Version of Record: https://www.sciencedirect.com/science/article/pii/S030439752200127X
Manuscript_7a7eee8b20a1e572014d206bd85329d5

On the Complexity of and Solutions to the Minimum Stopping and Trapping Set Problems

Alvaro Velasquez,
Information Directorate,
Air-Force Research Laboratory,
Rome, NY
alvaro.velasquez.1.@us.af.mil

K. Subramani,
LCSEE,
West Virginia University,
Morgantown, WV
k.subramani@mail.wvu.edu

Piotr Wojciechowski,
LCSEE,
West Virginia University,
Morgantown, WV
pwojciec@mail.wvu.edu

Abstract

In this paper, we discuss the problems of finding minimum stopping sets and trapping sets in Tanner graphs, using integer linear programming. These problems are important for establishing reliable communication across noisy channels. Indeed, stopping sets and trapping sets correspond to combinatorial structures in information-theoretic codes, which lead to errors in decoding once a message is received. In particular, these sets correspond to variables that are not eventually corrected by the decoder, thus causing failures in decoding when using iterative algorithms. We present integer linear programs (ILPs) for finding stopping sets and several trapping set variants. Furthermore, we prove that two of these trapping set problem variants are **NP-hard** for the first time. Additionally, we analyze these variants from the parameterized perspective. Finally, we model stopping set and trapping set problems as Integer Linear Programs (ILPs). The effectiveness of our approach is demonstrated by finding stopping sets of size up to 48 in the $(4896, 2474)$ Margulis code. This compares favorably to the current state-of-the-art, which finds stopping sets of size up to 26. For the trapping set problems, we show for which cases an ILP produces solutions efficiently and for which cases it performs poorly. The proposed approach is applicable to codes represented by regular and irregular graphs alike [1].

1 Introduction

The use of codes as mathematical formalisms lies at the heart of reliable communication. These codes provide a fault-tolerant structure to messages by appending redundant check bits via a generator matrix. After these bits have been added, the codeword is sent across a noisy channel, such as the binary erasure channel (BEC), binary symmetric channel (BSC), and additive white Gaussian noise (AWGN) channel. These channels differ in the way noise is modeled. For example, the BEC introduces probabilities of bit erasures while the BSC accounts for bit flips. Once the message arrives at its destination, it is decoded via a parity-check matrix. It is during this decoding step that certain combinatorial structures in the code lead to errors. In this paper, we are concerned with

[1] An extended abstract of this work appeared in ISCO 2018 [1]

finding such structures. This problem is central to the design of better codes and, consequently, more reliable communication.

One of the principal metrics used to test the reliability of a code is the bit error rate (BER). This corresponds to the number of bit errors per unit time. It is well-known that the BER of a code decreases as the signal-to-noise ratio increases. However, there is a point at which the BER plateaus. This region in the performance curve is known as the error floor of the code. Experiments were performed in [2] to analyze the error floor of the BSC and AWGN channel. It was demonstrated that all decoding failures in these channels were due to the presence of trapping sets. These sets correspond to variables that are not eventually corrected by the decoder, thus causing failures in decoding when using iterative algorithms [3]. Similar experiments have demonstrated that decoding failures over the BEC are due to the presence of stopping sets [4]. More specifically, it is the smallest stopping and trapping sets that cause poor decoding performance in their respective channels [5]. As such, it is important to develop codes that avoid such structures and to efficiently find these structures in existing codes. This remains a pervasive issue even when the connectivity matrix of the code is sparse, as is the case with popular low-density parity-check (LDPC) codes where there is significant performance degradation due to the presence of these structures [6].

Stopping sets and trapping sets are defined by simple combinatorial structures in the graph representation of the underlying code. There have been efforts to estimate their minimum size [7, 8], find the minimum sets [9], and enumerate such sets for up to some small-size parameter [10]. In this paper, we make two contributions. First, we propose integer linear programming solutions to these problems. We improve on the results of [9], which demonstrate that the minimum stopping set in the $(4896, 2474)$ Margulis code [11] is of size 24 and that no stopping sets of size 25 and 26 exist. Our results establish that the next largest stopping sets in said code are of sizes 36 and 48 (see Figures 11 and 12). As a point of reference, the number of points in the search spaces for finding stopping sets of sizes 26, 36, and 48 are $\binom{4896}{26} \approx 2 \times 10^{69}$, $\binom{4896}{36} \approx 1.61 \times 10^{91}$, and $\binom{4896}{48} \approx 8.28 \times 10^{115}$, respectively. This demonstrates the efficiency of a programming-based approach to explore much larger parameter spaces than competing methods. The second contribution we make pertains to the previously unknown complexities of two trapping set variants. We prove that these variants are **NP-hard**, thereby rounding out the complexity results in the literature. We also analyze these problems from the parameterized perspective.

The remainder of this paper is organized as follows. Section 2 provides background information and definitions for the stopping and trapping set problems. A brief exposition of related work is provided in Section 3. Computational complexity results for these problems are presented in Section 4. An analysis of these problems from a parameterized perspective is presented in Section 5. Our approach based on integer linear programming follows in Section 6, and experimental results are shown in Section 7. We conclude and allude to future work in Section 8.

2 Preliminaries

An (n, k) code is one whose codewords have length n and whose dimension is k. The dimension k of the code specifies the number of linearly independent codewords that form the basis $\underline{c}_1, \ldots, \underline{c}_k \in \{0,1\}^n$ for the code. That is, any codeword can be expressed as a linear combination of these basis vectors. Any given codeword of length n contains k original bits of information and $(n-k)$ redundant check bits that are used to detect and correct errors that arise during message transmission across a noisy channel.

Given an (n, k) code, its corresponding parity-check matrix $H \in \{0,1\}^{m \times n}$, where $m = n - k$, defines the linear relations among codeword variables. Each column in H corresponds to a bit in the codeword and each row corresponds to a redundant check bit. Given a codeword $\underline{c} = c_1, \ldots, c_n$, the entry H_{ij} is 1 if c_j is involved in a check operation, which is used to detect errors after transmission. For any such matrix H, let $G = (V \cup C, E)$ denote its representation as a bipartite graph, where $V = \{v_1, \ldots, v_n\}$ and $C = \{c_1, \ldots, c_m\}$ are the sets of variable and check nodes, and $E = \{(v_i, c_j) | H_{ji} = 1\}$ defines the adjacency set. This is known as the Tanner graph of a code. We can now define the trapping and stopping set problems.

Definition 1 ((a, b)-trapping set)**.** *Given a Tanner graph $G = (V \cup C, E)$, an (a, b)-trapping set $T \subseteq V$ is a set of variable nodes with cardinality $|T| = a$ such that b neighbors of T are connected to T an odd number of times [12].*

Definition 2 (Stopping set)**.** *Given a Tanner graph $G = (V \cup C, E)$, a stopping set $S \subseteq V$ is a set of variable nodes such that all neighbors of nodes in S are connected to S at least twice [9].*

As an example, suppose we are given the parity-check matrix H below with Tanner graph $G = (V \cup C, E)$. We can determine the minimum stopping set $S = \{v_7, v_9\}$ as pictured in Figure 1. For ease of presentation, we often argue in terms of induced subgraphs. Given a graph $G = (V \cup C, E)$, we denote by G_S the subgraph induced by the set $S \subseteq V$. That is, $G_S = (S \cup C^S, E^S)$, where $C^S = \{c_j \in C | s_i \in S, (s_i, c_j) \in E\}$ and $E^S = \{(s_i, c_j) | s_i \in S, (s_i, c_j) \in E\}$.

$$H = \begin{pmatrix} 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix}$$

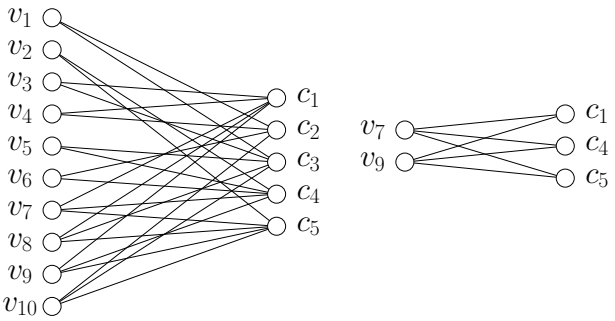

Figure 1: (*left*) Parity-check matrix H for some code. (*center*) Tanner graph $G = (V \cup C, E)$ of H. (*right*) Subgraph G_S induced by the minimum stopping set $S = \{v_7, v_9\}$ in G.

It is worth noting that there exist several trapping set variants. The most popular variant is known as an elementary trapping set. These sets, defined below, are frequently the structures that lead to iterative decoding

failures [13]. That is, the minimum trapping sets in a Tanner graph are often of elementary form. See Figure 2 for an example. An interesting lower-bound result that helps explain this prevalence was recently established in [14], where it was shown that the lower bound on the size of non-elementary trapping sets is greater than that of their elementary counterparts.

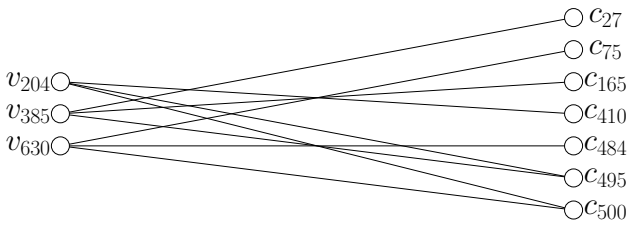

Figure 2: Elementary $(3, 5)$-trapping set found in the $(1008, 504)$ Mackay code [15]. Note that, of the seven neighbors of $T = \{v_{204}, v_{385}, v_{630}\}$, only five are connected exactly once to T.

Definition 3 (Elementary (a, b)-trapping set)**.** *Given a Tanner graph $G = (V \cup C, E)$, an elementary (a, b)-trapping set $T \subseteq V$ is a set of variable nodes with cardinality $|T| = a$ such that b neighbors of T are connected to T exactly once and the remaining neighbors are connected to T exactly twice [13].*

For both trapping sets and elementary trapping sets, we refer to b as the cost of T. For the remainder of this paper, we adopt the notation $[k]$ to denote the set $\{1, 2, \ldots, k\}$.

We are concerned with the computational complexities of and solutions to the following optimization problems:

1. SS: Find a minimum-cardinality stopping set $S \subseteq V$.
2. TS$(a, b)|_a$: Find a trapping set $T \subseteq V$ such that $|T| = a$ and b is minimized.
3. TS$(a, b)|_b$: Find a minimum-cardinality trapping set $T \subseteq V$ for a given b.
4. ETS$(a, b)|_a$: Find an elementary trapping set $T \subseteq V$ such that $|T| = a$ and b is minimized.
5. ETS$(a, b)|_b$: Find a minimum-cardinality elementary trapping set $T \subseteq V$ for a given b.

In particular, we are interested in the TS$(a, b)|_b$ and ETS$(a, b)|_a$ problems. Accordingly, the principal contributions of this paper are as follows:

1. Proving that the TS$(a, b)|_b$ problem is **NP-hard**.
2. Proving that the TS$(a, b)|_b$ problem is **para-NP-hard**, when parameterized by the maximum degree of the given Tanner graph.
3. Proving that the ETS$(a, b)|_a$ problem is **NP-hard**.

4. Proving that the ETS$(a, b)|_a$ problem is **para-NP-hard**, when parameterized by the maximum degree of the given Tanner graph.

5. Proving that the ETS$(a, b)|_a$ problem is **W[1]-hard** with respect to a.

6. Designing and analyzing an $O^*(|V|^a)$/$O^*(2^{|V|})$ algorithm for the ETS$(a, b)|_a$ problem.

3 Related Work

Many of the methods proposed in the literature stem from work presented in [16], wherein a search of the code's iterative decoding trees is performed. Boolean functions are defined on these trees to calculate the bit error rate of a code. Any time the bit error rate is 1, there are one or more stopping sets in the decoding tree. By taking as input a parity-check matrix H and a subset of variable nodes $V' \subset V$, this method outputs the stopping sets that contain V'. This method is used to find stopping sets S of size $|S| \leq 13$ in codes of length up to $n \approx 500$. An extension has been proposed in [17] to enumerate trapping sets of size $|T| \leq 11$. Based on this approach, a branch-and-bound algorithm for enumerating stopping sets is presented in [9, 18], wherein the lower bound on the size of the sets is computed via linear programming relaxations. Each variable can be constrained to be included in or excluded from the stopping set, or it can be unconstrained. The branching step chooses the unconstrained variable node v with neighboring check nodes connected to the most unconstrained nodes. This creates two new constraints. Namely, the ones that arise by including/excluding v in the stopping set. This approach was used successfully to search for stopping sets of size up to 26 in codes with thousands of nodes, like the $(4896, 2474)$ Margulis code [11]. A similar algorithm is proposed for finding trapping sets in [10]. Since modern integer programming solvers use similar branch-and-bound techniques complemented with cutting plane algorithms, we conjecture that the aforementioned approach is subsumed by a purely integer programming-based method like the one we present in Section 6. It is worth noting that solutions based on mathematical programming have also been adopted in a series of works for computing the related coding problems of the minimum distance of a code and for the maximum likelihood decoding problem [19, 20, 21, 22, 23], where the former problem seeks to determine the minimum Hamming distance between any two codewords [24].

In [25], it is demonstrated that an efficient procedure to find stopping sets can be used as a subroutine to find trapping sets. Given a Tanner graph G, the algorithm proceeds by finding k edges such that no two edges share an endpoint. The parity-check matrix is then modified to eliminate nodes in G that are not endpoints in this set. The algorithm from [16] is called using this modified parity-check matrix and endpoint nodes as input. This solves the ETS$(a, b)|_b$ problem with a runtime of $\mathcal{O}(n^b)$, where n is the length of the code. Another effective approach which finds trapping sets of increasing size from an initial set of small trapping sets is presented in [26]. This method exploits the relationship between trapping sets and cycles in the Tanner graph, which has been studied in [27] [28]. However, while it is very efficient at finding small trapping sets, its reliance on cycle enumeration as input makes it inadequate for large problem instances.

Probabilistic approaches have also been proposed. In [7], random subsets of the variables in a code are sampled, and the minimum stopping sets in these subsets are found. Even though this is an efficient way to

partition the search space, there is a probability that the minimum stopping set is missed during this procedure. Indeed, the size of the minimum stopping set in a $(1008, 504)$ MacKay code is predicted to be 28, which we prove to be incorrect in Section 7. The authors demonstrate how this error probability can be computed and how it decreases as the maximum number of attempts allowed is increased.

A different probabilistic approach is presented in [8]. This approach iteratively splits a permutation of the parity-check matrix H into two sub-matrices and uses a probabilistic codeword-finding algorithm to find stopping sets in said sub-matrices. This is repeated until a specified maximum iteration bound is reached. The probability of error is improved over the work in [7], and it correctly predicts with high probability that the size of the minimum stopping set in a $(1008, 504)$ MacKay code is 26.

4 Computational Complexity

Assume we are given the Tanner graph $G = (V \cup C, E)$ of some code. While the SS and TS problems are straightforward to define, their apparent simplicity belies significant complexity. In the case of the SS problem, it has been shown that it is **NP-hard** to approximate within a logarithmic factor of the problem input size [12]. An even stronger negative result is obtained when we assume that $\mathbf{NP} \nsubseteq \mathbf{DTIME}(n^{\text{poly}(\log n)})$, where n is the size of the problem's encoding and $k \in \mathbb{N}$. Under this assumption, it was shown in [29] that there is no polynomial-time algorithm to approximate a solution to the SS problem within a factor of $2^{(\log n)^{1-\epsilon}}$ for any $\epsilon > 0$.

Not surprisingly, hardness and inapproximability results have also been established for some trapping set problems. Namely, that you cannot α-approximate a solution to the TS$(a, b)|_a$ and ETS$(a, b)|_b$ problems unless **NP = RP** [12]. It has also been shown that a restricted form of the ETS$(a, b)|_a$ and ETS$(a, b)|_b$ problems, wherein the trapping set solution is required to be leafless, is **NP-hard** [30]. Such leafless trapping sets empirically appear to be the most problematic sub-structures when decoding LDPC codes over the AWGN channel [6], though elementary trapping sets and stopping sets remain the most problematic for the BSC and BEC channels, respectively [2, 4]. However, the computational complexities of the TS$(a, b)|_b$ and ETS$(a, b)|_a$ problems have remained open to date. We prove the hardness of the TS$(a, b)|_b$ and ETS$(a, b)|_a$ problems by reducing the MAX-k-LIN and REGULAR-k-INDEPENDENT-SET problems to the TS$(a, b)|_b$ and ETS$(a, b)|_a$ problems, respectively.

Definition 4 (MAX-k-LIN)**.** *Given a system of linear equations $\mathbf{D} \cdot \mathbf{x} = \mathbf{y}$ over the finite field $\mathbb{Z}_2$ with n variables $x_1, \ldots, x_n \in \{0, 1\}$ and m equations, where each equation has k variables, find a value of $\mathbf{x} = (x_1, \ldots, x_n)$ that maximizes the number of simultaneously satisfied equations.*

Definition 5 (REGULAR-k-INDEPENDENT-SET)**.** *Given an integer k and a δ_V-regular graph $G = (V, E)$, determine whether there is a subset $V^{ind} \subseteq V$ of cardinality $|V^{ind}| = k$ such that no two nodes in V^{ind} share an edge.*

Note that the MAX-k-LIN problem is **NP-hard** for any fixed $k \geq 2$ [31] and the REGULAR-k-INDEPENDENT-SET problem is a special instance of the independent set problem on regular graphs, which is known to be

NP-hard for graphs of degree $\delta_V \geq 3$ [32].

Theorem 1. *The TS$(a, b)|_b$ problem is* **NP-hard**.

Proof. We demonstrate how a polynomial time procedure for solving the TS$(a, b)|_b$ problem can be used to solve MAX-k-LIN for any odd k in polynomial time. Given $\mathbf{D} \in \{0,1\}^{m \times n}$, $\mathbf{y} \in \{0,1\}^m$, and an odd integer $k > 0$, we construct a graph $\mathbf{G}$ such that $\mathbf{G}$ has a trapping that connects to $b = n \cdot (m+1) + p$ vertices an odd number of times if and only if there is an assignment that satisfies exactly p equations in $\mathbf{D} \cdot \mathbf{x} = \mathbf{y}$. Let f_j denote the the j^{th} equation in $\mathbf{D} \cdot \mathbf{x} = \mathbf{y}$. Since we are dealing with the finite field $\mathbb{Z}_2$, addition is modulo 2. Let $N(j)$ be the set of variables in f_j. Note that $|N(j)| = k$. In any equation f_j,

$$\sum_{i=1}^{n} d_{ij} \cdot x_i \equiv \sum_{i \in N(j)} x_i \bmod 2.$$

Clearly, any equation f_j in which the right hand side is 1 is satisfied when an odd number of variables in $N(j)$ are set to 1. For equations f_j in which the right hand side is 0, the fact that $|N(j)| = k$ is odd yields the following:

$$\begin{aligned} \sum_{i=1}^{n} d_{ij} \cdot x_i \equiv 0 \bmod 2 &\iff \sum_{i \in N(j)} x_i \equiv 0 \bmod 2 \\ &\iff \sum_{i \in N(j)} (1 - x_i) \equiv 1 \bmod 2. \end{aligned}$$

Thus, any equation f_j in which the right hand side is 0 can be converted to an equivalent equation f'_j in which the right hand side is 1. Note that the equation f'_j is satisfied when an odd number of terms $(1 - x_i)$ equate to 1. Let $f'_1, \ldots, f'_m$ denote a new system of equations in which the right hand side of every equation is 1. Note that, if the right hand side of f_j is 1, then $f'_j = f_j$. Otherwise, we construct f'_j by transforming f_j as described above. Any equation in the system $f'_1, \ldots, f'_m$ is satisfied if and only if an odd number of terms x_i or $(1 - x_i)$ are 1.

We now demonstrate how to reduce the problem of finding a vector $\mathbf{x}$ that satisfies exactly p equations to an instance of the TS$(a, b)|_b$ problem. We will create a graph $G = (V \cup C, E)$ as follows. For each variable x_i, we create two nodes v_i and v_{i+n} representing the terms x_i and $(1 - x_i)$. For each pair of nodes (v_i, v_{i+n}), we create $(m+1)$ consistency nodes $c_1^{\text{con}_i}, \ldots, c_{m+1}^{\text{con}_i}$ such that v_i and v_{i+n} are connected to each of these nodes. These are used to ensure that two terms x_i and $(1 - x_i)$ are not assigned the same value. For each equation f'_j, we create a node c_j. Let $E = E^{\leq n} \cup E^{>n} \cup E^{\text{con}}$ denote the edge set defined below, where $E^{\leq n}$ and $E^{>n}$ represent the appearance of terms x_i and $(1 - x_i)$, respectively, in each equation f'_j.

$E^{\leq n} = \{(v_i, c_j) | i \leq n \text{ and } x_i \text{ is a term in } f'_j\}$

$E^{>n} = \{(v_i, c_j) | i > n \text{ and } (1 - x_i) \text{ is a term in } f'_j\}$

$E^{\text{con}} = \{(v_i, c_j^{\text{con}_i}), (v_{i+n}, c_j^{\text{con}_i}) | i \leq n, j \leq m+1\}$

Let C^{con} denote the set of all consistency nodes. We now have our graph $G = (V \cup C, E)$, where $V = \{v_1, \ldots, v_{2 \cdot n}\}$ and $C = \{c_1, \ldots, c_m\} \cup C^{\text{con}}$. Let $b = n \cdot (m+1) + p$. Then this instance of the TS$(a, b)|_b$ problem has a feasible solution if and only if p equations from the system $f'_1, \ldots, f'_m$ are satisfied.

First assume that $\text{TS}(a,b)|_{b=n\cdot(m+1)+p}$ yields a trapping set T. In order to ensure a consistent assignment of terms, it is impossible for nodes v_i and v_{i+n} to be in the trapping set T, as this would yield a value of $b < n \cdot (m+1)$ because nodes $c_1^{\text{con}_i}, \ldots, c_{m+1}^{\text{con}_i}$ would be connected an even number of times to T via nodes v_i and v_{i+n}. This, in turn, means that the terms x_i and $(1-x_i)$ can never have the same value, thus yielding a consistent assignment. Clearly, any such assignment will yield a value of $b \geq n\cdot(m+1)$, with $b = n\cdot(m+1)+p$ indicating that p of the nodes in $C \setminus C^{\text{con}}$ are connected to T an odd number of times. Therefore, we can construct a solution that satisfies exactly p equations from T as follows. If $v_i \in T (i \leq n)$, then let $x_i = 1$. Similarly, if $v_i \in T (i > n)$, then let $x_i = 0$. Repeating this procedure for an instance of the $\text{TS}(a,b)|_b$ problem with $n \cdot (m+1) + 1 \leq b \leq n \cdot (m+1) + m$ yields a solution to MAX-k-LIN for odd k.

Now, assume that we have a vector $\mathbf{x}$ which satisfies p equations from the system $f'_1, \ldots, f'_m$. We can construct a trapping set solution T to $\text{TS}(a,b)|_{b=n\cdot(m+1)+p}$ as follows. For every $x_i = 1$, choose v_i to be in T. Similarly, for every $x_i = 0$, choose v_{i+n} to be in T. Since $\mathbf{x}$ is an n-dimensional vector and every v_i is connected to $(m+1)$ consistency nodes, we have have $b \geq n \cdot (m+1)$ nodes connected exactly once to T. Since each of the p nodes corresponding to the satisfied equations is connected to an odd number of nodes in T, we have $b = n \cdot (m+1) + p$. □

Note that the $\text{TS}(a,b)|_b$ problem is an optimization problem. We can consider the corresponding decision problem. That is, given both a and b, does the graph $\mathbf{G}$ have a trapping set T, such that $|T| < a$ and exactly b vertices are connected to T an odd number of times. Since a trapping set can be verified in polynomial time, this decision version of the $\text{TS}(a,b)|_b$ problem is **NP-complete**.

We elucidate the preceding proof with a simple example. Suppose we are given the set of linear equations $\mathbf{D} \cdot \mathbf{x} = \mathbf{y}$ over $\mathbb{Z}_2$, where $\mathbf{x} = (x_1, x_2, x_3, x_4)$, $\mathbf{y} = (1,0,1)^T$, and $\mathbf{D}$ is defined below.

$$\mathbf{D} = \begin{pmatrix} 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \end{pmatrix}$$

We transform $\mathbf{D} \cdot \mathbf{x} = \mathbf{y}$ into the following equivalent set of equations:

$f'_1 \quad : \quad x_1 + x_2 + x_3 \equiv 1 \bmod 2$

$f'_2 \quad : \quad (1-x_1) + (1-x_3) + (1-x_4) \equiv 1 \bmod 2$

$f'_3 \quad : \quad x_2 + x_3 + x_4 \equiv 1 \bmod 2$

The graph construction $G = (V \cup C, E)$ corresponding to this system of equations can be seen in Figure 3. It is easy to see that if there is a solution for $b = 17, 18, 19$, then $p = 1, 2, 3$ equations can be satisfied, respectively.

Corollary 1. *The $\text{TS}(a,b)|_b$ problem is* **para-NP-hard** *when parameterized by the maximum degree of the given Tanner graph.*

Proof. This follows from the reduction in Theorem 1, which results in a Tanner graph of degree at most 5 for the MAX-3-LIN problem. See Figure 3 for an example.

Since the MAX-3-LIN problem is **NP-hard** [31], the $\text{TS}(a,b)|_b$ problem is **NP-hard** on Tanner graphs with degree at most 5. Since the $\text{TS}(a,b)|_b$ problem is **NP-hard** even for Tanner graphs with fixed maximum degree,

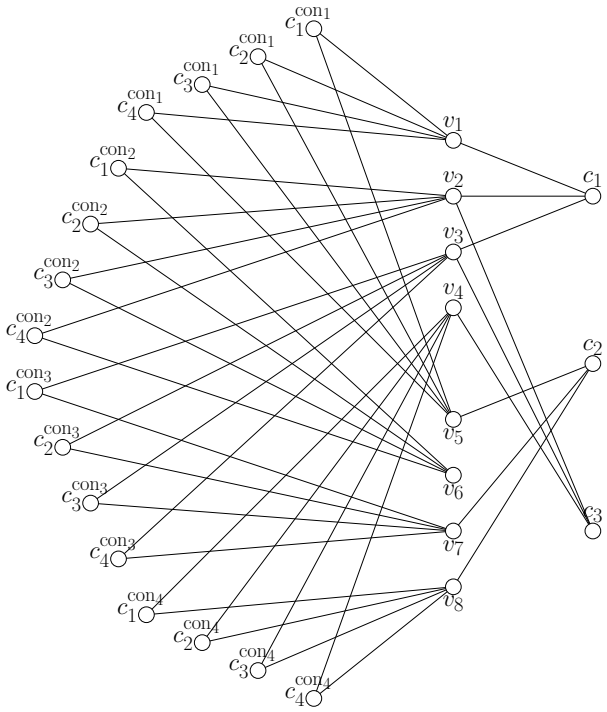

Figure 3: Graphical representation of $\mathbf{D} \cdot \mathbf{x} = \mathbf{y}$. Let $17 \leq b \leq 19$. The trapping set T of size $a = 4$ obtained from solving the TS$(a, b)|_b$ problem given the largest value of b will yield a solution this MAX-3-LIN instance.

this problem is **para-NP-hard** when parameterized by the maximum degree of the given Tanner graph. □

Theorem 2. *The ETS$(a, b)|_a$ problem is* **NP-hard**.

Proof. We demonstrate how a polynomial time procedure for solving the ETS$(a, b)|_a$ problem can be used to solve REGULAR-k-INDEPENDENT-SET in polynomial time. For simplicity, we will consider regular graphs of degree $\delta_{\mathscr{V}} = 3$, called cubic graphs, for the rest of this proof. An example of how the construction proposed herein can be used on a 4-regular graph can be seen in Figure 5. Note that REGULAR-3-INDEPENDENT-SET is **NP-complete** [32].

Given a 3-regular graph $\mathscr{G} = (\mathscr{V}, \mathscr{E})$, we construct a bipartite graph $G = (V \cup C, E)$ as follows. For every $v_i \in \mathscr{V}$, we create nodes $v_i, \hat{c}_i$ and for every $e_j \in \mathscr{E}$, we construct nodes $\hat{v}_j, c_j$. The edge set is defined by $E = E^{\mathscr{G}} \cup \hat{E}$, where $E^{\mathscr{G}} = \{(v_i, c_j) | v_i \in e_j \in \mathscr{E}\}$ and $\hat{E} = \{(v_i, \hat{c}_i)\}_{i=1}^{|\mathscr{V}|} \cup \{(\hat{v}_i, c_i)\}_{i=1}^{|\mathscr{E}|}$. See Figures 4 and 5 for examples.

The arguments of the proof follow from the following simple observation. Let $c_{i_1}, c_{i_2}, c_{i_3}, \hat{c}_i$ denote the neighbors of v_i in the reduced graph (see Figure 4). Without loss of generality, note that a minimum-cost selection of four nodes to be in the trapping set T is to choose v_i and $\hat{v}_{i_1}, \hat{v}_{i_2}, \hat{v}_{i_3}$, since this yields a cost of 1. That is, there is only one neighbor connected exactly once to T in the subgraph induced by these four nodes. This neighbor is $\hat{c}_i$. Any other selection of four nodes would yield a cost greater than 1. Note that this optimal selection of four nodes corresponds to a node $v_i \in \mathscr{V}$ and the three edges incident on it in the original graph $\mathscr{G} = (\mathscr{V}, \mathscr{E})$. The intuition behind this proof is that, if there is an independent set of size k in $\mathscr{G}$, then we can simply choose k such structures of cost 1 consisting of four nodes each. This yields a trapping set solution of cost k given $a = 4 \cdot k$.

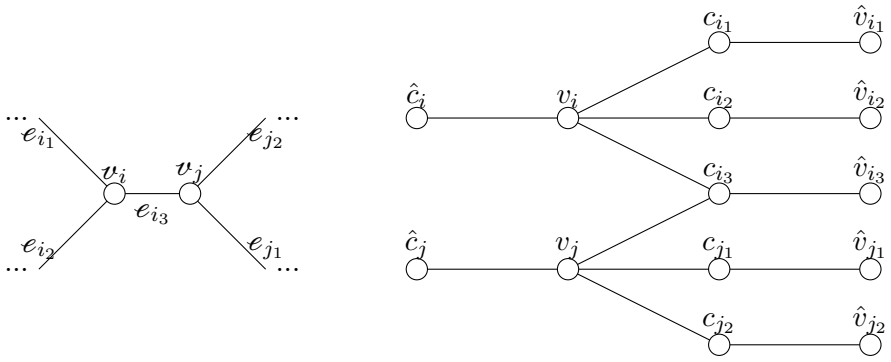

Figure 4: Two nodes v_i, v_j and the edges incident on them in a given 3-regular graph $\mathscr{G} = (\mathscr{V}, \mathscr{E})$ (*left*). Following the construction in Theorem 2, the reduced graph $G = (V \cup C, E)$ will contain the structure on the right.

We will now prove the following statement. Given a cubic graph $\mathscr{G} = (\mathscr{V}, \mathscr{E})$ and its reduced graph $G = (V \cup C, E)$, $\text{ETS}(a, b)|_{a=4\cdot k}$ yields a solution of cost $b = k$ in G if and only if there is an independent of size k in $\mathscr{G}$.

First, assume that $\mathscr{G} = (\mathscr{V}, \mathscr{E})$ does not contain an independent set of size k. We will show that $\text{ETS}(a, b)|_{a=4\cdot k}$ does not have a solution of cost $b = k$. Note that the subgraph $\mathscr{G}_{\mathscr{V}'}$ induced by any set of k nodes $\mathscr{V}' = \{v_{i_1}, \ldots, v_{i_k}\}$ will have at least one edge of the form $(v_{i_j}, v_{i_l}), 1 \leq j, l \leq k$. This follows from the fact that there is no independent set of size k. Therefore, any subgraph $G_{4\cdot k}$ induced by a minimum-cost trapping set T of $4 \cdot k$ nodes will have some check node connected to two nodes $v_i, v_j \in V \setminus \{\hat{v}_1, \ldots, \hat{v}_{|\mathscr{E}|}\}$ (in the original graph $\mathscr{G}$, these nodes correspond to nodes v_i, v_j that are adjacent and can thus never be in an independent set together). Let Δ denote the set of all such check nodes. Using the graph $G = (V \cup C, E)$ in Figure 4 as a reference, note that any optimal selection of 5 or more nodes in V must include v_i, v_j, which leads to $\Delta = \{c_{j_1}\}$.

Consider the best selection of nodes so as to minimize b. It follows that any trapping set $T \subseteq V$ solution of $\text{ETS}(a, b)|_{a=4\cdot k}$ will satisfy inequality in System (1).

$$
\begin{aligned}
b &\geq 2 \cdot |\Delta| + \left\lceil \frac{a - 2 \cdot |\Delta| - 2 \cdot (\delta_{\mathscr{V}} - 1) \cdot |\Delta|}{\delta_{\mathscr{V}} + 1} \right\rceil \\
&= \left\lceil \frac{a - 2 \cdot |\Delta|}{4} \right\rceil + |\Delta| \\
&= k + |\Delta| + \left\lceil -\frac{|\Delta|}{2} \right\rceil \\
&= k + \left\lceil \frac{|\Delta|}{2} \right\rceil \\
&> k
\end{aligned}
\tag{1}
$$

Since there is no independent set of size k in $\mathscr{G}$, we are guaranteed $|\Delta| \geq 1$. Using Figure 4 as a reference, note that for every node in Δ there is a cost of 2 associated with choosing v_i and v_j to be in the trapping set T. This is due to the nodes $\hat{c}_i$ and $\hat{c}_j$. We account for this using the term $2 \cdot |\Delta|$ in System (1). This leaves

$(a - 2 \cdot |\Delta|)$ possible choices for nodes to be in the trapping set. Choosing $\hat{v}_{i_1}$, $\hat{v}_{i_2}$, $\hat{v}_{j_1}$, and $\hat{v}_{j_2}$ does not incur a cost because c_{i_1}, c_{i_2}, c_{j_1}, and c_{j_2} would be connected twice to T via the pairs of nodes $(v_i, \hat{v}_{i_1})$, $(v_i, \hat{v}_{i_2})$, $(v_j, \hat{v}_{j_1})$, and $(v_j, \hat{v}_{j_2})$, respectively. There are $2 \cdot (\delta_{\mathscr{V}} - 1)$ such zero-cost choices for every node in Δ. Thus, from a we subtract the number $2 \cdot (\delta_{\mathscr{V}} - 1) \cdot |\Delta|$ of nodes whose inclusion in T yields an optimal solution. The remaining choices will have an optimal cost of 1 for every substructure of $(\delta_{\mathscr{V}} + 1)$ nodes as we mentioned earlier.

Now, assume that $\mathscr{G}$ does have an independent set $\mathscr{v}_{i_1}, \ldots, \mathscr{v}_{i_k}$ of size k. Then there exists a trapping set T of cost k structured as follows. Let $V^{\text{ind}} = \{v_{i_1}, \ldots, v_{i_k}\} \subset T$. For each $v_{i_j} \in V^{\text{ind}}$, add all three auxiliary nodes corresponding to edges of $\mathscr{v}_{i_j}$ in $\mathscr{G}$. That is, for every $v_{i_j} \in V^{\text{ind}}$, add the nodes in the set $\{\hat{v}_p | \mathscr{v}_{i_j} \in \mathscr{e}_p \in \mathscr{E}\}$ to T. It follows that the number of neighbors of T connected exactly once to T is k. This cost comes from the nodes $\hat{c}_{i_1}, \ldots, \hat{c}_{i_k}$. Thus, $\text{ETS}(a, b)|_{a=4 \cdot k}$ yields a solution of cost $b = k$. This result holds for regular graphs of arbitrary degree. Indeed, given a $\delta_{\mathscr{V}}$-regular graph $\mathscr{G} = (\mathscr{V}, \mathscr{E})$ and its reduced graph $G = (V \cup C, E)$, $\text{ETS}(a, b)|_{a=(\delta_{\mathscr{V}}+1) \cdot k}$ yields a solution of cost $b = k$ in G if and only if there is an independent of size k in $\mathscr{G}$. See Figure 5 for an example given a 4-regular graph $\mathscr{G} = (\mathscr{V}, \mathscr{E})$. □

Note that the $\text{ETS}(a, b)|_a$ problem is an optimization problem. We can consider the corresponding decision problem. That is, given both a and b, does the graph $\mathbf{G}$ have an elementary trapping set T, such that $|T| = a$ and at least b vertices are connected to T exactly once. Since an elementary trapping set can be verified in polynomial time, this decision version of the $\text{ETS}(a, b)|_a$ problem is **NP-complete**.

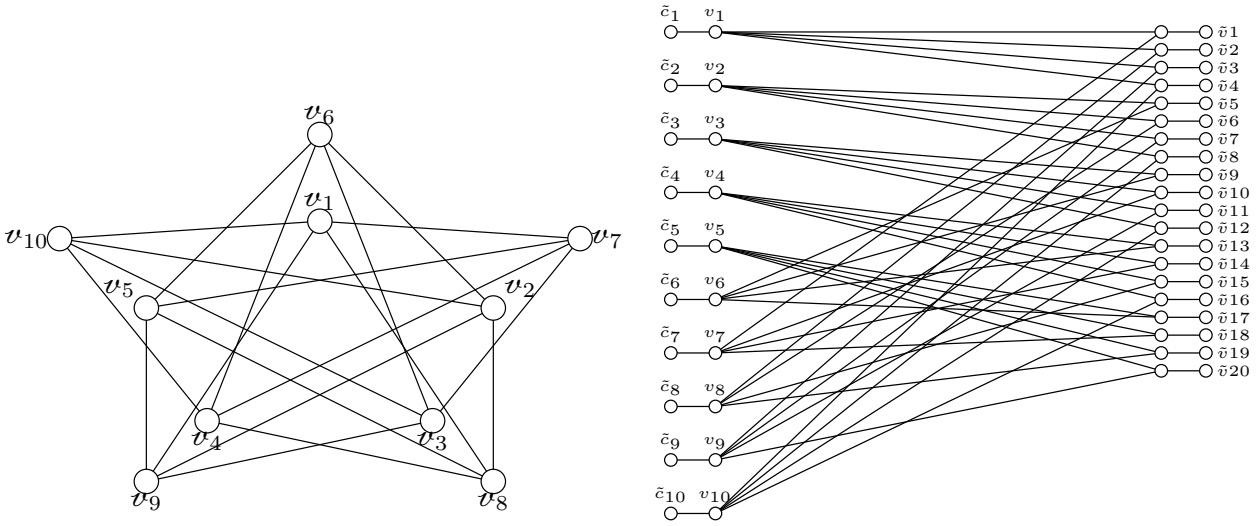

Figure 5: Given the 4-regular graph $\mathscr{G} = (\mathscr{V}, \mathscr{E})$ (*top*), the construction in Theorem 2 yields the graph $G = (V \cup C, E)$ (*bottom*). Note that $\mathscr{v}_1, \ldots, \mathscr{v}_5$ is an independent set of size 5 in $\mathscr{G}$. A minimum-cost trapping set $T = \{v_1, v_2, v_3, v_4, v_5, \hat{v}_1, \ldots, \hat{v}_{20}\}$ yields a solution of cost $b = 5$ to the $\text{ETS}(a, b)|_{a=(4+1)5=25}$ problem. This cost comes from the nodes $\hat{c}_1, \ldots, \hat{c}_5$.

Corollary 2. *The $\text{ETS}(a, b)|_a$ problem is* **para-NP-hard** *when parameterized by the maximum degree of the given Tanner graph.*

Proof. This follows from the reduction in Theorem 2, which results a in Tanner graph of degree at most 5 for the REGULAR-4-INDEPENDENT-SET problem. See Figure 5 for an example.

Since the REGULAR-4-INDEPENDENT-SET problem is **NP-hard** [32], the ETS$(a,b)|_a$ problem is **NP-hard** on Tanner graphs with degree at most 5. Since the ETS$(a,b)|_a$ problem is **NP-hard** even for Tanner graphs with fixed maximum degree, this problem is **para-NP-hard** when parameterized by the maximum degree of the given Tanner graph. □

Theorems 1 and 2 round out the complexity results previously established in the literature. Thus, the SS, TS$(a,b)|_a$, TS$(a,b)|_b$, ETS$(a,b)|_a$, and ETS$(a,b)|_b$ problems are all **NP-hard**. This raises important questions about the feasibility of solving problem instances of interest. Fortunately, the use of novel branch-and-bound algorithms has yielded some success in this area.

5 Parameterization

In this section, we examine the ETS$(a,b)|_a$ problem from the perspective of parameterized algorithms. First, we establish that the ETS$(a,b)|_a$ problem is unlikely to be Fixed-Parameter Tractable (FPT) when parameterized by a. We do this by showing that the problem is **W[1]-hard** when parameterized by a.

We will show this by a parameterized reduction from the problem of determining if a regular graph has a clique of size k. This problem is known to be **W[1]-complete** when parameterized by k [33]. From [33], we have the following definition of a parameterized reduction:

Definition 6. *Let $A, B \in \Sigma^* \times \mathbb{N}$ be two parameterized problems. A* parameterized reduction *from A to B is an algorithm that, given an instance $(\mathbf{x}, k)$ of A, outputs an instance $(\mathbf{x}', k')$ of B such that:*

1. *$(\mathbf{x}, k)$ is a yes-instance of A, if and only if $(\mathbf{x}', k')$ is a yes-instance of B.*
2. *$k' \leq g(k)$ for some computable function g.*
3. *The running time is $f(k) \cdot |x|^{O(1)}$ for some computable function f.*

Note that a polynomial time many-to-one reduction is insufficient to show **W[1]-completeness**. For example, there is a polynomial time many-to-one reduction from independent set to vertex cover. However, in a graph $\mathbf{G}$ with n vertices, an independent set of size k corresponds to a vertex cover of size $(n-k)$. Since n does not depend on k, there is no computable function g such that $n - k \leq g(k)$. In fact, vertex cover is in **FPT** when parameterized by the size of the cover while independent set is **W[1]-complete** when parameterized by the size of the independent set [33].

Let $\mathscr{G} = (\mathscr{V}, \mathscr{E})$ be an undirected graph where each node has degree r. From $\mathscr{G}$ we construct the bipartite graph $G = (V \cup C, E)$ as follows:

1. For each node $v_i \in \mathscr{V}$, add the node v_i to V.
2. For each edge $e_l = (v_i, v_j) \in \mathscr{E}$ add the node c_l to C. Additionally add the edges (v_i, c_l) and (v_j, c_l) to E.

Figure 6 shows the original regular graph $\mathscr{G}$ and the corresponding bipartite graph G constructed as in the reduction.

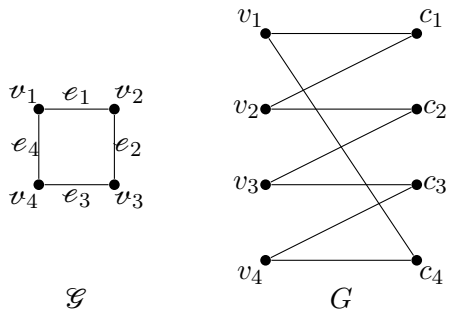

Figure 6: A regular graph $\mathscr{G}$ and the bipartite graph G constructed from it.

We will now show that $\mathscr{G}$ has a clique of size k, if and only if $\text{ETS}(a,b)|_{a=k}$ yields a solution of cost $b = r \cdot k - 2 \cdot \binom{k}{2}$ in G.

Lemma 1. *Let $\mathscr{G} = (\mathscr{V}, \mathscr{E})$ be an undirected graph where each node has degree r and let $G = (V \cup C, E)$ be the bipartite graph constructed from $\mathscr{G}$. $\mathscr{G}$ has a clique of size k, if and only if ETS$(a,b)|_{a=k}$ yields a solution of cost $b = r \cdot k - 2 \cdot \binom{k}{2}$ in G.*

Proof. First, we establish that any elementary trapping set T of G with $|T| = k$ has cost $b \geq r \cdot k - 2 \cdot \binom{k}{2}$.

Let $T \subseteq V$ be a set of nodes such that $|T| = k$. By construction of G, every node in C has degree 2. Thus, T is an elementary trapping set of G. Let $S_T \subseteq C$ be the set of nodes adjacent to exactly one node in T. By definition, T has cost $b = |S_T|$.

Consider a node $v_i \in T$. Let S_i be the set of nodes in S_T adjacent to v_i. Since, S_T is the set of nodes adjacent to exactly one node in T, we have that for any pair of nodes $v_i, v_j \in T$, $S_i \cap S_j = \emptyset$. Thus, $|S_T| = \sum_{v_i \in T} |S_i|$.

Consider an edge $\mathscr{e}_l$ in $\mathscr{G}$. If $\mathscr{e}_l$ does not use $\mathscr{v}_i$ as an endpoint, then by construction of G, the node c_l is not adjacent to v_i. Thus, $c_l \notin S_i$. If $\mathscr{e}_l$ uses $\mathscr{v}_i$ as an endpoint, then let $\mathscr{v}_j$ be the other endpoint of $\mathscr{e}_l$.

If $v_j \in T$, then c_l is adjacent to two nodes in T, namely v_i and v_j. Thus, $c_j \notin S_i$. If $v_j \notin T$, then c_l is adjacent to exactly one node in T. Thus, $c_l \in S_i$.

Thus, a node $c_l \in C$ is in S_i, if and only if the edge $\mathscr{e}_l$ is between the nodes $\mathscr{v}_i$ and $\mathscr{v}_j$ in $\mathscr{G}$ such that $v_j \notin T$. Since $\mathscr{v}_i$ is adjacent to r nodes in $\mathscr{G}$ and $|T| = k$, there are at least $(r - (k-1))$ such nodes. Thus, $|S_i| \geq (r - (k-1))$.

This means that

$$|S_T| = \sum_{v_i \in T} |S_i| \geq \sum_{v_i \in T} (r - (k-1)) = k \cdot (r - (k-1)) = r \cdot k - 2 \cdot \binom{k}{2}.$$

Consequently, any elementary trapping set T of G with $|T| = k$ has cost $b \geq r \cdot k - 2 \cdot \binom{k}{2}$.

We now establish that G has an elementary trapping set $T \subseteq V$ of size k with cost $b = r \cdot k - 2 \cdot \binom{k}{2}$, if and only if $\mathscr{G}$ has a clique of size k.

First, assume that $\mathscr{G}$ has a clique Q of size k. Let $T = \{v_i | \mathscr{v}_i \in Q\}$. As observed previously, T is an elementary trapping set of G. Consider a node $v_i \in T$. Note that $\mathscr{v}_i$ is adjacent to $(r - (k-1))$ nodes in $\mathscr{V} \setminus Q$. Thus, by the arguments used previously, $|S_i| = (r - (k-1))$. This means that

$$|S_T| = \sum_{v_i \in T} |S_i| = \sum_{v_i \in T} (r - (k-1)) = k \cdot (r - (k-1)) = r \cdot k - 2 \cdot \binom{k}{2}.$$

Consequently, T is an elementary trapping set of G with cost $b = k - 2 \cdot \binom{k}{2}$ as desired.

Now assume that G has an elementary trapping set $T \subseteq V$ of size k with cost $b = r \cdot k - 2 \cdot \binom{k}{2}$. Let $Q = \{\mathscr{v}_i | v_i \in T\}$. Consider a node v_i in T. From previous observations, we know that $|S_i| = (r - (k-1))$. Thus, by the arguments used previously, $\mathscr{v}_i$ is adjacent to $(k-1)$ nodes in Q. Since $|Q| = k$, this means that each node in Q is adjacent to every other node in Q. Thus, Q is a clique of size k as desired. □

From Lemma 1, we get the following result.

Theorem 3. *The ETS$(a, b)|_a$ problem is* **W[1]-hard** *when parameterized by a.*

Proof. Let $\mathscr{G} = (\mathscr{V}, \mathscr{E})$ be an undirected graph where each node has degree r and let $G = (V \cup C, E)$ be the bipartite graph constructed from $\mathscr{G}$. From Lemma 1, $\mathscr{G}$ has a clique of size k, if and only if ETS$(a, b)|_{a=k}$ yields a solution of cost $b = r \cdot k - 2 \cdot \binom{k}{2}$ in G.

The problem of determining if a regular graph has a clique of size k is **W[1]-complete** when parameterized by k [33]. Thus, the ETS$(a, b)|_a$ problem is **W[1]-hard** when parameterized by a. □

Even though the ETS$(a, b)|_a$ problem is **W[1]-hard**, we can still develop a parameterized algorithm for this problem. However, this algorithm is not an FPT algorithm.

Let $G = (V \cup C, E)$ be a bipartite graph and let T be a subset of V. For each node $c_j \in C$, we can count the number of nodes in T adjacent to c_j in polynomial time. Thus, we can check if T is an elementary trapping set in polynomial time. Additionally, we can determine the cost b of T in polynomial time.

Thus, we have the following algorithm for solving the ETS$(a, b)|_a$ problem.

Algorithm 1 Algorithm for the ETS$(a, b)|_a$ problem.

Input: Bipartite graph $G = (V \cup C, E)$ and integer k
Output: Minimum cost b of an elementary trapping set of G of size k.

```
1: procedure ETS-PARAM(G, k)
2:     b := |C| + 1.
3:     for (each T ⊆ V such that |T| = k) do
4:         if (T is an elementary trapping set of G) then
5:             Let b' be the cost of T.
6:             if (b' < b) then
7:                 b := b'.
8:     return b.
```

Since Algorithm 1 checks every subset of V of size k, it will find the cost of every elementary trapping set of G of size k. Thus, Algorithm 1 solves the ETS$(a, b)|_a$ problem.

For each subset T of V such that $|T| = k$, Algorithm 1 performs a pair of polynomial time checks to see if T is an elementary trapping set of G and to find the cost of T. There are at most $|V|^k$ subsets of V of size k. Additionally, checking if each subset T of V is an elementary trapping set can be done in polynomial time. Thus, Algorithm 1 runs in time $O^*(|V|^k)$.

Alternatively, there are at most $2^{|V|}$ subsets of V. Thus, Algorithm 1 also runs in time $O^*(2^{|V|})$.

6 ILP Formulations

We propose integer linear programs (ILPs) for the SS, TS$(a,b)|_a$, TS$(a,b)|_b$, ETS$(a,b)|_a$, and ETS$(a,b)|_b$ problems given the Tanner graph representation $G = (V \cup C, E)$ of a parity-check matrix $H \in \{0,1\}^{m \times n}$.

6.1 Stopping Sets

We want to find the stopping set $S \subseteq V$ of minimum cardinality. To do so, we define the following integer linear program with n variables and $(|E|+1)$ constraints, where $x_i \in \{0,1\}$ is 1 if $v_i \in V$ is in the minimum stopping set.

Integer Linear Program for the SS problem:

$$\min \sum_{v_i \in V} x_i \text{ such that}$$

$$\left(\sum_{(v_i,c_j),(c_j,v_k) \in E} x_k \right) - 2 \cdot x_i \geq 0, \forall (v_i, c_j) \in E$$

$$\sum_{v_i \in V} x_i \geq 2$$

$$x_i \in \{0,1\}, \forall i \in [n]$$

The first constraint looks at all paths $\{(v_i, c_j), (c_j, v_k)\}$ of length 2 from a node $v_i \in S$ in the minimum stopping set. If the destination node v_k is in the stopping set S, then its indicator variable x_k is 1. Adding over all the indicator variables corresponding to the destination nodes of all paths of length 2 gives us the number of variable nodes connected back to S. In order to check if this value is at least 2, we subtract from it the factor $2 \cdot x_i$. Two cases arise. First, if x_i is 1, then v_i is in the stopping set S and all of its neighbors must be connected back to S at least twice in order to satisfy the inequality. Second, if x_i is 0, then v_i is not in S, and the inequality is trivially satisfied. Thus, every neighbor of a node in S is connected to S at least twice. This is precisely the connectivity condition in Definition 2.

The second constraint enforces a minimum stopping set size of 2. This eliminates the possibility of a stopping set of size 0, which would be erroneous. It is worth noting that we can enforce a minimum stopping set size of any arbitrary value. For example, we set this lower bound to 25 and 37 in order to find the stopping sets of sizes 36 and 48 in Figures 10 and 11, respectively (See https://www.cs.ucf.edu/~velasquez/

StoppingSets/). While we are generally interested in stopping sets of minimum size, it is useful to be able to set lower bounds on the size of these sets.

6.2 Elementary Trapping Sets

For the elementary (a, b)-trapping set variants, another set of binary variables is introduced, where $y_j \in \{0, 1\}$ is 1 if $c_j \in C$ is a neighbor of some node in the trapping set T. The two ILPs below have $(n + m)$ variables and $(|E| + 2 \cdot m + 1)$ constraints. They encode the elementary (a, b)-trapping set problems ETS$(a, b)|_b$ and ETS$(a, b)|_a$.

Integer Linear Program for the ETS$(a, b)|_b$ problem:

$$\begin{aligned}
\min \sum_{v_i \in V} x_i \text{ such that} \\
& x_i - y_j \leq 0, \forall (v_i, c_j) \in E \\
& y_j - \sum_{(v_i, c_j) \in E} x_i \leq 0, \forall j \in [m] \\
& \sum_{(v_i, c_j) \in E} x_i \leq 2, \forall j \in [m] \\
& \sum_{c_j \in C} \left(2 \cdot y_j - \sum_{(v_i, c_j) \in E} x_i \right) = b \\
& x_i, y_j \in \{0, 1\}, \forall i \in [n], j \in [m]
\end{aligned}$$

The first constraint $x_i - y_j \leq 0$ ensures that, if v_i and c_j are neighbors and v_i is in T, then c_j is understood to be a neighbor of T. These $\mathbf{y}$ variables allow us to count the number of neighbors of T that satisfy more complicated properties than those required by stopping sets. For example, we need to count the number of neighbors of T connected to T exactly once or twice. The second constraint satisfies the proposition that, if c_j is a neighbor of the trapping set T, then there must be some neighbor of c_j which is in T. The third constraint ensures that all neighbors of T are connected to T at most twice. As such, we look over all check nodes c_j to make sure that the number of neighboring variable nodes v_i in T is at most 2.

Recall that we want an elementary trapping set with b neighbors of T connected to T exactly once. The fourth constraint achieves this as follows. Let us consider an arbitrary term $(2 \cdot y_j - \sum_{(v_i, c_j) \in E} x_i)$ in the outer sum. Two cases arise. First, if $y_j = 0$, then $x_i = 0$ for any node v_i connected to c_j. This follows from the first constraint, and the term in question will thus not contribute any value to the outer sum. Second, consider the case where y_j is 1. Note that the inner sum $\sum_{(v_i, c_j) \in E} x_i$ is guaranteed to be at least 1 (second constraint) and at most 2 (third constraint). Whenever this sum is 2, the term in question does not contribute any value to the outer sum. However, if the sum is 1, then this term contributes a value of 1 to the outer sum. Thus, the fourth constraint counts the number of neighbors of the trapping set T that are connected to T exactly once and enforces this number to be equal to b.

Integer Linear Program for the ETS$(a, b)|_a$ problem:

$$\min \sum_{c_j \in C} \left(2 \cdot y_j - \sum_{(v_i, c_j) \in E} x_i \right) \text{ such that}$$

$$x_i - y_j \leq 0, \forall (v_i, c_j) \in E$$

$$y_j - \sum_{(v_i, c_j) \in E} x_i \leq 0, \forall j \in [m]$$

$$\sum_{(v_i, c_j) \in E} x_i \leq 2, \forall j \in [m]$$

$$\sum_{v_i \in V} x_i = a$$

$$x_i, y_j \in \{0, 1\}, \forall i \in [n], j \in [m]$$

This ILP is largely similar to the previous one. However, the objective function which we are trying to minimize is the same as the fourth constraint in the previous ILP. Namely, we want to minimize the value of b given some value of a. As such, the fourth constraint in this ILP enforces the size of the trapping set T to be equal to a.

6.3 Trapping Sets

For the (a, b)-trapping set variants, we need to count the neighbors of T that are connected to T an odd number of times. For this, we introduce the parameter δ_{max} and new variables α_j and $\mathscr{I}_j^{\text{odd}}$, where δ_{max} denotes the largest degree of any check node in C. The indicator variable $\mathscr{I}_j^{\text{odd}}$ will be 1 if check node c_j is connected to T an odd number of times and 0 otherwise. The auxiliary variable α_j will be used to ensure that $\mathscr{I}_j^{\text{odd}}$ is assigned the correct value. The following ILPs have $(n + 2 \cdot m)$ variables and $(m + 1)$ constraints. They encode the TS$(a, b)|_b$ and TS$(a, b)|_a$ problems.

Integer Linear Program for the TS$(a, b)|_b$ problem:

$$\min \sum_{v_i \in V} x_i \text{ such that}$$

$$2 \cdot \alpha_j + \mathscr{I}_j^{\text{odd}} - \sum_{(v_i, c_j) \in E} x_i = 0, \forall j \in [m]$$

$$\sum_{c_j \in C} \mathscr{I}_j^{\text{odd}} = b$$

$$x_i, \mathscr{I}_j^{\text{odd}} \in \{0, 1\}, \alpha_k \in \{0, \ldots, \left\lfloor \frac{\delta_{\text{max}}}{2} \right\rfloor\}, \forall i \in [n], j, k \in [m]$$

The first constraint ensures that $\mathscr{I}_j^{\text{odd}}$ will be 1 if check node c_j is connected to T an odd number of times and 0 otherwise. Note that $2 \cdot \alpha_j$ can take any even value from 0 to δ_{max}. It follows that, if $\sum_{(v_i, c_j) \in E} x_i$ is odd, then the only way to satisfy the constraint is if $\mathscr{I}_j^{\text{odd}} = 1$. Similarly, if $\sum_{(v_i, c_j) \in E} x_i$ is even, then we have $\mathscr{I}_j^{\text{odd}} = 0$.

This allows us to count odd-degree nodes. The second constraint adds over all of these indicator variables and enforces this value to be equal to b.

Integer Linear Program for the TS$(a,b)|_a$ problem:

$$\min \sum_{c_j \in C} \mathscr{I}_j^{\text{odd}} \text{ such that}$$

$$2 \cdot \alpha_j + \mathscr{I}_j^{\text{odd}} - \sum_{(v_i,c_j)\in E} x_i = 0, \forall j \in [m]$$

$$\sum_{v_i \in V} x_i = a$$

$$x_i, \mathscr{I}_j^{\text{odd}} \in \{0,1\}, \alpha_k \in \{0,\ldots,\left\lfloor \frac{\delta_{\max}}{2} \right\rfloor\}, \forall i \in [n], j,k \in [m]$$

This ILP is similar to the previous one. However, we are now minimizing the number of nodes connected to the trapping set T an odd number of times. The second constraint ensures that the size of T is equal to a.

7 Experimental Results

We test our ILPs on codes taken from Mackay's encyclopedia of codes [15]. The experiments were performed using the Gurobi platform version 7.5.1 [34] on a Ubuntu 14.04 server with 64 GB memory and 64 Intel(R) Celeron(R) M processors running at 1.50 GHz. One of our main results is finding a stopping set of size 48 in the $(4896, 2474)$ Margulis Code (see Figure 11 in https://www.cs.ucf.edu/~velasquez/StoppingSets/) in 700451 seconds. This compares favorably to the method proposed in [9], which searches for stopping sets of size up to 26.

Minimum Stopping Set Size

Code	[7]	[8]	[9]	Us
$(504, 252)$ Mackay	16 (N/A)	16 (600 hours)	16 (N/A)	16 (37 seconds)
$(504, 252)$ PEG	N/A (N/A)	19 (25 hours)	19 (N/A)	19 (365 seconds)
$(1008, 504)$ Mackay	28 (N/A)	26 (3085 hours)	N/A (N/A)	26 (18.73 hours)
$(4896, 2474)$ Margulis	24 (N/A)	24 (162 hours)	24 (N/A)	24 (267 seconds)

Table 1: The size of minimum stopping sets in 4 popular codes are presented based on the results of various methods in the literature. The numbers in parentheses denote the execution time to find said sets.

In Table 1, we demonstrate that the size of the minimum stopping set in various codes is found by our approach to be the same as that reported in [8] and [9]. Our results disagree with those presented in [7] for the $(1008, 504)$ Mackay code. Our approach determined that the minimum size of a stopping set in this code is 26, while the method in [7] predicted a size of 28. However, see Figure 8 in https://www.cs.ucf.edu/~velasquez/StoppingSets/ for the minimum-size stopping set of size 26 found by our approach. We have also provided the execution time expended to find these sets. It is worth noting that our method is significantly faster than the

one proposed in [8].

- $(504, 252)$ Mackay Code: Minimum stopping set of size 16 was found in 37 seconds (See Figure 7). The approach in [8] took 600 hours.
- $(504, 252)$ PEG Code: Minimum stopping set of size 19 was found in 365 seconds (See Figure 8). The approach in [8] took 25 hours.
- $(1008, 504)$ Mackay Code: Minimum stopping set of size 26 was found in 67440 seconds, or 18.73 hours (See Figure 9). The approach in [8] took 3085 hours.
- $(4896, 2474)$ Margulis Code: Minimum stopping set of size 24 was found in 267 seconds (see Figure 10). The approach in [8] took 162 hours.

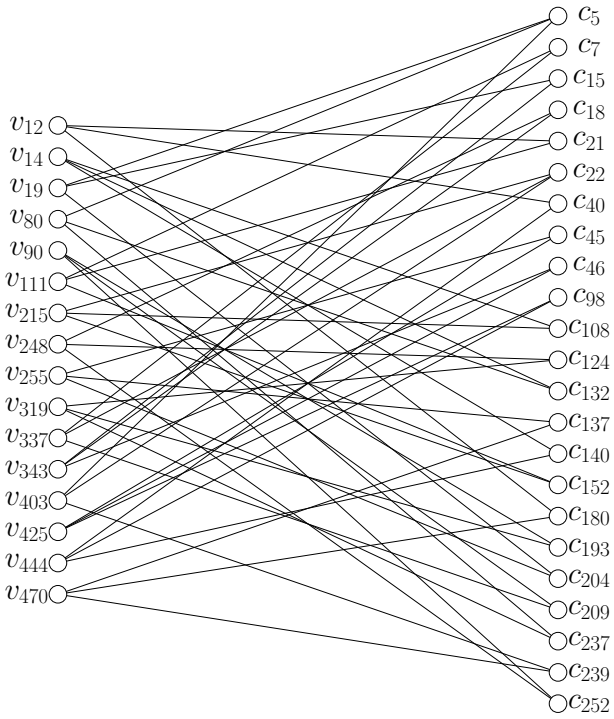

Figure 7: Graph induced by the minimum stopping set in the $(504, 252)$ Mackay code [15].

While these results demonstrate the effectiveness of a mathematical programming approach, they also raise an important question about the complexity of different codes. Consider the results for the $(1008, 504)$ Mackay and $(4896, 2474)$ Margulis codes. They are both represented by $(3, 6)$-regular Tanner graphs, with the latter having over four times more variables in the ILP formulation. However, note that finding a minimum stopping set for the former took over 250 times longer! This is a surprising result that reflects our limited knowledge of the stopping set polytope. A deeper understanding of the stopping set and trapping set polytopes formed by the ILP formulation is essential in order to extend the proposed approach to codes with over 10000 variables.

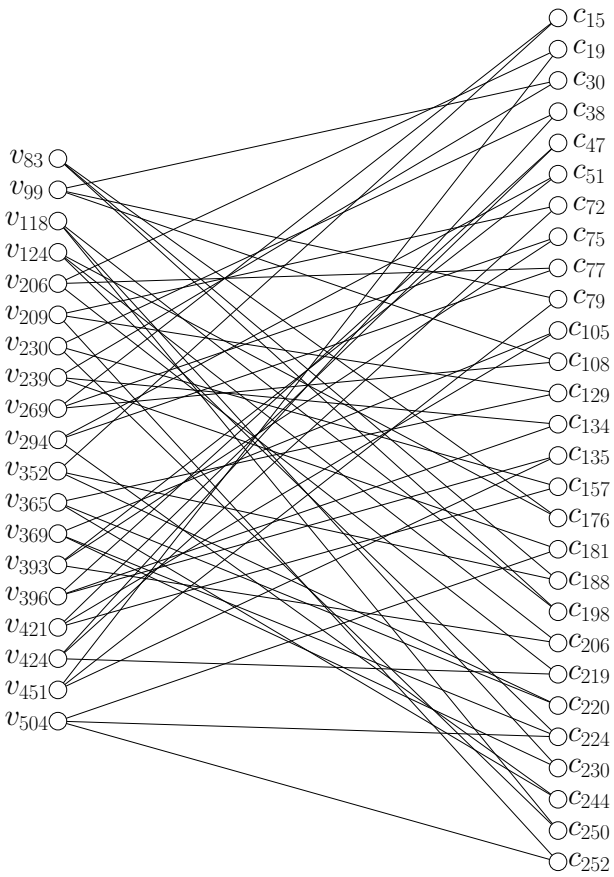

Figure 8: Graph induced by the minimum stopping set in the $(504, 252)$ progressive edge growth (PEG) code [15].

For the trapping set problems, we use the $(1008, 504)$ Mackay code [15] and the $(2640, 1320)$ Margulis code [11] as inputs. The values of a and b given range from 5 to 18. The time to solve each problem instance can be seen in Table 2. While the TS$(a, b)|_b$ and ETS$(a, b)|_b$ problems were solved surprisingly quickly by the proposed ILPs, the TS$(a, b)|_a$ and ETS$(a, b)|_a$ problems yielded similarly surprising results. Namely, that our programming approach was inefficient, especially in the case of the TS$(a, b)|_a$ problem. The stopping and trapping sets found, as well as the Gurobi output files, can be seen in the following repository:
https://www.cs.ucf.edu/~velasquez/StoppingSets/.

It is worth noting that every trapping set we found for the TS$(a, b)|_a$ and TS$(a, b)|_b$ problems was of elementary form. That is, each neighbor of the trapping set $T \subseteq V$ was connected to T at most twice. This correlates well with recent findings in [13] and [14], which demonstrate that minimum trapping sets are often of elementary form.

7.1 Caveats of Integer Programming

Modern integer linear programming (ILP) solvers utilize branch-and-bound algorithms to determine the optimal solution to an ILP. Branching is performed on fractional variable values by considering their ceiling and floor

	$(1008, 504)$ Mackay Code				$(2640, 1320)$ Margulis Code			
a, b value	TS$(a,b)\|_a$	TS$(a,b)\|_b$	ETS$(a,b)\|_a$	ETS$(a,b)\|_b$	TS$(a,b)\|_a$	TS$(a,b)\|_b$	ETS$(a,b)\|_a$	ETS$(a,b)\|_b$
5	53.42	0.41	236.17	1.61	520.95	0.37	5676.61	4.24
6	31.74	0.03	201.76	0.08	4373.78	0.14	39881.11	0.19
7	984.51	0.01	358.5	0.2	716.04	1.42	4308.11	0.49
8	4.1	0.41	211.67	0.67	18843.19	2.06	72311.44	1.51
9	4864.37	0.04	317.43	0.13	13504.13	0.12	4611.72	0.37
10	8143.11	0.39	712.88	0.19	–	0.17	92301.37	0.49
11	485919.5	0.39	870.13	0.71	–	1.55	15427.42	3.79
12	245.1	0.04	923.7	0.13	–	0.1	7566.81	0.35
13	–	0.03	1493.47	0.2	–	0.15	21374.26	0.49
14	–	0.39	1194.83	0.65	–	0.93	8148.08	1.51
15	–	0.04	6432.95	0.11	–	0.09	31408.17	0.49
16	–	0.04	737.11	0.19	–	0.08	618888.55	0.49
17	–	0.39	15557.55	0.67	–	0.16	65567.79	1.2
18	–	0.04	2174.66	0.11	–	0.09	1184184.74	0.53

Table 2: Execution time (in seconds) expended to solve each problem instance (in seconds) is reported for the $(1008, 504)$ Mackay code and $(2640, 1320)$ Margulis code.

integer values. The branch-and-bound tree is pruned as follows. The best current integer solution provides an upper bound on the objective function. The lower bound is computed from the linear programming relaxation of the problem instances in all active nodes of the tree. A careful study of the TS$(a,b)|_a$ and ETS$(a,b)|_a$ ILP performance reveals that the bottleneck in runtime stems from a lack of progress in the lower bound. Indeed, the upper bound obtained by the best integer solution quickly reaches the correct value. The main reason for this is that the difference between the polyhedron associated with the LP relaxation of the ETS$(a,b)_a$ ILP and the polyhedron consisting of the convex hull of all integer feasible solutions to the ETS$(a,b)_a$ ILP is too great [35]. Modern ILP solvers mitigate this difference by introducing additional constraints, called cuts, to the original formulation, thereby eliminating fractional solutions while maintaining all integer feasible solutions. Another way to improve the performance of a programming approach is by establishing an initial lower bound on our ILPs. Thus, developing new cutting plane algorithms and code-specific lower bounds for the TS$(a,b)_a$ and ETS$(a,b)_a$ is desirable.

8 Conclusion

In this paper, we examined both trapping sets and elementary trapping sets. In particular we showed that the TS$(a,b)|_b$ and ETS$(a,b)|_a$ problems are **NP-hard**. We also showed that these problems are **para-NP-hard** when parameterized by the degree of the Tanner graph. Additionally, we showed that the ETS$(a,b)|_a$ problem is **W[1]-hard** when parameterized by a and we designed an $O^*(|V|^a)$/$O^*(2^{|V|})$ algorithm for this problem.

We also modeled the problems of finding minimum stopping sets and trapping sets in codes using the language of mathematical programming. More specifically, integer linear programs were designed to find these sets. We demonstrated the effectiveness of our approach by finding these structures orders of magnitude faster than com-

peting approaches did. More importantly, we have shown that our method can handle larger problem instances than what is found in the literature. Indeed, for the $(4896, 2474)$ Margulis code, we have found stopping sets of size up to 48. As a point of reference, the previous state-of-the-art finds stopping sets of size up to 26 for this same Margulis code.

While we have been effective in exploring large problem instances, a unified approach which leverages the enumerative capabilities of competing methods is desirable. We believe the foregoing is a precursor to such an approach. As we argued earlier, an extensive study of the stopping set and trapping set polytopes is required in order to discover problem-specific cuts that may be added to the integer linear programs (ILPs) proposed in this paper. This is especially important for the (a, b) trapping set problems when a is given since the performance of our approach was poor for these problems. We plan to extend our programming-based method with such cuts as well as code-specific program lower bounds. We do this in the hope that the stopping set and trapping set problems will enjoy the same success that traditional combinatorial optimization problems have experienced with innovations in integer programming practices.

Future research into trapping sets and elementary trapping sets can proceed in the following directions:

1. In this paper, we showed that the $\text{ETS}(a, b)|_a$ problem is **W[1]-hard** when parameterized by a. However, we did not establish an upper bound on the parameterized complexity of this problem. Note that the $O^*(|V|^a)$ algorithm in this paper establishes that the decision version of this problem belongs to the class **XP**. It would be interesting to find a better upper bound on the parameterized complexity of this problem.

2. The problems examined in this paper are all optimization problems. However, we only looked at the complexity of finding the exact value of the optimum. Future research could examine the complexity of finding approximate solutions to these problems.

Acknowledgments

Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the Air Force Research Laboratory or the National Science Foundation. This research was supported in part by the National Science Foundation through Award CCF-1305054.

This research was supported in part by the Air-Force Office of Scientific Research through Grant FA9550-19-1-0177 and in part by the Air-Force Research Laboratory, Rome through Contract FA8750-17-S-7007. The views expressed are those of the authors and do not reflect the official policy or position of the Department of Defense or the US Government.

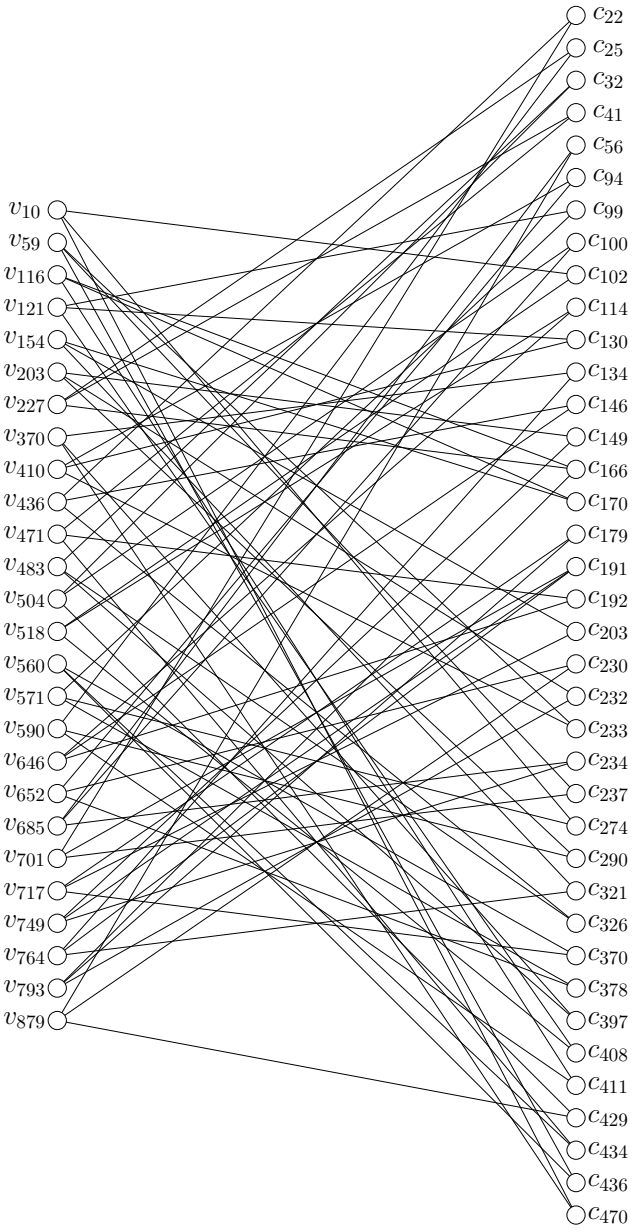

Figure 9: Graph induced by the minimum stopping set in the $(1008, 504)$ Mackay code [15].

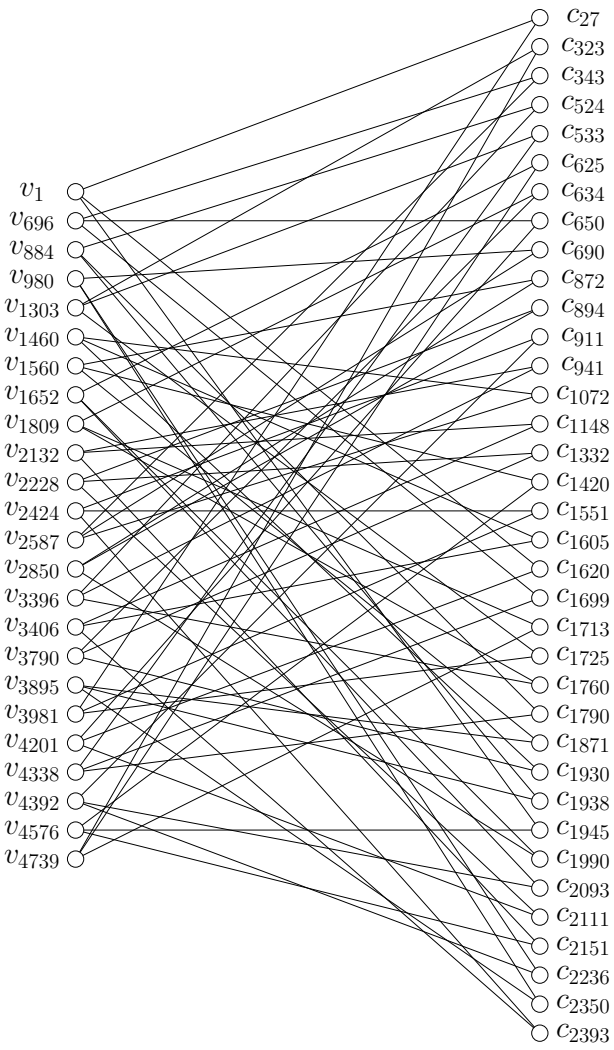

Figure 10: Graph induced by the minimum stopping set in the $(4896, 2474)$ Margulis code [11].

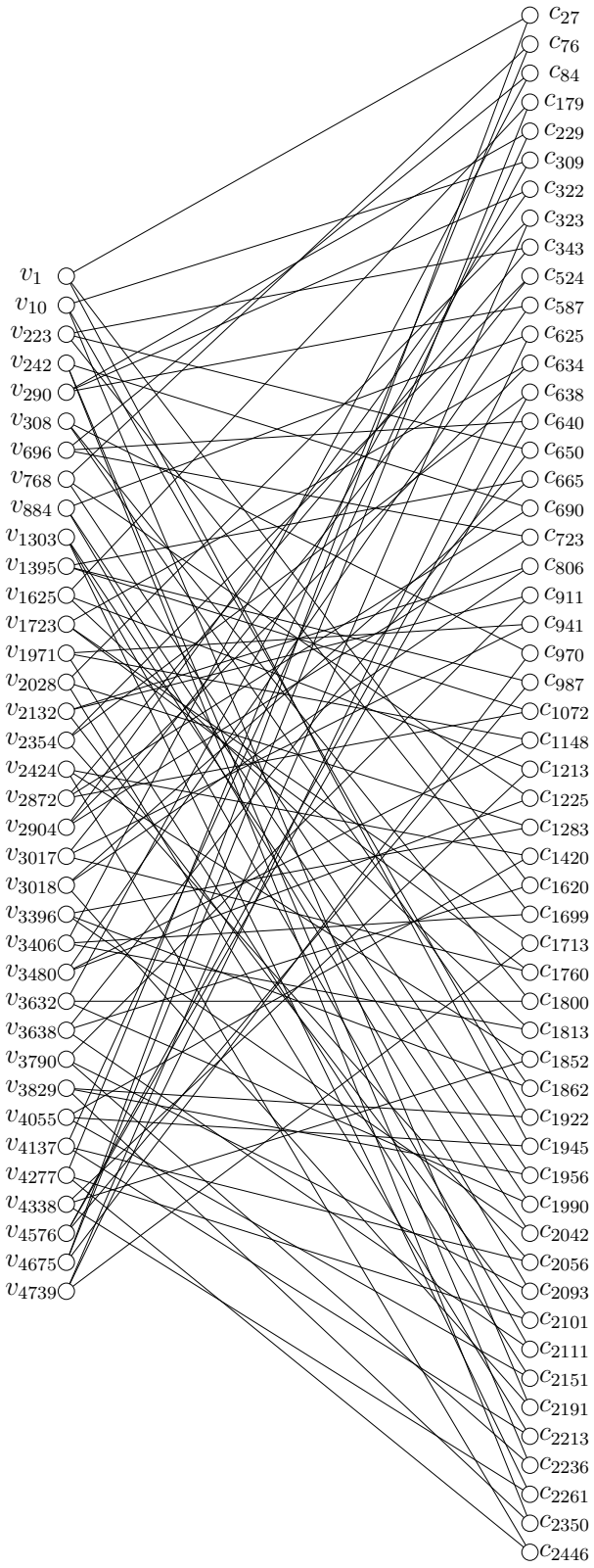

Figure 11: Graph induced by a stopping set of size 36 in the $(4896, 2474)$ Margulis code [11].

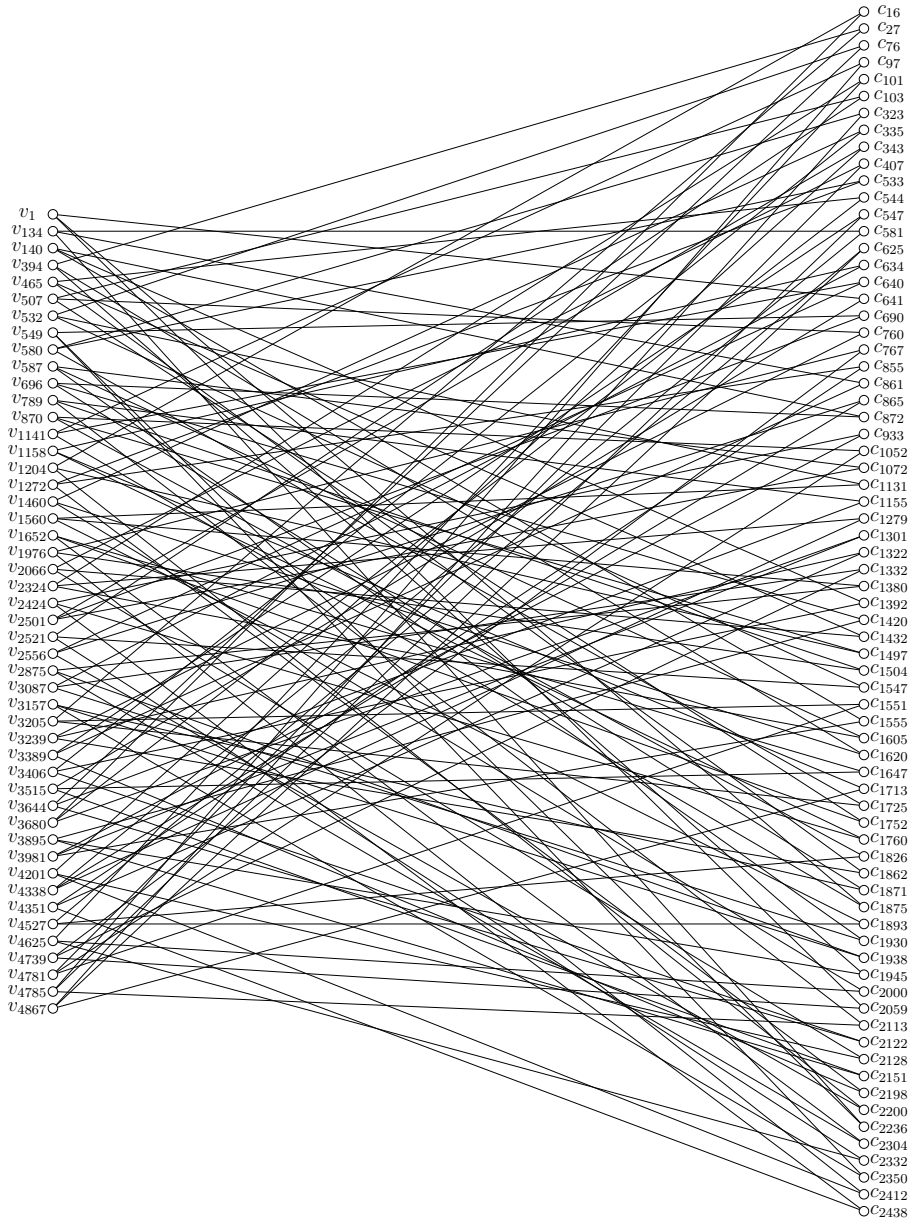

Figure 12: Graph induced by a stopping set of size 48 in the $(4896, 2474)$ Margulis code [11].

References

[1] Alvaro Velasquez, K. Subramani, and Steven L. Drager. Finding minimum stopping and trapping sets: An integer linear programming approach. In 5^{th} *International Symposium on Combinatorial Optimization (ISCO), 2018, Marrakesh, Morocco, April 11-13, 2018, Proceedings*, 2018.

[2] Tom Richardson. Error floors of ldpc codes. In *Proceedings of the annual Allerton conference on communication control and computing*, volume 41, pages 1426–1435. The University; 1998, 2003.

[3] Hua Xiao and Amir H Banihashemi. Estimation of bit and frame error rates of finite-length low-density parity-check codes on binary symmetric channels. *IEEE Transactions on Communications*, 55(12):2234–2239, 2007.

[4] Changyan Di, David Proietti, I Emre Telatar, Thomas J Richardson, and Rüdiger L Urbanke. Finite-length analysis of low-density parity-check codes on the binary erasure channel. *IEEE Transactions on Information theory*, 48(6):1570–1579, 2002.

[5] K Murali Krishnan and Priti Shankar. On the complexity of finding stopping set size in tanner graphs. In *40th Annual Conference on Information Sciences and Systems*, pages 157–158, 2006.

[6] Brian K Butler and Paul H Siegel. Error floor approximation for ldpc codes in the awgn channel. *IEEE Transactions on Information Theory*, 60(12):7416–7441, 2014.

[7] Xiao-Yu Hu and Evangelos Eleftheriou. A probabilistic subspace approach to the minimal stopping set problem. In *Turbo Codes&Related Topics; 6th International ITG-Conference on Source and Channel Coding (TURBOCODING), 2006 4th International Symposium on*, pages 1–6. VDE, 2006.

[8] Masanori Hirotomo, Yoshiho Konishi, and Masakatu Morii. On the probabilistic computation algorithm for the minimum-size stopping sets of ldpc codes. In *Information Theory, 2008. ISIT 2008. IEEE International Symposium on*, pages 295–299. IEEE, 2008.

[9] Eirik Rosnes and Øyvind Ytrehus. An efficient algorithm to find all small-size stopping sets of low-density parity-check matrices. *IEEE Transactions on Information Theory*, 55(9):4167–4178, 2009.

[10] Gyu Bum Kyung and Chih-Chun Wang. Exhaustive search for small fully absorbing sets and the corresponding low error-floor decoder. In *Information Theory Proceedings (ISIT), 2010 IEEE International Symposium on*, pages 739–743. IEEE, 2010.

[11] J Rosenthal and PO Vontobel. Construction of ldpc codes based on ramanujan graphs and ideas from margulis. In *Proc. 38th Annual Allerton Conf. on Communication, Computing and Control*, pages 248–257.

[12] Andrew McGregor and Olgica Milenkovic. On the hardness of approximating stopping and trapping sets. *IEEE Transactions on Information Theory*, 56(4):1640–1650, 2010.

[13] Mehdi Karimi and Amir H Banihashemi. On characterization of elementary trapping sets of variable-regular ldpc codes. *IEEE Transactions on Information Theory*, 60(9):5188–5203, 2014.

[14] Yoones Hashemi and Amir H Banihashemi. Lower bounds on the size of smallest elementary and non-elementary trapping sets in variable-regular ldpc codes. *IEEE Communications Letters*, 2017.

[15] David JC MacKay. Encyclopedia of sparse graph codes, 2005.

[16] Chih-Chun Wang, Sanjeev R Kulkarni, and H Vincent Poor. Upper bounding the performance of arbitrary finite ldpc codes on binary erasure channels. In *Information Theory, 2006 IEEE International Symposium on*, pages 411–415. IEEE, 2006.

[17] Chih-Chun Wang, Sanjeev R Kulkarni, and H Vincent Poor. Exhausting error-prone patterns in ldpc codes. *arXiv preprint cs/0609046*, 2006.

[18] Eirik Rosnes, Øyvind Ytrehus, Marcel A Ambroze, and Martin Tomlinson. Addendum to ?an efficient algorithm to find all small-size stopping sets of low-density parity-check matrices?. *IEEE transactions on information theory*, 58(1):164–171, 2012.

[19] Akin Tanatmis, Stefan Ruzika, Horst W Hamacher, Mayur Punekar, Frank Kienle, and Norbert Wehn. Valid inequalities for binary linear codes. In *2009 IEEE International Symposium on Information Theory*, pages 2216–2220. IEEE, 2009.

[20] Akin Tanatmis, Stefan Ruzika, Mayur Punekar, and Frank Kienle. Numerical comparison of ip formulations as ml decoders. In *2010 IEEE International Conference on Communications*, pages 1–5. IEEE, 2010.

[21] Mayur Punekar, Frank Kienle, Norbert Wehn, Akin Tanatmis, Stefan Ruzika, and Horst W Hamacher. Calculating the minimum distance of linear block codes via integer programming. In *2010 6th International Symposium on Turbo Codes & Iterative Information Processing*, pages 329–333. IEEE, 2010.

[22] Michael Helmling, Stefan Ruzika, and Akın Tanatmis. Mathematical programming decoding of binary linear codes: Theory and algorithms. *IEEE transactions on information theory*, 58(7):4753–4769, 2012.

[23] Michael Helmling, Eirik Rosnes, Stefan Ruzika, and Stefan Scholl. Efficient maximum-likelihood decoding of linear block codes on binary memoryless channels. In *2014 IEEE International Symposium on Information Theory*, pages 2589–2593. IEEE, 2014.

[24] Alexander Vardy. The intractability of computing the minimum distance of a code. *IEEE Transactions on Information Theory*, 43(6):1757–1766, 1997.

[25] Chih-Chun Wang. On the exhaustion and elimination of trapping sets: Algorithms & the suppressing effect. In *Information Theory, 2007. ISIT 2007. IEEE International Symposium on*, pages 2271–2275. IEEE, 2007.

[26] Mehdi Karimi and Amir H Banihashemi. An efficient algorithm for finding dominant trapping sets of irregular ldpc codes. In *Information Theory Proceedings (ISIT), 2011 IEEE International Symposium on*, pages 1091–1095. IEEE, 2011.

[27] Ralf Koetter and Pascal O Vontobel. Graph-covers and iterative decoding of finite length codes. In *Proc. 3rd Intern. Symp. on Turbo Codes and Related Topics*, pages 1–5, 2003.

[28] Tao Tian, Christopher R Jones, John D Villasenor, and Richard D Wesel. Selective avoidance of cycles in irregular ldpc code construction. *IEEE Transactions on Communications*, 52(8):1242–1247, 2004.

[29] K Murali Krishnan and L Sunil Chandran. Hardness of approximation results for the problem of finding the stopping distance in tanner graphs. In *International Conference on Foundations of Software Technology and Theoretical Computer Science*, pages 69–80. Springer, 2006.

[30] Ali Dehghan and Amir H Banihashemi. Hardness results on finding leafless elementary trapping sets and elementary absorbing sets of ldpc codes. *IEEE Transactions on Information Theory*, 65(7):4307–4315, 2019.

[31] Robert Crowston, Gregory Gutin, and Mark Jones. Note on max lin-2 above average. *Information Processing Letters*, 110(11):451–454, 2010.

[32] Paola Alimonti and Viggo Kann. Some apx-completeness results for cubic graphs. *Theoretical Computer Science*, 237(1-2):123–134, 2000.

[33] Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015.

[34] Gurobi Optimization. Inc.,?gurobi optimizer reference manual,? 2014. *URL: http://www. gurobi. com*, 2014.

[35] Ed Klotz and Alexandra M Newman. Practical guidelines for solving difficult mixed integer linear programs. *Surveys in Operations Research and Management Science*, 18(1):18–32, 2013.