, the benefits of both types of joining[2] can be obtained without the drawbacks of using both individually.

[2]The parity check columns of parallel connections, and the redundancy of repeated message portions fro direct sums.

Chapter 7

Expository Work

SageMath Code

```
def three_sum_incidence_matrix(size1, size2):
'''
Inputs:
    size1: The size of the first cycle graph.
    size2: The size of the second cycle graph, the joining graph.
Returns:
    Produces the unreduced incidence matrix of the parallel connection of graphs
    size1 and size2 (combined_matrix).
'''
# Create single-entry row vectors of size size1 and size2
row_vector1 = matrix(1, size1)
row_vector2 = matrix(1, size2)

# Generate cycle graphs of size size1 and size2
cycle_graph1 = graphs.CycleGraph(size1)
cycle_graph2 = graphs.CycleGraph(size2)

# Get the incidence matrices of the cycle graphs
incidence_matrix1 = cycle_graph1.incidence_matrix()
incidence_matrix2 = cycle_graph2.incidence_matrix()

# Convert the incidence matrices to GF(2)
gf2_matrix1 = matrix(GF(2), incidence_matrix1)
gf2_matrix2 = matrix(GF(2), incidence_matrix2)

# Initialize temporary matrices
temp_matrix1 = row_vector1
temp_matrix2 = row_vector2
```

```
# Extend temp_matrix1 by stacking copies of row_vector1 (abs(size2-3) times)
# for _ in range(abs(size2 - 3)):
    temp_matrix1 = temp_matrix1.stack(row_vector1)
extended_matrix1 = gf2_matrix1.stack(temp_matrix1)

# Extend temp_matrix2 by stacking copies of row_vector2 (abs(size1-3) times)
# for _ in range(abs(size1 - 3)):
    temp_matrix2 = temp_matrix2.stack(row_vector2)
extended_matrix2 = temp_matrix2.stack(gf2_matrix2)

# Remove the first column from extended_matrix2
extended_matrix2 = extended_matrix2.delete_columns([0])

# Augment extended_matrix1 with extended_matrix2 to form the unreduced
# incidence matrix of the parallel connection
combined_matrix = extended_matrix1.augment(extended_matrix2)

# Return the final matrix
return combined_matrix
```

The code above was used to join two cycle graphs A and B with sizes c and d respectively. During the testing process, both graphs sizes ranged from 3 to 15, and were joined by a common edge resulting in a matrix of size $(a+b-2) \times (a+b-1)$. [1]

Question 7.1. Is it possible to construct generator matrices for codes that can both detect and correct multiple errors that may occur during transmission?

Yes, it is possible. It is possible to begin with graphs and construct a matrix that can be used as a generator matrix for the encoding and transmission of message words as codewords. These generator matrices work and output codewords, but the simplicity makes the codes too inefficient for modern use. These codes can encode long message words and detection of some errors, but don't allow for reasonable and useful message lengths with the ability to correct errors.

[1] One restriction with the current code is that the graph that we begin with, graph A, must be of the same size or larger than graph B. If graph B is larger, then the resulting matrix from row reducing will be different than what was expected. Although the cause for this is not yet known, the results from the code and the equation used to determine the graphs incidence matrix size are correct when done properly.

7.1 Possible Generator Matrices From Cycle Graphs

The incidence matrices are constructed by fixing a matrix rank, fixing vertex-labeling pattern, and only using parallel connection to join graphs for their incidence matrix. The first graph must be greater or equal to the same size as the rest of the graphs being joined to it.[2]

Let $Rank = 2$.

Triangle.

$$\begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix} \longrightarrow \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 0 \end{bmatrix}.$$

The triangle is the smallest possible cycle graph we can use and is the only cycle graph with a rank of 2. This type of matrix can be used as a generator matrix to send two-bit messages (11) as three-bit codewords (110). This does not allow for error correction and we can only detect when a single error occurs in one of the two message bits. If both message bits were to change during transmission, we would not be able to detect the error and would have to use the incorrect message or request the message to be re-sent so we can compare the first message received with the second.

Let $Rank = 3$.

Two triangles

$$\begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 \end{bmatrix} \longrightarrow \begin{bmatrix} 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

This type of matrix can be used to send three bit messages as five-bit codewords. After deleting the zero row of the row reduced incidence matrix, the result is a 3×5 generator matrix containing two parity check columns that share all components except one. This

[2]This rule is required by the parallel connection sagemath code, but does not need to be followed if not using the sagemath code to construct matrices.

allows for the received messages to detect errors, but it will not be able to determine the exact location of the errors and correct them. An example of this can be seen with the House Graph Example in Example 5.12.

Square

$$\begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \end{bmatrix} \longrightarrow \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}.$$

This matrix can encode four-bit message words into five-bit codewords. Similar to the generator matrix produced from the triangle graph, this code is not very useful since it can only detect when a single error occurs during transmission, but cannot correct the error. If more than one error were to occur during transmission, the best and sometimes only course of action would be to request for it to be re-sent.

Let $Rank = 4$.

Three triangles

$$\begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix} \longrightarrow \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 \end{bmatrix}.$$

The generator matrix constructed by the three triangles incidence matrix contains three unique parity check equations. The three equations help produce a minimum difference of three for the code. All three equations contain the first two components, the first two columns of the message portion, which means parity check equations will not be able to determine which of the two components an error occurred in if one does happen during transmission. This code is able to detect errors in the first two components, the third, and the fourth component since it cannot differentiate between the first two components, and with an operator on the receiving end should be able to correct single errors caused

during transmission.

Pentagon

$$\begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 \end{bmatrix} \longrightarrow \begin{bmatrix} 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 \end{bmatrix}.$$

The pentagon is a simple graph that contains a single parity check equation with all message components making it up. The generator matrix produced from this graphs incidence matrix can only add a simple point of redundancy. This type of code can simply detect odd number of errors in the message portion, but cannot correct any received words correctly. Due to the single parity equation containing all components, the code cannot determine where an even number of errors occur since any combination of even errors results in a codeword since all codewords form a closed set by word addition over $\mathbb{F}_2$. When a codeword undergoes an odd number of errors during transmission, the code can detects them, but it cannot determine if the errors occurred in the message portion, in the redundancy portion, or both for words with multiple errors.

House graph

$$\begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix} \longrightarrow \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 \end{bmatrix}.$$

An example of the code produced by the parallel connection of a square and triangle graph can be seen in Example 5.12.

Let $Rank = 5$.

Hexagon

$$\begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 \end{bmatrix} \longrightarrow \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix}.$$

Similar to the previous simple graphs above, a hexagons incidence matrix results in a generator matrix with a single parity check containing all components of the message portion. At the message length produced by the hexagon, there are $2^5 = 32$ possible message words and codewords. The length of the message words results in a larger number of possible error locations without any benefit when compared to the shorter message words and codewords from the simple triangle or square above.

Four Triangles

$$\begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix} \longrightarrow \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

Similar to the generator matrix of three triangles above, the generator matrix obtained by the parallel connection of four triangles increases the number of locations we can detect errors by one, as well as increases the message word length by one. Although the code allows us to detect another error location by the added parity equation, the number of message words, and possible incorrect words double with the increase in message length.

1-Square and 2-Triangles

$$\begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix} \longrightarrow \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \end{bmatrix}.$$

This codes generator matrix is actually very similar to the generator matrix produced by the parallel connection of three triangles from rank-4. This code allows for similar parity equations as three triangles would, but with worse error detection between the first three components. The three equations of this generator matrix contained the same components, the first three components of the message portion. This implies, that when decoding using parity check equation decoding, we will not be able to determine which of the first three components is incorrect if an error is detected. This code is worse than the code produced by three triangle since its error detection capabilities are worse due to the parity check equation all containing the first three components in them. This causes the parity check equation matrix used in decoding to have its first three columns be identical and not allow for the locating errors in the first three message components, even if we know an error exists.

1-Pentagon and 1-Triangle

$$\begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix} \longrightarrow \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 \end{bmatrix}.$$

The two parity check equations of this code share all components except one. This implies that when decoding using parity check matrix decoding if an error is detected and is one of the first four message components, then parity check matrix decoding will not allow us to determine which of the first four components was the incorrect one since those four

columns of the parity check matrix are identical.

Let $Rank = 6$.

Heptagon

$$\begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \longrightarrow \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix}.$$

A heptagon produces a generator matrix similar to the hexagon cycle graphs generator matrix above. Again, such a code is not effective encoding and doesn't offer any useful error detection capabilities, aside from single error detection, due to having a single parity column.

Two-story house

$$\begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix} \longrightarrow \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \end{bmatrix}.$$

The two-story house graph produces a generator matrix with three parity check equations. The parity check equations produce a parity check matrix whose first three columns are identical, and whose fourth and fifth columns are identical. This will cause any errors that are found using parity check matrix decoding to be grouped into being an error in the first three components, an error in the fourth or fifth, or an error in the sixth column that represents the last component of the message portion.

1-Pentagon and 1-Square

$$\begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \end{bmatrix} \longrightarrow \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \end{bmatrix}.$$

The pentagon and square worsen the detection capabilities of the previous incidence matrix above by reducing the number of parity equations to two. This causes the parity check matrix of this generator matrix to have identical first through fourth columns, and identical fourth and fifth columns. Because of this, any errors detected using parity check matrix decoding will group errors into being one of the first four components or as being an error in the fourth or fifth component of the received word, we cannot locate the exact component that an error occurred in.

1-Hexagon and 1-Triangle

$$\begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix} \longrightarrow \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \end{bmatrix}.$$

The code produced from one hexagon and one triangle does not improve the error detection capabilities compared to the previous generator matrix. The parity check equations produce a parity check matrix whose first five columns are identical, and the sixth column is identical to the last parity column, the eighth column. As states before, this implies we cannot determine where an error occurred during transmission and the operator will be left with no other choice than to request for the codeword to be re-sent.

Five triangles

$$\begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix} \longrightarrow \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

Bibliography

[Fra67] John B. Fraleigh. *A first course in abstract algebra.* Addison-Wesley Publishing Co., 1967.

[GM12] Gary Gordon and Jennifer McNulty. Cambridge University Press, 2012.

[Lev21] Oscar Levin. *Discrete Mathematics: An Open Introduction.* Independently Published, 2021.

[Oxl11] James Oxley. *Contemporary Abstract Algebra.* Oxford University Press, Belmont, CA, 2 edition, 2011.