Machine Learning and Algorithmic Techniques for Error Correction

Anxiao (Andrew) Jiang
Computer Science and Engineering Department
Texas A&M University
College Station, TX 77843-3112, USA
ajiang@cse.tamu.edu

***Abstract*—Many types of data, such as natural languages, have rich internal redundancy, which can be used to substantially improve the error correction performance. This work studies machine learning and algorithmic techniques for correcting errors in compressed or uncompressed languages.**

I. Introduction

The amount of data stored in the Internet is growing exponentially fast. With this growth, how to ensure long-term data reliability for all data also becomes more challenging. To assist error-correcting codes (ECC), the redundancy in the content of data itself can be utilized. This type of redundancy – such as features in languages, images and videos, structures in HTML files and databases, etc. – is referred to as *natural redundancy (NR)*, which supplements the more structured redundancy added by error-correcting codes [14], [15]. NR exists in both uncompressed and imperfectly compressed data, which are abundant in storage systems. That makes NR a promising tool to enhance data reliability.

With NR, a decoding system can be considered as consisting of two decoders: an ECC-Decoder, and an NR-Decoder. They work collaboratively to correct errors or erasures in the ECC codeword. We illustrate it by an example.

Example 1. *Consider texts compressed by an LZW algorithm that uses a fixed dictionary of size 2^ℓ. The dictionary has 2^ℓ text strings (called patterns) of variable lengths, where every pattern is encoded as an ℓ-bit codeword. Given a text to compress, the LZW algorithm scans T and partitions it into patterns, and maps them to codewords. For instance, if $\ell = 20$ and the text is "Flash memory is an electronic $\cdots$", the partitioning and LZW-codewords can be as illustrated in Fig.* 1 *(a).*

Now suppose some bits in the LZW-codewords are erased. An NR-Decoder can check all the possible solutions, map each solution back to patterns, and use a dictionary of words to eliminate those solutions that contain invalid words. (Such a dictionary of words has been commonly used in spell checkers.) If all the remaining solutions agree on the value of an erased bit, then that erasure is decoded by the NR-Decoder. For instance, suppose each LZW-codeword in Fig. 1 *(a) suffers from two erasures, which lead to four possible solutions/patterns (see Fig.* 1 *(b)). By combining the patterns for each codeword, we can rule out many solutions. For instance, the combination "should becnomially ars an ele" can be eliminated due to the invalid word "becnomially". In fact, the only combination without invalid words (without considering words on the boundary of the string, which might be part of a longer word) is "Flash memory is an ele", so the NR-Decoder can recover all six erasures in the three codewords.*

Suppose that the LZW-codewords, seen as information bits, are protected by a systematic ECC. Then the ECC-Decoder can correct erasures by parity-check constraints, and the NR-Decoder can correct erasures by NR. They can work collaboratively to maximize the number of correctable erasures. □

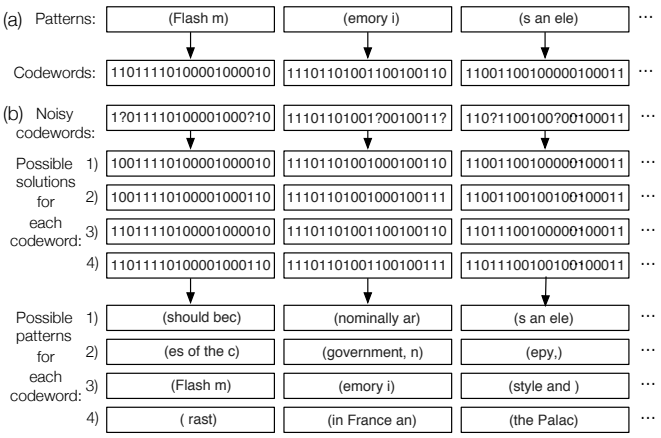

Fig. 1. (a) Compress a text by LZW. (b) NR-decoding for erasures.

As this paper is motivated by language-based NR, we would like to mention that an LZW algorithm with a dictionary of 2^{20} patterns (as in the above example) can compress the English language to 2.94 bits per character. The UNIX Compress command uses LZW with a smaller dictionary and so achieves a lower compression ratio. There are compression algorithms for languages with higher compression ratios (e.g., syllable-based Burrows-Wheeler Transform achieving 2 bits/character [19]). However, there is still a gap toward Shannon's estimation of 1.34 bits/character for the entropy of English [30], which gives motivation for NR-Decoders. And one may reasonably conjecture that a similar scenario exists for images and videos.

In this work, we propose a relatively generic decoding model for collaborative ECC-Decoding and NR-Decoding that is motivated by language-based NR. The model is shown in Fig. 2. The (compressed or uncompressed) data, seen as information bits, are encoded into a systematic ECC codeword. The NR-decoder uses a sliding window of L bits to check a segment of the data each time, and uses its NR to correct errors/erasures in it. We bound the size of the window to L bits because due to the lack of structures in NR, NR-decoding is often not as efficient as ECC-decoding and its complexity grows with L, so a finite L bounds the acceptable complexity of NR-decoding. The NR-Decoder works jointly with the ECC-Decoder to correct errors/erasures.

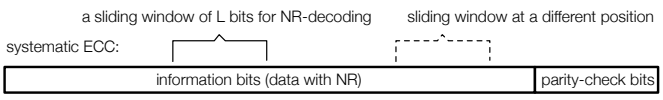

Fig. 2. A model for collaborative ECC-decoding and NR-decoding.

The above model can be applied to languages compressed by LZW codes or Huffman codes, where some practical decoding algorithms have been presented [14], [15], [20], [22], [31], [32]. In this paper, we study a basic theoretical problem for LDPC codes: when the number of erasures in a noisy LDPC codeword exceeds the decoding capability of the LDPC code's ECC-Decoder, what is the minimum number of erasures that an NR-Decoder needs to help correct so that the remaining erasures are decodable by the ECC-Decoder?

We have proposed and studied the above problem in [13], [16], where some results on the analysis of the problem's computational complexity, the design of approximation algorithms and the design of exact algorithms were presented. In this paper, we present a brief overview of those results in [13], [16].

It is worthwhile to mention that two main applications have been introduced in [13], [16] for the studied problem, one for *distributed storage*, and the other for *satellite-to-ground communication with feedback*. The application introduced here is yet another notable one, and is actually what motivated the study of the problem in the first place.

Let us define the problem more specifically. Let the LDPC code's ECC-Decoder be the following widely-used iterative belief-propagation (BP) algorithm: in each iteration, use every parity-check equation involving exactly one erasure to decode that erasure; and repeat until every equation involves zero or at least two erasures. If the ECC-Decoding fails, then we are left with a *stopping set*, which is a set of erasures such that every parity-check equation involving any of them involves at least two of them. If we represent the LDPC code by a bipartite Tanner graph, then a stopping set is a subset of variable nodes (representing erasures) such that a check node adjacent to any of them is adjacent to at least two of them.

We now define the capability and limitations of the NR-Decoder. Suppose that for any sliding window of L bits, the NR-Decoder can always correct its erasures if the number of erasures in the window is at most α. Given a stopping set, the objective is to use the NR-Decoder to correct sufficiently many erasures so that the remaining erasures are correctable by the ECC-Decoder. However, notice that since NR-Decoding is typically less efficient than ECC-Decoding, there is an associated cost. Let β denote the number of L-bit windows (at β different locations) used by NR-Decoding. Whether the NR-decoding is implemented in hardware or software (where decoding circuits or software can choose the location of each window), the overall circuit complexity and/or time complexity is proportional to β. (For instance, if circuits are used to decode the windows in parallel, then β circuits are needed for β windows.) Therefore, we need to *minimize* the number of windows used by NR-decoding, and choose the locations of the windows carefully for that purpose.

In this paper, we will study a special case of the above problem by setting $L = \alpha = 1$, and we assume that the sliding windows can cover both information bits and parity-check bits. Although it may seem too restrictive at first sight, there are several reasons that still make it quite meaningful. First, storage systems often have low raw bit-error-rates (e.g., less than 0.5%), so for a relatively short window (e.g, tens of bits), the number of erasures in it is often at most 1, which can usually be corrected very effectively by NR-Decoding [15]. In such cases, having $L \geq 1$, $\alpha = 1$ is similar to having $L = \alpha = 1$. Second, ECCs in storage systems often have high rates (e.g., over 0.93), which can make the sliding window's access to all codeword bits similar to accessing only information bits (since information bits are the majority of bits). Third, understanding the basic case of $L = \alpha = 1$ will be the basis for understanding the more general case of $L \geq \alpha \geq 1$. And last but not least, the case $L = \alpha = 1$ corresponds to a fundamental problem for LDPC codes: assume there is a powerful and unrestricted Oracle decoder that can correct any erasure, but its decoding comes at a high cost; then, how to minimize the number of erasures the Oracle decoder needs to correct in order to make the remaining erasures decodable by the ECC-Decoder? We believe the problem is theoretically important in its own right.

The problem to study can now be defined formally as follows. Let $G = (V \cup C, E)$ be a bipartite graph, where V (representing erasures) is a subset of the variable nodes in an LDPC code's Tanner graph, C is a subset of the check nodes in the same Tanner graph such that every node in C is adjacent to at least one node in V, and E is the set of edges in the Tanner graph with one endpoint in V and another endpoint in C. If every node in C has degree two or more, then G is called a *Stopping Graph* and V is called a *Stopping Set*. If an iterative BP algorithm (as introduced earlier) that runs on G can decode all the variable nodes in V (where every variable node in V is an erasure), then V is called a *Decodable Set* (or simply *decodable*); otherwise, it is a *Non-Decodable Set* (or simply *non-decodable*). Note that a Stopping Set must be a Non-Decodable Set, but not *vice versa*. The problem we study, called *Stopping-Set Elimination (SSE) Problem*, is as follows.

Definition 2. *Given a Stopping Graph $G = (V \cup C, E)$, how to remove the minimum number of variable nodes from V such that the remaining variable nodes are decodable?*

The removed variable nodes represent NR-decoded erasures. Clearly, after the removal, the remaining nodes will no longer contain any Stopping Set.

The rest of the paper is organized as follows. In Section II, we review works related to error correction by natural redundancy (NR). In Section III, we show that the SSE Problem is NP hard. In Section IV, we discuss an approximation algorithm for the SSE_1 Problem. In Section V, we analyze the effect of RBER (raw bit error rate) on achieving good approximation ratios, and present algorithms that return optimal solutions to SSE Problems when the stopping graphs form trees. In Section VI, we extend the study to stopping graphs with cycles, and present a number of results, including analyzing properties of optimal solutions and the design of approximate and exact algorithms. In Section VII, we conclude the paper.

II. Related Works

In this section, we present a brief review of existing works that are related to error correction by NR.

Error-correction with NR is related to joint source-channel coding and denoising. The idea of using the inherent redundancy in a source – or the leftover redundancy at the output of a source encoder – to enhance the performance of the ECC has been studied within the field of joint source-channel coding. In [11], source-controlled channel coding using a soft-output Viterbi algorithm is considered. In [4], a trellis based decoder is used as a source decoder in an iterative decoding scheme. Joint decoding of Huffman and Turbo codes is proposed in [10]. In [12], joint decoding of variable length codes (VLCs) and convolutional/Turbo codes is analyzed. Applications of turbo codes to image/video transmission are shown in [8], [25] and [17]. Joint decoding using LDPC codes for VLCs and images are illustrated in [26] and [27], respectively. However, not many works have considered JSCC specifically for language-based sources, and exploiting the redundancy in the language structure via an efficient decoding algorithm remains as a significant challenge. Related to joint source-channel coding, denoising is also an interesting and well studied technique [2], [5], [6], [7], [21], [24], [23], [28], [35]. A denoiser can use the statistics and features of input data to reduce its noise level for further processing. For discrete memoryless channels with stationary input sequences, a universal algorithm that performs asymptotically as well as optimal denoisers are given in [33]. The algorithm is also universal for a semi-stochastic setting, where the channel input is an individual sequence and the randomness in the channel output is solely due to the channel's noise.

Spell-checking softwares are a typical example of using NR to correct errors in languages. They are widely used in text editors. A spell-checking software usually works at the character level (namely, it does not consider how characters or text strings are encoded by bits), is for uncompressed texts, and uses the validity of words and the correctness of grammar to correct errors that appear in the typing of texts.

Using NR to correct errors at the bit level in compressed texts has been studied in a number of works. In [20], texts compressed by Huffman coding is considered, and a dynamic programming algorithm is used to partition the noisy bit sequence into subsequences that represents words, and to select likely solutions based on the frequencies of words and phrases. In [14], texts that are compressed by Huffman coding and then protected by LDPC codes are studied. An efficient greedy algorithm is used to decompress the noisy bit string, and partition it into stable and unstable regions based on whether each region contains recognizable words and phrases. The stable and unstable regions have polarized RBERs, which are provided as soft information to the LDPC code for better decoding performance. The algorithm is enhanced in [22] by a machine learning method for content recognition, and an iterative decoding algorithm between the NR-Decoder and the ECC-Decoder is used to further improve performance. In [32], texts compressed by Huffman coding and protected by Polar codes are studied. The validity of words is used to prune branches in a list sequential decoding algorithm, and a trie data structure for words is used to make the algorithm more efficient. A concatenated-code model that views the text with NR as the outer code and the Polar code as the inner code is considered, and the rate improvement for the Polar code due to NR is analyzed. That model is further studied in [31], where an optimal algorithm that maximizes the code rate improvement by unfreezing some frozen bits to store information is presented. A model that views NR as the output of a side information channel at the channel decoder is also studied, where NR is shown to improve the random error exponent.

III. NP-Hardness of SSE Problem

It has been proved in [16] that the SSE Problem is NP-hard. The proof has two steps: first, using the well-known Set Cover Problem, we prove that a related covering problem where nearly all elements (more specifically, all but at most one) are covered – which we call the *Pseudo Set Cover Problem* – is NP-complete; then, we reduce the latter problem to the SSE Problem.

Theorem 3. *The SSE Problem is NP-hard.*

We can extend the SSE Problem by considering the time for BP decoding [16]. After the nodes in an Elimination Set are removed (namely, after NR-decoding corrects those erasures), the remaining erasures are guaranteed to form a Decodable Set, and therefore the BP decoder can correct them. However, there is no guarantee on *how many iterations* are needed by the BP decoder to correct the remaining erasures. Here we assume a standard parallel-implementation of BP decoding: in each iteration, first, all variable nodes transmit their values to neighboring check nodes in parallel; then, all check nodes use incoming messages to correct erasures and send the decoding

results back to variable nodes, also in parallel. So the time for BP decoding can be measured by the number of BP iterations.

It can be seen that for a Stopping Set of n variable nodes (namely, n erasures), after an Elimination Set is removed, the BP decoder may still use as many as $\Theta(n)$ iterations to correct the remaining erasures.

For BP decoding, its decoding time is an important measure of performance. So it is useful to limit the number of iterations needed by BP decoding, which offers a performance guarantee. That motivates us to study this extended SSE Problem.

Definition 4. *Given a Stopping Graph $G = (V \cup C, E)$ and an integer k, how to remove the minimum number of variable nodes from V such that the remaining variable nodes can be corrected by the BP decoder in no more than k iterations?*

We call the above problem the SSE_k Problem. In comparison, the SSE Problem studied earlier has no constraint on k, so it can be seen as the SSE_∞ Problem.

We have already proved that SSE_∞ is NP-hard. The question now is: if k is a constant – namely, we want the BP decoding to finish within a fixed number of iterations – does the SSE_k problem become polynomial-time solvable? A positive answer seems possible at first sight, because having a small k puts more constraints on solutions and limits its search space. For example, if $k = 1$, to correct all remaining erasures in just one iteration, in the subgraph induced by the remaining variable nodes and their adjacent check nodes, every variable node needs to be adjacent to at least one check node of degree one. That is a very local property for the bipartite graph and can possibly make the problem simpler. However, our study below will give a negative answer. We will prove that even the

$$SSE_1$$

Problem is NP-hard.

There have been a number of works on the *node-deletion problem* (also called the *maximum subgraph problem*) [1], [9], [18], [34], which can be generally stated as follows: find the minimum number of vertices to delete from a given graph so that the remaining subgraph satisfies a property π. The node-deletion problem includes many well-known problems as special cases. Some examples are:

- Max Clique Problem: the property π is that the remaining subgraph is a complete graph.
- Feedback Vertex Set Problem: the property π is that the remaining subgraph has no cycles.
- Vertex Cover Problem: the property π is that the remaining subgraph contains only isolated nodes, without edges.

Some node-deletion problems are NP-complete on both general graphs and bipartite graphs, such as the feedback vertex set problem. However, some are NP-complete on general graphs but polynomial-time solvable on bipartite graphs, such as the vertex cover problem.

The SSE_k Problem is different from the previously studied problems in several ways. First, its property π is for the remaining subgraph to be decodable within k iterations, which is different from the property π in other problems. Second, the previous works focus on properties π that are *hereditary on induced subgraphs*, namely, whenever a graph G satisfies π, by deleting nodes from G, the remaining subgraphs also satisfies π [1], [9], [18], [34]. (For example, the property π for the max clique problem is hereditary because when nodes are removed from a complete graph, the remaining subgraph is also a complete graph. The same holds for the feedback vertex set problem and the vertex cover problem.) However, for the SSE_k Problem, the property π is *not hereditary*, because when a check node is removed, it may turn a Decodable Set into a Non-decodable Set. An example is shown below.

Example 5. *A Decodable Set is shown in Fig. 3 (a), which satisfies the property π of the SSE_k problem. As shown in Fig. 3 (b), after the check nodes c_1 and c_3 are removed, the remaining subgraph becomes non-decodable, which violates the property π. So for the SSE_k Problem, the property π is not hereditary.* □

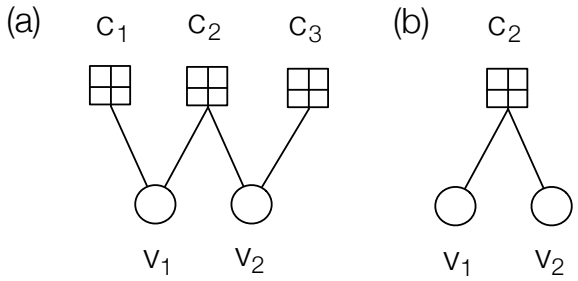

Fig. 3. (a) A graph with a Decodable Set. (b) After check nodes c_1 and c_3 are removed, the remaining variable nodes form a Non-decodable Set.

We have proved the NP-hardness of the SSE_1 Problem [16]. We use a reduction from the NP-complete Not-all-equal SAT Problem [29], similar to a proof technique used in [34]. However, due to the differences between the SSE_k problem and the previously studied node-deletion problems (as mentioned above), the two proofs also have significant differences: they use different mappings from the Not-all-equal SAT Problem to the target problem, which also lead to some substantially different properties in the mapped structures.

Theorem 6. *The SSE_1 Problem is NP-hard.*

IV. Approximation Algorithm for SSE_1 Problem

In [16], an approximation algorithm has been presented for the SSE_1 problem, for Stopping Graphs whose degrees of variable nodes and check nodes are upper bounded by d_v and d_c, respectively [16]. Its approximation ratio is

$$d_v(d_c - 1).$$

(Clearly, the same result also applies to regular (d_v, d_c) LDPC codes and irregular codes with the same constraint on

maximum degrees.) Note that the optimization objective is to minimize the size of the elimination set (namely, the number of removed variable nodes). So the approximation ratio means the maximum ratio of the size of an elimination set produced by the approximation algorithm to the size of an optimal (i.e., minimum) elimination set.

Due to space limitation, we skip the details of the algorithm here. The approximation algorithm has time complexity $O(d_v^2 d_c^2 |V|)$. And analysis shows that it achieves the following approximation ratio: let d_v and d_c denote the maximum degrees of variable nodes and check nodes, respectively, in the Stopping Graph $G = (V \cup C, E)$; then thealgorithm has an approximation ratio of

$$d_v(d_c - 1).$$

V. Analysis and Algorithms for SSE_k Problems

In this section, we present more analysis and algorithms for SSE_k Problems, including $k = \infty$. We first analyze how an important factor, RBER (raw bit-erasure rate), affects the performance of approximation algorithms, and show that for high-rate codes with high actual erasure rates, all algorithms have good approximation ratios. We then present exact algorithms for SSE_∞ and SSE_k Problems when the Stopping Graph is a tree (or a forest). The algorithms output optimal solutions and have linear time complexity.

A. Effect of RBER for Approximation Algorithms

We first analyze the effect of RBER for approximation algorithms. Consider an (N, K) LDPC code with N codeword bits and K information bits (where $K < N$), whose code rate is $R \triangleq K/N$. Let $G = (V \cup C, E)$ be its Stopping Graph, where V is the Stopping Set. The higher RBER is, the greater $|V|$ becomes on average. Let $\epsilon \triangleq |V|/N$ be called the *actual erasure rate relative to stopping set* V. The following result can be derived [13].

Theorem 7. *For the SSE_k Problem, if $\epsilon > 1 - R$, then the approximation ratio of any algorithm is at most*

$$\frac{\epsilon}{\epsilon - (1 - R)}.$$

So for high rate codes (where R approaches 1), if the RBER is high (which approaches 1), then with high probability, ϵ also approaches 1. In this case, $\frac{\epsilon}{\epsilon-(1-R)}$ approaches 1, so all algorithms have good approximation ratios.

B. Exact Algorithm for SSE_∞ Problem with Stopping Tree

The Stopping Graph $G = (V \cup C, E)$ can be a tree, especially when the RBER is low. In this case, we call G a *Stopping Tree*. Note that if G is a forest, the SSE_k Problem can be solved for each of its tree components independently.

In this subsection, we discuss an efficient and exact algorithm for the SSE_∞ Problem, which has been presented in [16]. The algorithm will be extended to the SSE_k Problem for general k subsequently.

The algorithm first runs BFS to generate G_{BFS}. It then processes the nodes in the reverse order of their labels (from $v_{|V|+|C|}$ to v_1). Every time it comes to a node v_i, if v_i is a variable node and has siblings (of smaller labels), its siblings are included in the Elimination Set. The root v_1 is also included in the Elimination Set. For details of the algorithm, please refer to [16].

It is proved that the algorithm returns an optimal (i.e., minimum-sized) Elimination Set of $G = (V \cup C, E)$. And its time complexity is $O(|V| + |C|)$.

C. Exact Algorithm for SSE_k Problem with Stopping Tree

We now extend the previous analysis, and design an exact algorithm for the SSE_k Problem of linear time complexity.

The algorithm first runs BFS on G to get the tree G_{BFS} that labels nodes by $v_1, v_2, \cdots, v_{|V|+|C|}$, where v_1 is the root. Then (similar to the algorithm for SSE_∞), it processes the nodes in the reverse order of their labels, and keeps reducing the SSE_k Problem – actually, a more general form of the SSE_k Problem, – to smaller and smaller subtrees. For details of the algorithm, please refer to [16].

The algorithm can be proved to return an optimal (i.e., minimum-sized) k-iteration Elimination Set of $G = (V \cup C, E)$. It also has time complexity $O(|V| + |C|)$.

VI. SSE Problem for p-Cyclic Stopping Graphs

In the above, we introduced exact algorithms for stopping graphs that form trees. A more general case is that the stopping graph can be cyclic but its number of cycles is bounded. That also matches the most useful cases in practice, where the number of erasures exceeds but is not far away from the LDPC code's decoding threshold. So we focus on this more general case in this paper. Specifically, we focus on p-Cyclic Stopping Graphs defined as follows.

Definition 8. *For a Stopping Graph $G = (V \cup C, E)$, if at most p variable nodes can be removed to make the remaining graph acyclic, then G is called a p-Cyclic Stopping Graph.*

In the following, we analyze properties of the SSE_∞ Problem and present a scheme that finds optimal elimination sets for stopping graphs. We present an efficient linear-complexity algorithm based on the scheme for stopping graphs containing one cycle. We also present an approximation algorithm for the SSE_∞ Problem for p-cyclic stopping graphs, which has approximation ratio $\frac{2p}{c+1} + 1$ and polynomial time complexity, where $c \geq 0$ is an integer parameter that can be selected freely. We then present an extended polynomial-time approximation algorithm with the same approximation ratio for the SSE_k Problem for p-cyclic stopping graphs. Due to space limitation, we skip many details. More detailed analysis and algorithms can be found in [13].

A. SSE_∞ Problem for 1-Cyclic and p-Cyclic Stopping Graphs

In this sub-section, we study the SSE_∞ Problem. We first analyze the properties of stopping graphs, and present an (not

necessarily polynomial-time) algorithm that finds an optimal elimination set for any stopping graph. We then present two polynomial-time algorithms for stopping graphs with one cycle and for p-cyclic stopping graphs, respectively, which are variations of the first algorithm.

Let us first define several concepts. We call a graph *smooth* if every node in it is of degree at least two. We call a node in a graph a *smooth-node* if it is either in a cycle, or on a simple path that connects two cycles in the graph. We call a node v in a graph G a *tree-node* if v has degree one, or if removing v from G will break G into two or more disjoint subgraphs and at least one of those subgraphs is a tree that has only one node adjacent to v in G. Given a stopping graph $G = (V \cup C, E)$, if a node v in it is both a smooth-node and a tree-node, then we call v a *bridge-node*; furthermore, if $v \in V$ is a variable node (resp., if $v \in C$ is a check node), then we call v a *bridge-variable-node* (resp., a *bridge-check-node*). We let $\Lambda(G)$ denote the set of bridge-variable-nodes of G, and let $\Pi(G)$ denote the set of bridge-check-nodes of G.

Given a stopping graph $G = (V \cup C, E)$, let $\mathcal{S}_G$ denote the subgraph of G obtained by removing from G all the nodes that are tree-nodes but not bridge-nodes. We call $\mathcal{S}_G$ the *smooth-component* of G. Furthermore, let $\mathcal{C}_G$ denote the subgraph of $\mathcal{S}_G$ obtained by removing from $\mathcal{S}_G$ all the nodes in $\Pi(G)$. We call $\mathcal{C}_G$ the *core-component* of G.

Lemma 9. *[13] For any stopping graph $G = (V \cup C, E)$, $\mathcal{S}_G$ is a smooth graph. All the cycles in G are also in $\mathcal{S}_G$, and vice versa. The nodes in $\mathcal{S}_G$ are exactly those smooth-nodes in G. Furthermore, both $\mathcal{S}_G$ and $\mathcal{C}_G$ are stopping graphs (i.e., every check node in them is of degree at least two).*

Consider a stopping graph $G = (V \cup C, E)$ and a bridge-node v in it. By definition, removing v from G will break G into multiple subgraphs, at least one of which is a tree with exactly one node (say node u) adjacent to v in G. We call such a tree a *peripheral-tree* $\mathcal{T}_{pt,G}(u)$ in G, and call u its root. Let $\mathcal{A}_G(v)$ denote the set of roots of peripheral trees adjacent to v in G. Let $\mathcal{T}_{bt,G}(v)$ denote the subgraph of G containing these nodes: v, and all the nodes of those peripheral trees whose roots are in $\mathcal{A}_G(v)$ (namely, are adjacent to v in G). It can be seen that $\mathcal{T}_{bt,G}(v)$ is a tree, which we call a *bridge-tree*, and call v its root. It is not hard to see that G contains a smooth-component and a set of disjoint bridge-trees, where each bridge-tree overlaps the smooth-component at its root (a bridge node).

We will use the two algorithms in [16] that find optimal elimination sets for stopping trees – one algorithm for the SSE_∞ Problem and the other for the SSE_k Problem – in the algorithms of this paper. Due to space limitation, we do not include their details, and use them only as modules. For convenience, let us call the two algorithms the $Tree_\infty$ *Algorithm* and $Tree_k$ *Algorithm*, respectively. We note that the $Tree_\infty$ Algorithm has a property: *it can choose any variable node v in the stopping tree and ensure that the optimal elimination set it finds contains v.* (v was seen as the tree's root in [16].) That property will be used in our future analysis.

We now present a recursive algorithm (called $OPT_\infty(G)$) that finds an optimal elimination set for any stopping graph for the SSE_∞ Problem. It takes a stopping graph $G = (V \cup C, E)$ as input, and outputs an elimination set $\mathcal{E} \subseteq V$ for G.

Algorithm 10. $OPT_\infty(G)$*: [13]*

1) *If G is smooth, find an optimal elimination set $\mathcal{E} \subseteq V$ for G, and return $\mathcal{E}$. Otherwise, go to Step 2.*
2) *$\mathcal{E}_{core} \leftarrow OPT_\infty(\mathcal{C}_G)$. (Namely, use the core-component $\mathcal{C}_G$ as input to this algorithm to obtain an elimination set $\mathcal{E}_{core}$ for $\mathcal{C}_G$.)*
3) *For every bridge-node $v \in \Lambda(G) \cup \Pi(G)$, use the $Tree_\infty$ Algorithm to find an optimal elimination set $\mathcal{F}(v)$ for the bridge-tree $T_{bt,G}(v)$.*
4) *Return $\mathcal{E}_{core} \cup \big(\cup_{v \in \Lambda(G)} \mathcal{F}(v) - \{v\}\big) \cup \big(\cup_{v \in \Pi(G)} \mathcal{F}(v)\big)$.*

We now analyze the algorithm $OPT_\infty(G)$, and prove that it returns an optimal elimination set for $G = (V \cup C, E)$.

First, let $G_\alpha = (V_\alpha \cup C_\alpha, E_\alpha)$ denote the subgraph of G obtained by removing all peripheral trees rooted at nodes in $\cup_{v \in \Lambda(G)} \mathcal{A}_G(v)$, namely, peripheral-trees adjacent to bridge-variable-nodes. (Here $V_\alpha \subseteq V$ and $C_\alpha \subseteq C$.)

Lemma 11. *[13] Let $G = (V \cup C, E)$ be a stopping graph. For the SSE_∞ Problem, there exists an optimal elimination set $\mathcal{E}^* \subseteq V$ such that $\mathcal{E}^* \cap V_\alpha$ is an elimination set for G_α.*

Lemma 12. *[13] Let $G = (V \cup C, E)$ be a stopping graph, and let $\mathcal{C}_G = (V_{core} \cup C_{core}, E_{core})$ be its core-component, where $V_{core} \subseteq V$ and $C_{core} \subseteq C$. Then for the SSE_∞ Problem, there exists an optimal elimination set $\mathcal{F} \subseteq V$ such that $\mathcal{F} \cap V_{core}$ is an elimination set for $\mathcal{C}_G$.*

For any stopping graph G', let $\chi(G')$ denote the size of an optimal elimination set for G'.

Theorem 13. *Let $G = (V \cup C, E)$ be a stopping graph. Then*

$$\chi(G) = \chi(\mathcal{C}_G) - |\Lambda(G)| + \sum_{v \in \Lambda(G) \cup \Pi(G)} \chi(\mathcal{T}_{bt,G}(v)).$$

Theorem 14. *The algorithm $OPT_\infty(G)$ returns an optimal elimination set for $G = (V \cup C, E)$.*

The algorithm $OPT_\infty(G)$ finds an optimal elimination set, but it may not be a polynomial-time algorithm, because it is NP-hard to find an optimal elimination set for a smooth stopping graph (which can be derived from the NP-hardness of the SSE_∞ Problem). In this following, we first study the SSE_∞ Problem for stopping graphs containing just one cycle, which is the basic case for cyclic stopping graphs, and show that an efficient algorithm exists.

For a stopping graph $G = (V \cup C, E)$ that contains one cycle, its core-component $\mathcal{C}_G$ is either a cycle (if $|\Pi(G)| = 0$) or $|\Pi(G)|$ disjoint paths (if $|\Pi(G)| > 0$). If $\mathcal{C}_G$ is a cycle, then

its optimal elimination set contains a single variable node in the cycle. If $\mathcal{C}_G$ consists of $|\Pi(G)|$ disjoint paths, then its optimal elimination set contains $|\Pi(G)|$ variable nodes (one for each path). So based on Theorem 13, we see that an optimal elimination set for G has size

$$\chi(G) = \max\{|\Pi(G)|, 1\} - |\Lambda(G)| + \sum_{v \in \Lambda(G) \cup \Pi(G)} \chi(\mathcal{T}_{bt,G}(v)).$$

Based on the above analysis, the algorithm $OPT_\infty(G)$ can be simplified accordingly. (We skip details of the algorithm due to space limit.) Note that the $Tree_\infty$ Algorithm has linear time complexity [16]. It is not hard to see that the algorithm here also has linear time complexity $O(V + C + E)$.

We now present an approximation algorithm for p-cyclic stopping graphs. It uses an approximation algorithm for the Minimum Feedback Vertex Set (MFVS) Problem as a tool. Given an undirected graph $G' = (V', E')$, a *feedback vertex set* (FVS) is a set of vertices $S \subseteq V'$ such that every cycle in G' contains at least one vertex of S (namely, removing S will turn G' into an acyclic graph). The MFVS Problem is defined as follows: given an undirected graph $G' = (V', E')$ where every vertex $v \in V'$ has a non-negative cost $c(v)$, find an FVS in G' whose cost is minimized. The MFVS Problem has a 2-approximation algorithm [3], which we shall call the *MFVS Algorithm*.

Our approximation algorithm for the SSE_∞ Problem for a p-cyclic stopping graph $G = (V \cup C, E)$ is a modification of the Algorithm $OPT_\infty(G)$. In Step 1 of $OPT_\infty(G)$, instead of finding an optimal elimination set $\mathcal{E} \subseteq V$ for the smooth graph G, we find an approximate solution as follows:

- Step 1.1: Let c be a non-negative integer. Use exhaustive search to check if G has an elimination set of size at most c. If yes, return an optimal elimination set for G; otherwise, go to Step 1.2.
- Step 1.2: In G, let all variable nodes have cost 1, and let all check nodes have cost ∞. Run the MFVS Algorithm to find an FVS $F \subseteq V$ for G. (The high cost for check nodes ensures that F contains only variable nodes.)
- Step 1.3: Remove the nodes in F from G, and use BP decoding to further remove those nodes that become decodable until we get a new stopping graph $\hat{G} = (\hat{V} \cup \hat{C}, \hat{E})$. ($\hat{G}$ is acyclic because F is an FVS.) Then find an optimal elimination set $S^* \subseteq \hat{V}$ for $\hat{G}$ using the $Tree_\infty$ Algorithm. Return $F \cup S^*$ as the elimination set for G.

We let Step 2 through Step 4 of the algorithm $OPT_\infty(G)$ remain the same. Let us call the new algorithm $Approx_\infty(G)$. It has polynomial time complexity for constant parameter c because all its elements (the MFVS Algorithm, the $Tree_\infty$ Algorithm, the exhaustive search, the number of recursive calls) are of polynomial time. We now analyze its approximation ratio (define based on the size of the elimination set).

Theorem 15. *Algorithm $Approx_\infty(G)$ has approximation ratio*

$$\frac{2p}{c+1} + 1.$$

So when p is small, the approximation ratio can be small.

The SSE_k Problem is a generalization of the SSE_∞ Problem. In [13], we have presented an approximation algorithm for it for a p-Cyclic Stopping Graph $G = (V \cup C, E)$. It is the same as Step 1.1 through Step 1.3 of the Algorithm $Approx_\infty(G)$, except that we use the $Tree_k$ Algorithm to replace the $Tree_\infty$ Algorithm in Step 1.3; and we do not use the remaining steps (namely Step 2 through Step 4 of $OPT_\infty(G)$). (That is because here G is the original input to the algorithm, not its smooth subgraph obtained during the recursion.) Let us call the new algorithm $Approx_k(G)$.

Algorithm $Approx_k(G)$ also has an approximation ratio of $\frac{2p}{c+1} + 1$ and polynomial time complexity. The analysis is very similar to Algorithm $Approx_\infty(G)$, so we skip it details. Note that although the two algorithms have the same approximation ratio (which considers the worst case performance) and Algorithm $Approx_k(G)$ appears simpler (i.e., with fewer steps), Algorithm $Approx_\infty(G)$ is better optimized for the SSE_∞ Problem because its extra steps can further reduce the size of the output elimination set.

VII. Concluding Remarks

This paper summarizes a number of results on the stopping set elimination problem for LDPC decoding, which has applications to error correction that utilizes natural redundancy (NR) in data. How to mine NR in data effectively is a challenging problem. We are currently conducting research that uses deep learning techniques for mining information from NR that is useful for ECC decoding. The work can also be extended by studying more specific LDPC code constructions (e.g., spatially-coupled codes, etc.), and design corresponding SSE_k algorithms.

References

[1] B. Addis, M. D. Summa and A. Grosso, "Removing Critical Nodes from a Graph: Complexity Results and Polynomial Algorithms for the Case of Bounded Treewidth," available at $http://www.optimization-online.org/DB_FILE/2011/07/3112.pdf$.

[2] L. Alvarez, P. Lions and J. Morel, "Image Selective Smoothing and Edge Detection by Nonlinear Diffusion. II," in *SIAM Journal on numerical analysis*, vol. 29, no. 3, pp. 845–866, 1992.

[3] V. Bafna, P. Berman and T. Fujito, "A 2-approximation algorithm for the undirected feedback vertex set problem," in *SIAM J. Discret. Math.*, vol. 12, no. 3, pp. 289-297, 1999.

[4] R. Bauer and J. Hagenauer, "On Variable Length Codes for Iterative Source/Channel Decoding," in *Proceedings of Data Compression Conference*, pp. 273–282, 2001.

[5] A. Buades, B. Coll and J. Morel, "A Review of Image Denoising Algorithms, with a New One," in *Multiscale Modeling & Simulation*, vol. 4, no. 2, pp. 490530, 2005.

[6] P. Chatterjee and P. Milanfar, "Is denoising dead?" in *IEEE Transactions on Image Processing*, vol. 19, no. 4, pp. 895911, 2010.

[7] R. Coifman and D. Donoho, *Translation-invariant De-noising*, Springer, 1995.

[8] M. Fresia and G. Caire, "Combined Error Protection and Compression with Turbo Codes for Image Transmission Using a JPEG2000-like Architecture," in *Proc. ICIP*, pp. 821–824, 2006.

[9] T. Fujito, "Approximating Node-Deletion Problems for Matrodial Properties," in *Journal of Algorithms*, vol. 31, pp. 211–227, 1999.

[10] L. Guivarch, J. Carlach and P. Siohan, "Joint Source-channel Soft Decoding of Huffman Codes with Turbo-codes," in *Proceedings of Data Compression Conference (DCC)*, pp. 83-92, 2000.

[11] J. Hagenauer, "Source-controlled Channel Decoding," in *IEEE Transactions on Communications*, vol. 43, no. 9, pp. 2449–2457, 1995.

[12] M. Jeanne, J. Carlach and P. Siohan, "Joint Source-channel Decoding of Variable-length Codes for Convolutional Codes and Turbo Codes," in *IEEE Transactions on Communications*, vol. 53, no. 1, pp. 10-15, 2005.

[13] A. Jiang, "Elimination of Cyclic Stopping Sets for Enhanced Decoding of LDPC Codes," submitted to IEEE International Symposium on Information Theory, 2018.

[14] A. Jiang, Y. Li and J. Bruck, "Error Correction through Language Processing," in *Proc. IEEE Information Theory Workshop (ITW)*, 2015.

[15] A. Jiang, P. Upadhyaya, E. F. Haratsch and J. Bruck, "Error Correction by Natural Redundancy for Long Term Storage," in *Proc. Non-Volatile Memories Workshop (NVMW)*, 2017.

[16] A. Jiang, P. Upadhyaya, Y. Wang, K. R. Narayanan, H. Zhou, J. Sima and J. Bruck, "Stopping set elimination for LDPC codes," in *Proc. 55th Allerton Conference on Communication, Control and Computing*, 2017. Available at $http://faculty.cse.tamu.edu/ajiang/Publications/2017/StoppingSetElimination\_Allerton.pdf$.

[17] A. N. Kim, S. Sesia, T. Ramstad, and G. Caire, "Combined Error Protection and Compression using Turbo Codes for Error Resilient Image Transmission," in *Proceedings of International Conference on Image Processing (ICIP)*, vol. 3, pp. III-912-15, 2005.

[18] M. Kumar, S. Mishra, N. S. Devi and S. Saurabh, "Approximation Algorithms for Node Deletion Problems on Bipartite Graphs with Finite Forbidden Subgraph Characterization," in *Theoretical Computer Science*, vol. 526, pp. 90–96, 2014.

[19] J. Lansky, K. Chernik and Z. Vlckova. "Syllable-Based Burrows-Wheeler Transform," 2007.

[20] Y. Li, Y. Wang, A. Jiang and J. Bruck, "Content-assisted File Decoding for Nonvolatile Memories," in *Proc. 46th Asilomar Conference on Signals, Systems and Computers*, pp. 937–941, Pacific Grove, CA, 2012.

[21] M. Lindenbaum, M. Fischer, and A. Bruckstein, "On Gabor's Contribution to Image Enhancement," in *Pattern Recognition*, vol. 27, no. 1, pp. 18, 1994.

[22] J. Luo, Q. Huang, S. Wang and Z. Wang, "Error Control Coding Combined with Content Recognition," in *Proc. 8th International Conference on Wireless Communications and Signal Processing*, pp. 1–5, 2016.

[23] E. Ordentlich, G. Seroussi, S. Verdu, M. Weinberger and T. Weissman, "A Discrete Universal Denoiser and Its Application to Binary Images," in *Proc. International Conference on Image Processing*, vol. 1, pp. 117, 2003.

[24] E. Ordentlich, G. Seroussi, S. Verdu, and K. Viswanathan, "Universal Algorithms for Channel Decoding of Uncompressed Sources," *IEEE Trans. Information Theory*, vol. 54, no. 5, pp. 2243–2262, May 2008.

[25] Z. Peng, Y. Huang and D. Costello, "Turbo codes for Image Transmission – A Joint Channel and Source Decoding Approach," in *IEEE Journal on Selected Areas in Communications (JSAC)*, vol. 18, no. 6, pp. 868–879, 2000.

[26] C. Poulliat, D. Declercq, C. Lamy-Bergot, and I. Fijalkow, "Analysis and Optimization of Irregular LDPC Codes for Joint Source-channel Decoding," *in IEEE Communications Letter*, vol. 9, no. 12, pp. 1064–1066, 2005.

[27] L. Pu, and Z. Wu, A. Bilgin, M. Marcellin, and B. Vasic, "LDPC-based Iterative Joint Source-channel Decoding for JPEG2000," in *IEEE Transactions on Image Processing*, vol. 16, no. 2, pp. 577–581, 2007.

[28] L. Rudin, S. Osher and E. Fatemi, "Nonlinear Total Variation based Noise Removal Algorithms," in *Physica D: Nonlinear Phenomena*, vol. 60, no. 1, pp. 259–268, 1992.

[29] T. J. Schaefer, "The Complexity of Satisfiability Problems," in *Proc. 10th Annual ACM Symposium on Theory of Computing (STOC)*, pp. 216–226, 1978.

[30] C. E. Shannon, "Prediction and Entropy of Printed English," in *Bell System Technical Journal*, vol. 30, no. 1, pp. 50–64, 1951.

[31] Y. Wang, K. R. Narayanan and A. Jiang, "Exploiting Source Redundancy to Improve the Rate of Polar Codes," in *IEEE International Symposium on Information Theory (ISIT)*, Aachen, Germany, June 2017.

[32] Y. Wang, M. Qin, K. R. Narayanan, A. Jiang and Z. Bandic, "Joint Source-channel Decoding of Polar Codes for Language-based Sources," in *Proc. IEEE Global Communications Conference (Globecom)*, Washington D.C., December 2016.

[33] T. Weissman, E. Ordentlich, G. Seroussi, S. Verdu and M. Weinberger, "Universal Discrete Denoising: Known Channel," in *IEEE Transactions on Information Theory*, vol. 51, no. 1, pp. 5–28, 2005.

[34] M. Yannakakis, "Node-Deletion Problems on Bipartite Graphs," in *SIAM Journal on Computing*, vol. 10, no. 2, pp. 310–327, May 1981.

[35] L. Yaroslavsky and M. Eden, *Fundamentals of Digital Optics: Digital Signal Processing in Optics and Holography*, Springer-Verlag New York, Inc., 1996.