Decimation-Enhanced Finite Alphabet Iterative Decoders for LDPC codes on the BSC

Shiva Kumar Planjery, Bane Vasic
Dept. of Electrical and Computer Eng.
University of Arizona
Tucson, AZ 85721, U.S.A.
Email: {shivap,vasic}@ece.arizona.edu

David Declercq
ETIS
ENSEA/UCP/CNRS UMR 8051
95014 Cergy-Pontoise, France
Email: declercq@ensea.fr

Abstract—Finite alphabet iterative decoders (FAID) with multilevel messages that can surpass BP in the error floor region for LDPC codes on the BSC were previously proposed in [1]. In this paper, we propose decimation-enhanced decoders. The technique of decimation which is incorporated into the message update rule, involves fixing certain bits of the code to a particular value. Under appropriately chosen rules, decimation can significantly reduce the number of iterations required to correct a fixed number of errors, while maintaining the good performance of the original decoder in the error floor region. At the same time, the algorithm is much more amenable to analysis. We shall provide a simple decimation scheme for a particularly good 7-level FAID for column-weight three messages that can surpass BP in the error floor region with much lower complexity. These decoders were derived by identifying potentially harmful subgraphs that could be trapping sets present in any finite-length code and designing a decoder that estimates the variables in the $k$-SAT problem. In this paper, we propose decimation-enhanced finite alphabet iterative decoders for LDPC codes on the BSC. Decimation, a method originally developed for solving constraint satisfaction problems in statistical physics, involves guessing the values of certain variables and fixing them to these values while continuing to estimate the remaining variables. In [13], Montanari et al. analyzed a BP-guided randomized decimation procedure that estimates the variables in the $k$-SAT problem. Dimakis and Wainwright used a similar notion in the form of facet guessing for linear programming (LP) based decoding in [14], and Chertkov proposed a bit-guessing algorithm in order to reduce error floors of LDPC codes under LP decoding [15]. In contrast, we propose a simple decimation procedure...
in this paper that serves as a guide to help the multilevel FAID algorithm to converge faster on a small number of errors. Our main insight is that the role of decimation should not necessarily be to correct errors, but to ensure that more variable nodes in the graph that initially receive right values from the channel are shielded from the erroneous messages emanating from the error nodes by decimating those correct variable nodes.

The rest of the paper is organized as follows. Section [I] provides preliminaries. In Section [III] we provide a detailed description of the decimation-aided FAID algorithm. Finally in Section [IV] we provide some theoretical as well as numerical results and conclude with a discussion.

II. PRELIMINARIES

Let $G = (V \cup C, E)$ denote the Tanner graph of an $(n, m)$ binary LDPC code $C$ with the set of variable nodes $V = \{v_1, \cdots, v_n\}$ and set of check nodes $C = \{c_1, \cdots, c_m\}$. $E$ is the set of edges in $G$. A code $C$ is said to be $d_v$-left-regular if all variable nodes in $V$ of graph $G$ have the same degree $d_v$.

The degree of a node is the number of its neighbors.

Let $r = (r_1, r_2, \cdots, r_n)$ be the input to the decoder from the BSC. A trapping set $T$ is a non-empty set of variable nodes in $G$ that are not eventually corrected by the decoder [8].

Note that during analysis of decoders in this paper, it is implicitly assumed that the all-zero codeword is transmitted. This is a valid assumption since we consider only symmetric decoders, as explained in [3].

A multilevel FAID $\mathcal{F}$, as defined in [1], is a 4-tuple given by $\mathcal{F} = (M, Y, \Phi_v, \Phi_c)$. The messages are levels confined to a message alphabet $M$ which is defined as $M = \{0, \pm L_i : 1 \leq i \leq M\}$, where $L_i \in \mathbb{R}^+$ and $L_i > L_j$ for any $i > j$. The set $Y$ denotes the set of possible channel values. For the case of BSC, $Y$ is defined as $Y = \{\pm C\}$, and for each variable node $v_i$ in $G$, the channel value $y_i \in Y$ is determined by $y_i = (1)^{v_i}C$, i.e., we use the mapping $0 \rightarrow C \land 1 \rightarrow -C$.

$\Phi_v : Y \times M^{d_v-1} \rightarrow M$ is the update rule used at a variable node with degree $d_v$. The map $\Phi_v$ can be described in closed form as a linear or non-linear threshold function, or simply as a look-up table (LUT). For this paper, we shall use the LUT form. The LUT of $\Phi_v$ for a particularly good 7-level FAID that we shall be using throughout this paper is given in Table [I] for $y_i = +C$ (for $y_i = -C$, the LUT can be obtained by symmetry).

$$
\Phi_v(m_1, \ldots, m_{d_v-1}) = \prod_{j=1}^{d_v-1} \min_{j \in \{1, \ldots, d_v-1\}} (|m_j|) \quad \text{where } \text{sgn denotes the standard signum function.}
$$

TABLE I.

| $m_1 \backslash m_2$ | $-L_3$ | $-L_2$ | $-L_1$ | 0 | $L_1$ | $L_2$ | $L_3$ |
|---------------------|-------|-------|-------|---|------|------|------|
| $-L_3$              | $-L_3$| $-L_2$| $-L_2$| $-L_1$ | $L_1$ | $L_1$ | $L_1$ |
| $-L_2$              | $-L_2$| $-L_1$| $-L_1$| 0 | $L_1$ | $L_2$ | $L_2$ |
| 0                   | $-L_1$| 0     | 0     | $L_1$ | $L_2$ | $L_3$ | $L_3$ |
| $L_1$               | $-L_1$| $L_1$ | $L_2$ | $L_2$ | $L_3$ | $L_3$ | $L_3$ |
| $L_2$               | $-L_1$| $L_1$ | $L_2$ | $L_3$ | $L_3$ | $L_3$ | $L_3$ |
| $L_3$               | $-L_1$| $L_1$ | $L_3$ | $L_3$ | $L_3$ | $L_3$ | $L_3$ |

Remark: In this paper, we only consider decoders that use a single decimation rule $\beta$ but the same rule may be used in different iterations. Hence, $\beta$ is not iteration dependent, and whenever we refer to $\beta$, it implies the output of $\beta$ for node $v_i$ at the end of some $l^{th}$ iteration. Note that the function $\beta$ is symmetric.

A decimation-enhanced multilevel FAID $\mathcal{F}^D$ is defined as a 4-tuple given by $\mathcal{F}^D = (M, Y, \Phi_v^D, \Phi_c^D)$, where $M, Y, \Phi_c$ are the same maps defined for a multilevel FAID. The map $\Phi_v^D : Y \times M^{d_v-1} \times \{0, 1\} \rightarrow M$ is similar to $\Phi_v$ of the decoder $\mathcal{F}$ except that it uses the output of $\beta$ in some $l^{th}$ iteration, $\beta^*_l$, as an additional argument in the function. For the sake of simple exposition, we shall define $\Phi_v^D$ for the case of column-weight the messages being propagated within the subgraph for a certain a number of iterations. Consequently, for a certain number of iterations, decoding on the subgraph can be carried out in an isolated manner without explicit knowledge of its neighborhood.
three codes and 7 levels. Let \( m_1 \) and \( m_2 \) denote incoming messages to a node \( v_i \in V \) in the \( k^{th} \) iteration. Then \( \Phi^D_v \) is defined as

\[
\Phi^D_v(m_1, m_2, y_i, \beta_i) = \begin{cases} 
\Phi_v(m_1, m_2, y_i), & \beta_i = 0 \\
\beta_i L_3, & \beta_i = \pm 1 
\end{cases}
\]

We now provide a simple example to illustrate the potential benefits of decimation, and then we will describe the basic decimation scheme used in this paper.

A. Motivating example

Consider a particular 4-error configuration on a Tanner graph \( G \), whose induced subgraph forms an 8-cycle as shown in Fig. 1. In the figure, black circles represent the variable nodes initially in error, whereas white circles represent the initially correct nodes that are in the neighborhood of the 4-error pattern. The black and white squares denote the degree one and degree two checks in the induced subgraph respectively.

Let \( V' = \{v_1, v_2, v_3, v_4\} \) denote the set of variable nodes initially in error. Let \( C^1 = \{c_1, c_3, c_5, c_7\} \) denote the set of degree one checks and \( C^2 = \{c_2, c_4, c_6, c_8\} \) denote the set of degree two checks. We shall now examine the behavior of MP on this particular error configuration from the context of multilevel FAID algorithms without any assumptions on its neighborhood. Messages with a positive sign will be referred to as good messages, and messages with a negative sign will be referred to as bad messages (under all-zero codeword assumption). Also a weakly good or bad message refers to \( \pm L_1 \), and a strongly good or bad message refers to \( \pm L_1 \) where \( L_1 > L_2 \). In the first iteration, for all \( v_i \in V' \), \( m_1(v_i, N(v_i)) \) will be weakly bad, and for all \( v_j \in N(C^1 \cup C^2) \setminus V' \), \( m_1(v_j, N(v_j) \cap (C^1 \cup C^2)) \) entering into the subgraph will be weakly good messages. In the second iteration, for all \( v_i \in V' \), \( m_2(v_i, N(v_i) \cap C^2) \) will be either weakly good or weakly bad depending on the \( \Phi_v \) (such as Table 1), but \( m_2(v_i, N(v_i) \cap C^1) \) which are messages sent to checks in \( C^1 \), will be strongly bad. As a result, variable nodes \( v_i \in N(C^1) \setminus V' \) will start receiving strongly bad messages. If the decoder does not converge within the next few iterations, then these bad messages become more strongly bad and can subsequently spread further to other nodes in the graph depending how dense the neighborhood is. Eventually too many nodes get corrupted by the bad messages being propagated in the graph causing the decoder to fail.

Remarks: 1) At the \( k^{th} \) iteration, there may have been many variable nodes \( v_i \) such that \( v_i \notin N(C^1 \cup C^2) \), whose incoming messages already converged to the right value in some \( k' < k \) iteration, but eventually these nodes became corrupted by the bad messages flowing out of the subgraph. 2) If certain variable nodes initially correct in the neighborhood of the subgraph induced from the error pattern, are isolated from the bad messages possibly through decimation, then the decoder is more likely to converge. This is precisely where the role of decimation becomes important.

B. Basic scheme for the decimation-enhanced FAID algorithm

We propose a scheme that uses successive decimations for a certain number of iterations. Let the number of successive decimations be \( N_d \). The skeleton of the algorithm is given below. Note that for this proposed scheme, decimation starts at the end of the third iteration (reasons for which we will explain later), and after each decimation, the decoder is restarted.

**Algorithm 1 Decimation-enhanced FAID algorithm**

1. Initialize \( \beta_i = 0 \) \( \forall v_i \in V \).
2. Run the decoder using update maps \( \Phi_v \) and \( \Phi_c \) already defined for the 7-level FAID decoder.
3. At the end of third iteration, perform decimation using the rule \( \beta \) for every node \( v_i \in V \).
4. Restart the decoder by setting all the messages to zero.
5. Repeat step 3) for nodes whose \( \beta_i = 0 \), followed by 4) until \( N_d \) decimations have been carried out.
6. Run the decoder for the remainder of iterations using maps \( \Phi^D_v \) and \( \Phi_c \).

Note that restarting the decoder implies, that the decimated nodes \( v_i \) will send \( \beta_i L_3 \) and the non-decimated nodes \( v_j \) will send \( \Phi_v(0, 0, y_j) \).

Remarks: 1) The reasons for choice of three iterations to start the decimation are as follows. First, there should exist messages in the graph that have a magnitude of \( L_3 \) in order to have a reliable decimation process. Secondly and more importantly, decimating only after three iterations makes the algorithm much more amenable to analysis. It becomes possible to analyze whether a particular node will be decimated or not and derive explicit conditions on the graph under which the nodes in error will not get decimated to the wrong value for a particular configuration. We shall in fact derive such conditions for the previous example. 2) Restarting the decoder after each decimation simplifies analysis.

C. Design of decimation rule \( \beta \)

The design of the rule \( \beta \) can be considered as a procedure of selecting the sets of incoming messages to node \( v_i \) for which \( \beta_i = \pm 1 \). We would like to do the selection with particular focus on correcting small number of errors typically associated with trapping sets in the error floor region. Referring back to the previous example, a good decimation rule would be one where \( \beta_i \) for most or all nodes \( v_j \in N(C^1 \cup C^2) \setminus V' \) is 1 and \( \beta_i \) for nodes \( v_i \in V' \) is 0 or 1, at the end of all decimations. We will now describe a good decimation rule selected for
the particular 7-level FAID whose LUT is shown in Table II. Before we describe the rule, we highlight two main points to be considered during the selection.

Firstly, while considering a particular set of incoming messages, the magnitudes of the incoming messages should play a role in the selection.

Secondly, the inherent structure of the particular $\Phi_v$ used in the decoder must be taken into consideration during selection. For this, we need to look at what outgoing messages a variable node would send for that particular set of incoming messages, and then decide if this set is good to select for decimation. For example, if a variable node $v_i$ whose channel value is $+C$, receives $-L_2,-L_3,-L_2$, this set might seem to be a possible candidate (to decimate $v_i$ to 1). However, the outgoing messages will be $-L_3,-L_1,-L_3$, which perhaps indicates that this may not be a reliable node to decimate since all outgoing messages are not $-L_3$ or even $-L_2$.

Table II shows all possible distinct sets of incoming messages with $y_i = +C$ for which $\beta_i = 1$. Using the symmetry of $\beta$, we can derive the sets of incoming messages with $y_i = -C$, for which $\beta_i = -1$. Note that an important condition that was used in defining the rule $\beta$, was that there must be a strict majority of messages between all the messages coming to a node $v_i$ and channel value $y_i$, and the majority sign must match with the sign of $y_i$, in order for $v_i$ to be decimated.

IV. Analysis of Decimation-Enhanced FAID Algorithms

A. Theoretical results

We first state the following lemma which is obtained due to the conditions used for decimation.

Lemma 1: The decimation-enhanced FAID algorithm will never decimate a node initially correct to a wrong value, and a node initially wrong to a correct value.

Proof: By virtue of $\beta$ that requires strict majority of signs of messages between incoming messages and $y_i$, and the majority sign matching sign of $y_i$. Remark: This simplifies the analysis as we need to only be concerned about decimation of nodes that are initially in error. At the same time, note that decimation alone can never correct errors.

Corollary 1: As a consequence of Lemma 1, the only necessary condition for success of a decimation-enhanced multilevel decoder is that a node initially in error must not be decimated.

Lemma 2: Given an error pattern, if no node initially in error gets decimated at the end of third iteration, then a node initially in error will never get decimated in the subsequent iterations.

Proof: (Details omitted) By restarting the decoder after each decimation, and by virtue of $\beta$, a node $v_i$ initially in error will not receive the required messages for $\beta_i = -1$.

Now for a particular error configuration in a graph $G$, we can analyze the conditions under which a node that is initially in error is decimated at the end of third iteration and then use Lemma 2. We can then place conditions on the graph such that the node in error is never decimated. To show this, we revert back to the example of the 4-error configuration whose induced subgraph forms an 8-cycle, and provide such conditions in the following theorem. Note that the proof will involve using Tables I and II and also the same notations previously defined in the example.

Theorem 1: Consider the 4-error pattern contained in a graph $G$, whose induced subgraph forms an 8-cycle. Also consider the decimation-enhanced 7-level FAID for decoding on this error pattern. If the graph $G$ has girth-8, and no three check nodes of the 8-cycle share a common variable node, then the nodes initially in error will not be decimated by this decoder in any iteration.

Proof: Firstly note that by virtue of $\Phi_v$ of the 7-level FAID (Table I), the highest magnitude of a message that any node $v_i \in V$ can send is $L_1$ in the first iteration and $L_2$ in the second iteration. Since a node $v_j \in N(C \cup C^2) \setminus V'$ can be connected to at most two checks in subgraph, the node $v_j$ in the worst case receives two $-L_1$ messages from checks in $C^1 \cup C^2$ and $L_1$ from outside at the end of first iteration. Node $v_i \in V'$ will also receive two $-L_1$ messages from check nodes in $C^2$ and $L_1$ from $c_k \in C^1 \cap N(v_i)$. At the end of the second iteration, the node $v_i \in V'$ will once again receive two $-L_1$ from checks in $C^2$, and $L_1$ from $c_k \in C^1$. This means that node $v_i$ will receive two $-L_1$ messages once gain from checks in $C^2$ at the end of third iteration. In order for it to be decimated, from Table II it must receive $-L_3$ from $c_k \in C^1 \cap N(v_i)$. This means that the node $v_j$ in the worst case has to receive at least one $-L_3$ at the end of the second iteration, but this is not possible by virtue of $\Phi_v$ in the second iteration. Hence, a node initially in error can not get decimated at the end of third iteration and using Lemma 2, will never get decimated.

Remarks: Note that the above condition is easily satisfied in most practical codes. This implies that on most practical codes, 4 errors on an 8-cycle will not be decimated.

Similarly, we can analyze under what conditions a node initially correct is decimated. For example, we may be able to derive conditions on the neighbors of the 4-error configuration such that they get decimated. In this manner, we can link analytical results of decimation to guaranteed error-correction capability.

Given that nodes initially in error are not decimated, it would be interesting to know whether the decimation-enhanced FAID can perform as good as multilevel FAID. In other words, given the necessary condition is satisfied, if a 7-level decoder corrects a particular $K$-error pattern,
The decimation-enhanced FAID was run using error rate curves for various decoders are shown in Fig. 2. The performance of the decimation-enhanced FAID. The frame-error rate performance comparison of Belief Propagation (BP), Finite Alphabet Iterative Decoder (FAID) and Decimation-enhanced FAID (DFAID) on the (155,93) Tanner code.

**Theorem 2:** Let there be a $K$-error pattern on a code $C$ with a Tanner graph $G$, whose graph satisfies conditions such that nodes initially in error are not decimated under the decimation-enhanced FAID. Assume that the multilevel FAID corrects this error pattern in $I$ iterations. If under the decimation-enhanced FAID algorithm, the decimation of a particular node does not lead to a bad message being sent from one of its adjacent check nodes in any iteration, then the decimation-enhanced FAID algorithm is also guaranteed to correct the pattern in atmost $I$ iterations.

**Proof:** Due to page constraints, proof is omitted. But the idea of the proof involves analyzing how messages flowing along the edges of the computation tree towards the root under the FAID algorithm, are affected by using the decimation-enhanced FAID algorithm.

**Remark:** Although Theorem 2 includes a specific condition on the decimation (under the assumption that nodes initially in error are not decimated), such a condition typically occurs for larger errors and decoding with much larger number of iterations. Therefore on small number of errors in trapping sets with a smartly chosen decimation rule, if nodes initially in error are not decimated, then decimation-enhanced FAID will most likely correct the pattern.

**B. Numerical Results and Discussion**

In this subsection, we present numerical results on the well-known (155,93) Tanner code in order to evaluate the performance of the decimation-enhanced FAID. The frame-error rate curves for various decoders are shown in Fig. 2. The decimation-enhanced FAID was run using $N_d = 4$ and all decoders used a maximum of 100 iterations. Note that the decimation-enhanced FAID was designed primarily to correct a fixed number of errors (in this case 5 errors) in fewer iterations compared to 7-level FAID. On the Tanner code, with $N_d = 1$, the decimation-enhanced FAID corrects all 5 errors within 10 iterations (after decimation) whereas the 7-level FAID requires 15 iterations. At the same time, we see that decimation-enhanced FAID performs just as good as the 7-level FAID (which was known beforehand to surpass BP), if not better.

We conclude by mentioning that our main goal was to provide a simple decimation scheme for multilevel FAID decoders that allows us to analyze their behaviour while maintaining good performance. From the theoretical analysis, we see that the role of decimation is important not just in improving the decoder performance or reducing the decoder speed but more so in terms of increasing the feasibility to obtain provable statements on the performance MP decoders such as FAID that are known to be empirically good. We finally remark that with more sophisticated versions of decimation such as use of multiple decimation rules, it might be possible to obtain an even more significant performance improvement.

**ACKNOWLEDGMENT**

This work was funded by NSF under the grants CCF-0830245 and CCF-0963726, and by the NANO2012 project.

**REFERENCES**

[1] S. K. Planjery, D. Declercq, S. K. Chilappagari, and B. Vasic, “Multilevel decoders surpassing belief propagation on the binary symmetric channel,” Proc. Int. Symp. on Inform. Theory, pp. 769–773, Jul. 2010.
[2] R. G. Gallager, Low Density Parity Check Codes. Cambridge, MA: M.I.T. Press, 1963.
[3] T. Richardson and R. Urbanke, “Capacity of low-density parity-check codes under message-passing decoding,” IEEE Trans. Inf. Theory, vol. 47, pp. 599–618, Feb. 2001.
[4] N. Wiberg, Codes and Decoding on General Graphs. PhD thesis, Linkoping University, Sweden, 1996.
[5] C. A. Kelly and D. Srithara, “Pseudocodewords of Tanner graphs,” IEEE Trans. Inf. Theory, vol. 53, pp. 599–618, Feb. 2001.
[6] P. Vontobel and R. Koetter, “Graph-Cover Decoding and Finite-Length Analysis of Message-Passing Iterative Decoding of LDPC Codes,” [http://arxiv.org/abs/cs/0512078](http://arxiv.org/abs/cs/0512078).
[7] C. Di, D. Proietti, I. E. Telatar, T. J. Richardson, and R. L. Urbanke, “Finite-length analysis of low-density parity-check codes on the binary erasure channel,” IEEE Trans. on Inform. Theory, vol. IT48, no. 6, pp. 1570–1579, 2002.
[8] T. Richardson, “Error floors of LDPC codes,” in Proc. 41st Annual Allerton Conf. on Commun., Control and Computing, 2003.
[9] V. Chervyak, M. Chertkov, M. Stepanov, and B. Vasic, Instanton method of posterior-correction analytical evaluation, in Proc. IEEE Inform. Theory Workshop, pp. 220-224, Oct. 2004.
[10] D. Burshtein and G. Miller, “Expander arguments for message-passing algorithms,” IEEE Trans. Inf. Theory, vol. 47, Issue 2, pp. 782–790, Feb. 2001.
[11] M. Ivkovic, S. K. Chilappagari, and B. Vasic, “Eliminating trapping sets in low-density parity-check codes by using Tanner graph covers,” IEEE Trans. on Inform. Theory, vol. 54, no. 8, pp. 3763–3768, 2008.
[12] D. Declercq, L. Danjean, E. Li, S. Planjery, and B. Vasic, “Finite alphabet iterative decoding (FAID) of the (155,64,20) Tanner code,” in Proc. 6th Int. Symp. on Turbo codes (ISTC’10), Sept. 2010.
[13] A. Montanari, F. Ricci-Tersenghi, and G. Semerjian, Solving constraint satisfaction problems through belief propagation-guided decimation, in Proc. Allerton Conf. on Commun., 2007.
[14] A. G. Dimakis, A. A. Gohari, M. J. Wainwright, “Guessing Facets: Polytope Structure and Improved LP Decoder,” [IEEE Trans. Inf. Theory](http://ieeexplore.ieee.org/document/5580162/), vol. 55, Issue 8, pp. 3479–3487, Aug. 2009.
[15] M. Chertkov, “Reducing the error floor,” in *Proc. IEEE Inf. Theory Workshop*, pp. 230–235, Sept. 2007.