Abstract—A low-density parity-check (LDPC) code is a linear block code described by a sparse parity-check matrix, which can be efficiently represented by a bipartite Tanner graph. The standard iterative decoding algorithm, known as belief propagation, passes messages along the edges of this Tanner graph. Density evolution is an efficient method to analyze the performance of the belief propagation decoding algorithm for a particular LDPC code ensemble, enabling the determination of a decoding threshold. The basic problem addressed in this work is how to optimize the Tanner graph so that the decoding threshold is as large as possible. We introduce a new code optimization technique which involves the search space range which can be thought of as minimizing randomness in differential evolution or limiting the search range in exhaustive search. This technique is applied to the design of good irregular LDPC codes and multi-edge type LDPC codes.

I. INTRODUCTION

The key objective of digital communications is to transmit information reliably from one point to another. With the introduction of iterative error correction codes (such as turbo, low-density parity-check and repeat-accumulate codes), error correction technology has become a vital means of achieving this aim in most current communication systems. A key performance measure of a coding scheme is its decoding threshold, which is the maximum noise level at which it can correct errors. In this paper, we design an efficient optimization technique to maximize the threshold of low-density parity-check (LDPC) codes.

In [1]–[3] a numerical technique, called Density Evolution (DE) was formulated to find the threshold of the belief propagation (BP) decoding algorithm for a given LDPC ensemble. An LDPC ensemble is the set of all LDPC codes with a particular property set, usually the degree distribution of their graphical (Tanner graph) representation. DE determines expected iterative decoding performance of a particular code ensemble by tracking the probability density function of Tanner graph edge messages through the iterative decoding process. This problem for the code designer is then to search for the ensemble with the best threshold from which a specific code may then be chosen.

Multi-edge type LDPC (MET-LDPC) codes [4] are a generalization of LDPC codes. Unlike standard LDPC ensembles which contain a single statical equivalence class of Tanner graph edges, in the multi-edge setting several edge classes can be defined and every node is characterized by the number of connections to edges of each class. The advantage of the MET generalization is greater flexibility in code structure and improved decoding performances.

The code optimization of LDPC and MET-LDPC codes is a non-linear cost function maximization problem, where the DE threshold is the cost function and the Tanner graph structure and edge distribution gives the variables to be optimized. In the majority of previous research in code optimization found in the literature, the optimization algorithm called Differential Evolution (Di.E) has been applied to finding good degree distributions for LDPC codes. This technique has been successfully applied to the design of good irregular LDPC codes for a range of channels [5]. Shokrollahi and Sorn [6] used an improved version of Di.E by proposing a new step called Discrete Recombination in order to increase the diversity of the new parameters in the search. Richardson and Urbanke [4] suggested using hill-climbing method to optimize MET-LDPC codes. In our work, we develop a new code optimization technique to optimize codes more efficiently. This technique can be thought of as minimizing the randomness in Di.E, [5], [6] or limiting the search space in ordinary exhaustive search and hill-climbing. This technique is then successfully applied to design good irregular LDPC codes and MET-LDPC codes.

In previous research of code optimization [4]–[6], the structure of the LDPC and MET-LDPC Tanner graph is determined via trial and error or exhaustive search, while only the edge distributions within a given structure are optimized. In this research, we propose a new nested method to optimize both the structure and edge distribution for LDPC and MET-LDPC codes. This is particularly important for MET-LDPC codes where, to date, it is not clear a priori which structures will be good.

This paper is organized as follows. Section II briefly reviews the basic concepts of standard LDPC codes and MET-LDPC codes. In section III we review the code optimization problem for standard and MET-LDPC codes and discuss our proposed code optimization technique. In Section IV we discuss the code optimization result obtained for several examples. Section V concludes the paper.

II. LOW-DENSITY PARITY-CHECK (LDPC) CODES

A. Standard LDPC codes

As the name suggests, an LDPC code is a linear block code described by a sparse parity-check matrix. An LDPC parity-check matrix can be represented in graphical form by a Tanner graph. Suppose the LDPC parity-check matrix, $H$ has $N$ columns and $M$ rows; the corresponding Tanner graph consists of $N$ variable nodes, $M$ check nodes, and an edge for every non-zero entry in $H$. Each variable node represents
The MET-LDPC code ensemble can be specified through two multi-edge degree and multinationals associated to the variable and check nodes [7].

Traditionally, the optimization problem is considered for a fixed code rate, r and maximum number of decoder iterations, l, where r ∈ [0, 1] and l ∈ {1, 2, . . .}. Max $\epsilon^*$, $0 \leq \epsilon^* \leq 1$

Subject to: $f(\epsilon^*, l, \lambda, \rho) = 0$ where $f(\epsilon^*, l, \lambda, \rho)$ is a recursive function described by $f(\epsilon^*, l, \lambda, \rho) = \epsilon^*(l) = \epsilon^*(1 - \rho(1 - \epsilon^{(l-1)}))$ (5)

$\epsilon^*(0) = \epsilon^*$ (6)

Variables: $\lambda, \rho$ where $\lambda(x)$ and $\rho(x)$ are given by (1) and (2) respectively.

Constraints:

$\sum_{i=2}^{d_{(v\text{, max})}} \lambda_i = 1$ (7)

$\sum_{i=2}^{d_{(c\text{, max})}} \rho_i = 1$ (8)

$\sum_{i=2}^{\lambda(\Gamma)} \rho_i/i = 1 - r$ (9)

$\lambda(0)/\rho(1) < 1/\epsilon^*$ (10)

where $\lambda'(0) = \frac{d}{dx}\lambda(x)|_{x=0}$ and $\rho'(1) = \frac{d}{dx}\rho(x)|_{x=1}$

Traditionally, the optimization problem is considered for a fixed $\Lambda$ and $\Gamma$ chosen via trail and error or intuition. That is the allowed degrees (for which $\lambda_i$ and $\rho_i$ are non-zero) are fixed in advance. Here we include the allowed degrees as variables in the optimization. This allows later generalization to MET-LDPC codes where the choice of $\Lambda$ and $\Gamma$ is not straightforward.

In this case we add constraints to restrict the number of non-zero entries in $\lambda(x)$ and $\rho(x)$ and their maximum range as follows:

$|\Lambda| \leq \Lambda_{\text{max}}$ (11)

$|\Gamma| \leq \Gamma_{\text{max}}$ (12)

Max($\Lambda$) $\leq d_{(v\text{, max})}$ (13)

Max($\Gamma$) $\leq d_{(c\text{, max})}$ (14)
B. Problem statement for MET-LDPC codes

Here we show the problem statement for a MET-LDPC code. In this case $\epsilon^{(l)}$ is a vector $\epsilon^{(l)} = [\epsilon_1^{(l)}, \ldots, \epsilon_{m_e}^{(l)}]$ where $m_e$ is the number of edge classes and $\epsilon_i^{(0)} = \epsilon^*$ for all $i$.

For a fixed code rate, $r$ and maximum number of decoder iterations, $l$, where $r \in [0,1]$ and $l \in \{1, 2, \ldots\}$.

Max $\epsilon^*$, $0 \leq \epsilon^* \leq 1$
Subject to: $f(\epsilon^{(0)}, l, \lambda, \rho) = 0$
where $f(\epsilon^{(0)}, l, \lambda, \rho)$ is a recursive function described by

\[
f(\epsilon^{(l)}, l, \lambda, \rho) = \epsilon^{(l)} = \lambda(\epsilon^{(0)}, \rho(\epsilon^{(l-1)}))
\]

where

\[
\lambda(r, x) = \left(\frac{L_{x_1}(r, x)}{L_{x_1}(1, 1)}, \ldots, \frac{L_{m_e}(r, x)}{L_{m_e}(1, 1)}\right)
\]

\[
\rho(x) = \left(\frac{R_{x_1}(x)}{R_{x_1}(1)}, \ldots, \frac{R_{m_e}(x)}{R_{m_e}(1)}\right)
\]

Constraints:

\[
L_{x_i}(r, x) = \frac{d}{d_{x_i}} L(r, x)
\]

\[
R_{x_i}(x) = \frac{d}{d_{x_i}} R(x)
\]

\[
L_b(1, 1) = 1
\]

\[
L(1, 1) - R(1) = r
\]

\[
L_{x_i}(1, 1) = R_{x_i}(1)
\]

where $L(r, x)$ and $R(x)$ are given by [5] and [4] respectively and 1 denotes a vector of all 1’s with the length determined by the context.

We will consider these optimization problems using a range of existing non-convex optimization methods and a proposed method.

C. Proposed Code optimization Algorithm: Adaptive Range Method

The Adaptive Range (AR) method forms the next set of points for evaluation as a random selection of the set of points $C$. Proposed Code optimization Algorithm: Adaptive Range (AR) method forms the next set of existing non-convex optimization methods and a proposed method.

We will consider these optimization problems using a range of existing non-convex optimization methods and a proposed method.

C. Proposed Code optimization Algorithm: Adaptive Range Method

The Adaptive Range (AR) method forms the next set of points for evaluation as a random selection of the set of points $C$. Proposed Code optimization Algorithm: Adaptive Range (AR) method forms the next set of existing non-convex optimization methods and a proposed method.

We will consider these optimization problems using a range of existing non-convex optimization methods and a proposed method.

C. Proposed Code optimization Algorithm: Adaptive Range Method

The Adaptive Range (AR) method forms the next set of points for evaluation as a random selection of the set of points $C$. Proposed Code optimization Algorithm: Adaptive Range (AR) method forms the next set of existing non-convex optimization methods and a proposed method.

We will consider these optimization problems using a range of existing non-convex optimization methods and a proposed method.

C. Proposed Code optimization Algorithm: Adaptive Range Method

The Adaptive Range (AR) method forms the next set of points for evaluation as a random selection of the set of points $C$. Proposed Code optimization Algorithm: Adaptive Range (AR) method forms the next set of existing non-convex optimization methods and a proposed method.

We will consider these optimization problems using a range of existing non-convex optimization methods and a proposed method.

C. Proposed Code optimization Algorithm: Adaptive Range Method

The Adaptive Range (AR) method forms the next set of points for evaluation as a random selection of the set of points $C$. Proposed Code optimization Algorithm: Adaptive Range (AR) method forms the next set of existing non-convex optimization methods and a proposed method.

We will consider these optimization problems using a range of existing non-convex optimization methods and a proposed method.

C. Proposed Code optimization Algorithm: Adaptive Range Method

The Adaptive Range (AR) method forms the next set of points for evaluation as a random selection of the set of points $C$. Proposed Code optimization Algorithm: Adaptive Range (AR) method forms the next set of existing non-convex optimization methods and a proposed method.

We will consider these optimization problems using a range of existing non-convex optimization methods and a proposed method.

C. Proposed Code optimization Algorithm: Adaptive Range Method

The Adaptive Range (AR) method forms the next set of points for evaluation as a random selection of the set of points $C$. Proposed Code optimization Algorithm: Adaptive Range (AR) method forms the next set of existing non-convex optimization methods and a proposed method.

We will consider these optimization problems using a range of existing non-convex optimization methods and a proposed method.

C. Proposed Code optimization Algorithm: Adaptive Range Method

The Adaptive Range (AR) method forms the next set of points for evaluation as a random selection of the set of points $C$. Proposed Code optimization Algorithm: Adaptive Range (AR) method forms the next set of existing non-convex optimization methods and a proposed method.

We will consider these optimization problems using a range of existing non-convex optimization methods and a proposed method.
10 trails. AR and Dif.E algorithms return the same threshold was selected for RM and SR (initial setting) respectively. When including the allowed degrees in the optimization, the $\lambda$ with an inner optimization of AR for the ($\Lambda, \rho$) set as found in [6]. Thus blind optimization returns the same value with an accuracy of three significant decimal points for the same ($\Lambda, \rho$), although the ($\lambda, \rho$) sets are slightly different. This is because there can be more than one ($\lambda, \rho$) set for each ($\Lambda, \rho$) that obtains the same threshold when it is evaluated within three significant figures.

### B. Generalization of degree distribution

Next, we jointly optimized ($\Lambda, \Gamma$) and ($\lambda, \rho$) using the AR method, Dif.E and Random Search (RS) for the ($\Lambda, \Gamma$) variables with an inner optimization of AR for the ($\lambda, \rho$) variables. When including the allowed degrees in the optimization, the task now is to solve an integer optimization, as the node degrees can only be integer values. In the AR method, optimization of ($\lambda, \rho$) was done using RM and SR (initial setting) as described in section IV.A. For optimization of ($\Lambda, \Gamma$), 1 and 15 was selected for RM and SR (initial setting) respectively. To compare the three optimization algorithms, we consider LDPC codes with rate-half, $|A| = 4, \max(A) = 30$ and constraint on the maximum bit node degree of $d_{v, \text{max}} = 30$. In all cases $\lambda_1, \rho_1$ were initialized with the Queens move method. The last three columns of Table I show our results. The combined optimization using either AR or Dif.E returns the same ($\Lambda, \Gamma$) set as found in [6]. Thus blind optimization returns the same degree set found in [6] via intuition and trail and error. The AR method was significantly faster than the Dif.E method and the RS method failed to achieve the best degree combination in all 10 trails. AR and Dif.E algorithms return the same threshold value with an accuracy of three significant decimal points for the same ($\Lambda, \Gamma$), although the ($\lambda, \rho$) sets are slightly different. This is because there can be more than one ($\lambda, \rho$) set for each ($\Lambda, \rho$) that obtains the same threshold when it is evaluated within three significant figures.

### V. Optimization of MET-LDPC Codes

In this section we apply our optimization techniques to MET-LDPC ensembles where the combined optimization technique is particularly useful to explore the wide range of possible structures.

Similarly to standard LDPC codes we found that, for the inner optimization (i.e., optimizing the node fractions, denoted as $a_i$ and $c_j$ in Fig. 3 for a fixed code structure, denoted as $i^c_1$ and $j^c_2$ in the same figure), the AR method returns a higher optimal threshold than Dif.E. The MET-DE cost function when allowed degrees are fixed, has a global maximum and no other local maxima. An example of this cost function for the 4-edge class MET-LDPC code given in Table VI of [4] on a BI-AWGN channel is shown in Fig. 3.

### A. Generalization of degree distribution

To examine the properties of the MET-DE cost function for the generalization of the MET-LDPC code structure, we consider MET-LDPC codes with four variable node classes, maximum allowable variable node degree of 10 and edge class 4 containing degree-1 variable nodes as in the example in [3].

---

**Table I**

The degree distributions found using 3 different optimization techniques for rate-half codes considered in [6].

| $|A|$ | AR | Dif.E | Dif.E.R | AR | Dif.E | Dif.E.R | AR | Dif.E | Dif.E.R | AR | Dif.E | Dif.E.R |
|------|----|------|--------|----|------|--------|----|------|--------|----|------|--------|
| 2, 3, 6, 20 | 0.2962 | 0.2985 | 0.2987 | 0.2774 | 0.2750 | 0.2770 | 0.2621 | 0.2630 | 0.2636 | 0.2610 | 0.2672 | 0.2860 |
| 2, 3, 7, 25 | 0.1749 | 0.1740 | 0.1741 | 0.2020 | 0.2040 | 0.2025 | 0.1816 | 0.1810 | 0.1801 | 0.1832 | 0.1758 | - |
| 2, 3, 7, 30 | - | - | - | - | - | - | - | - | - | - | 0.3326 | - |
| 4 | 0.2418 | 0.2485 | 0.2489 | - | - | - | - | - | - | - | - | 0.1834 |
| 7 | - | - | - | 0.2626 | 0.2560 | 0.2610 | 0.2670 | 0.2690 | 0.2706 | 0.2640 | 0.2797 | - |
| 13 | - | - | - | - | - | - | - | - | - | - | - | 0.1834 |
| 20 | 0.2872 | 0.2790 | 0.2784 | - | - | - | - | - | - | - | - | 0.1834 |
| 25 | - | - | - | 0.2580 | 0.2650 | 0.2595 | - | - | - | - | - | 0.1834 |
| 30 | 0.3094 | 0.3533 | 0.3571 | 0.1083 | 0.0748 | 0.1026 | 0.2893 | 0.2870 | 0.2856 | 0.2918 | 0.2722 | 0.1980 |
| $\rho_7$ | 0.6976 | 0.6467 | 0.6429 | 0.8917 | 0.9252 | 0.8974 | 0.6171 | 0.6338 | 0.6418 | 0.6036 | 0.6912 | 0.8872 |
| $\rho_8$ | 0.6976 | 0.6467 | 0.6429 | 0.8917 | 0.9252 | 0.8974 | 0.6171 | 0.6338 | 0.6418 | 0.6036 | 0.6912 | 0.8872 |

- $c^*$ - Ensemble Threshold, NOG - No of Generations before algorithm halted, NTT - Number of times we check the degree distribution at best threshold so far (Number of candidates considered), NFE - Number of times we find a new Threshold (New degree distribution improves all previous best), CPU.T - CPU time in seconds, AVG - Mean value, SD - Standard deviation

which is a type of local search optimization technique works well in the absence of multiple local maxima.

### TABLE I

The degree distributions found using 3 different optimization techniques for rate-half codes considered in [6].
The MET-DE cost function for this situation has local maxima. Fig. 4 shows an example of this cost function as the node degree of the class 1 and class 2 variable nodes are varied. In this scenario Dif.E outperforms the AR method by escaping local maxima which is reflected in the performance of each algorithm as shown in Table II. Hence we propose that, MET-LDPC codes be optimized using the AR method as the MET degree distribution optimization technique and Dif.E to optimize the MET code structure. Through the joint optimization we gain a threshold improvement of 0.007 compared to the reference MET-LDPC code (Table VI of [4]) which has a threshold of $\sigma^* = 0.9682$ over the BI-AWGN channel. Here we see the benefit of MET-LDPC codes as standard LDPC codes require a maximum node degree of 85 to obtain a threshold this high [8].

VI. CONCLUSION

In this paper, we introduced a novel code optimization technique which can be successfully applied to optimize degree distributions for both standard LDPC codes and MET-LDPC codes. We then proposed a joint optimization technique for MET-LDPC codes which allows the optimization of the MET structure and degree distribution given just the number of edge classes and maximum node degrees. We found that our proposed AR method works best for optimizing the edge degree distribution for a given set of allowed degrees while Dif.E works best for optimizing the set of allowed degrees.

REFERENCES

[1] T. Richardson and R. Urbanke, “The capacity of low-density parity-check codes under message-passing decoding,” Information Theory, IEEE Transactions on, vol. 47, no. 2, pp. 599–618, Feb 2001.
[2] R. Gallager, “Low density parity check codes,” 1963.
[3] M. G. Luby, M. Mitzenmacher, M. A. Shokrollahi, and D. A. Spielman, “Improved low-density parity-check codes using irregular graphs,” in Proc. IEEE Symposium on Information Theory, 1999, p. 117.
[4] T. Richardson, R. Urbanke et al., “Multi-edge type LDPC codes,” in Workshop honoring Prof. Bob McEliece on his 60th birthday, California Institute of Technology, Pasadena, California, 2002.
[5] T. J. Richardson, M. A. Shokrollahi, and R. L. Urbanke, “Design of capacity-approaching irregular low-density parity-check codes,” IEEE Trans. Inform. Theory, vol. 47, no. 2, pp. 619–637, Feb. 2001.
[6] A. Shokrollahi and R. Storn, Design of efficient erasure codes with differential evolution. Springer, 2005, pp. 413–427.
[7] T. Richardson and R. L. Urbanke, Modern Coding Theory. Cambridge University Press, 2008.
[8] S.-Y. Chung, “On the construction of some capacity-approaching coding schemes,” Ph.D. dissertation, MIT, Cambridge, MA, 2000.