This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE ICC 2009 proceedings

Design and Analysis of E^2RC Codes Using EXIT Chart

Cuizhu Shi and Aditya Ramamoorthy

Department of Electrical and Computer Engineering, Iowa State University, Ames, Iowa 50011
Email: {cshi, adityar}@iastate.edu

***Abstract*—We present the design and analysis of a family of codes based on the efficiently-encodable rate-compatible irregular LDPC codes (E^2RC codes) introduced by Kim, Ramamoorthy and Mclaughlin '06. The basic idea is to utilize the structured parity part (of the parity check matrix) of E^2RC codes and design optimal degree distributions for the systematic part based on EXIT chart analysis. In this work, we propose a new technique for computing EXIT functions of the structured parity part without resorting to Monte Carlo simulations. Our method provides smoother EXIT functions in substantially lesser time. Furthermore, we pose and solve the problem of designing good codes using linear programming. Next, we consider the performance of rate-compatible codes under puncturing using EXIT charts. Finally, we propose joint optimization of our codes at any specified code rate(s) which has not been explored in previous design of rate-compatible punctured codes.**

I. Introduction

Low-density parity-check (LDPC) codes [1] have found widespread acceptance in different areas, due to their superior performance and low complexity decoding. Among them, rate-compatible punctured LDPC codes have the flexibility of operating at different code rates while having a single encoder-decoder pair. Rate-compatible punctured codes are defined by specifying a systematic mother code that operates at the lowest code rate. The parity bits of the higher rate codes in a rate-compatible code family are subsets of those of lower rate codes. A number of papers have investigated issues around the design of good rate-compatible punctured LDPC codes. The work of [2] presents methods for finding optimal degree distributions for puncturing. In [3][4][5], algorithms for finding good puncturing patterns for a given mother code were proposed. There have also been attempts to design mother codes (along with puncturing patterns) with good performance under puncturing [6][7].

E^2RC codes introduced in [6] are linear-time encodable and have good puncturing performance across a wide range of rates at short block lengths. In this work we present the systematic design and analysis of semi-structured E^2RC-like codes using EXIT charts [8]. Let $H = [H_1|H_2]$ denote the parity check matrix of a systematic LDPC code where H_1 denotes the systematic part and H_2 the parity part. The H_2 part of semi-structured E^2RC codes is deterministic, following the lower triangular form introduced in [6]. We assume a random edge interleaver between systematic variable nodes and check nodes, which divides the code into a structured part and an unstructured part, as shown in Fig. 1. We solve the problem of finding optimal degree distributions for the unstructured part using EXIT charts so that the code can be optimized at any specified code rate(s). Our joint optimization of rate-compatible punctured codes is different from previous designs of rate-compatible punctured codes. Our codes have uniformly good performance across the range of code rates compared with other approaches.

This paper is organized as follows. Background and related works are reviewed in Section II. We explain the problem formulation in Section III. Section IV presents a new method of computing EXIT functions of structured part of E^2RC codes without resorting to Monte Carlo simulations. The design of capacity-approaching E^2RC codes based on linear programming is discussed in Section V. We also analyze the puncturing performance of E^2RC codes and propose joint optimization of our codes at multiple code rates simultaneously. Section VI outlines our conclusions.

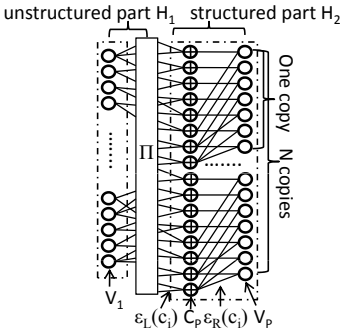

Fig. 1. Tanner graph representation of E^2RC codes.

II. Background and Related Work

A. Efficiently Encodable Rate-Compatible LDPC Codes

Here we briefly explain the construction of E^2RC codes introduced in [6] as our work is inspired by it. Let $H = [H_1|H_2]$ denote the parity-check matrix of E^2RC codes. We say that a parity node in H_2 is k-step recoverable (or k-SR) if it can be recovered in exactly k iterations of iterative decoding assuming that all the parity bits are punctured and all the information bits are known (Fig. 2 shows an example). Intuitively, a large number of low-SR nodes tend to reduce the required number of decoding iterations in the high SNR regime and result in good puncturing performance.

978-1-4244-3435-0/09/$25.00 ©2009 IEEE

In [6] the submatrix H_2 consists of exclusively degree-2 and degree-1 nodes. Moreover if the number of check nodes is a power of 2, then half the nodes in H_2 are 1-SR, one-fourth are 2-SR and so on. The special structure of H_2 for E^2RC codes results in good puncturing performance with a simple puncturing pattern, where lower-SR nodes should be punctured earlier.

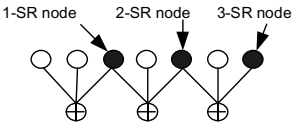

Fig. 2. The figure shows an example of 1-SR, 2-SR and 3-SR nodes.

B. EXIT Chart Overview

EXIT charts were first proposed for understanding the convergence behavior of iteratively decoded parallel concatenated codes [8] and were later generalized for the analysis of LDPC codes [9] [10] [11] [12]. The components of an EXIT chart are the EXIT functions of the constituent code components of the iterative decoder. The EXIT function relates the a priori mutual information available to a code component, denoted I_A and the extrinsic mutual information generated by it after decoding, denoted I_E. The advantage of EXIT charts is that the code design problem can be reduced to a curve fitting problem between the code components (usually two in number).

For log-domain belief propagation decoding of LDPC codes, if the incoming messages to a variable nodes v of degree d_v are assumed to be Gaussian and independent, we can write its EXIT function as (refer to [10] for more details)

$$I_{E,v}(I_{A,v}) = J(\sqrt{(d_v-1)[J^{-1}(I_{A,v})]^2 + \sigma^2_{ch,v}}), \quad (1)$$

where $\sigma^2_{ch,v} = 4/\sigma^2_n$ for unpunctured v (σ^2_n channel noise variance), $\sigma^2_{ch,v} = 0$ for punctured v and the function $J(\sigma)$ is given by

$$J(\sigma) = 1 - \int_{-\infty}^{+\infty} \frac{e^{-(\xi-\frac{\sigma^2}{2})^2}}{2\sigma^2 \cdot \sqrt{2\pi\sigma^2}} \log_2[1+e^{-\xi}]d\xi.$$

Similarly the EXIT function for a degree-d_c check node is given as

$$I_{E,c}(I_{A,c}) = 1 - J(\sqrt{(d_c-1)[J^{-1}(1-I_{A,c})]^2}). \quad (2)$$

III. Problem Formulation

In the original construction of E^2RC codes [6], an optimal degree distribution pair (λ, ρ) for unstructured irregular LDPC codes is used for constructing $H = [H_1|H_2]$ of mother code. H_2 is constructed according to the designed structure and H_1 is constructed by attempting to match the chosen pair (λ, ρ), i.e., the structure of H_2 is not taken into account in designing the optimal degree distribution. In this paper we design a new class of E^2RC-like codes, where the structure of H_2 is taken into account in the design process.

We denote the set of variable nodes by $V = V_1 \cup V_2$, where V_1 is the subset of nodes in the unstructured part and V_2 is the subset of nodes in the structured part. The check node set is denoted by C. In semi-structured E^2RC codes, H_2 has a base protograph structure of the form proposed in [6]. The base protograph is parameterized by the number of check nodes in it denoted by M. The H_2 of semi-structured E^2RC codes is obtained by simply replicating the base protograph an appropriate number of times. For example, the case of $M = 8$ in shown in Fig. 1. Check nodes are connected to the set V_1 by a random interleaver (denoted Π in Fig. 1). Also we don't use degree-1 variable nodes. We group the original last degree-1 parity node in H_2 with information nodes and allow it to have higher degree.

We shall frequently refer to the protograph representation of H_2. Denote the set of variable nodes in the protograph representation of H_2 by V_p, and the check nodes by C_p. Let $\varepsilon(v_i)$ and $\varepsilon(c_i)$ denote the set of edges connected to $v_i \in V_p$ and $c_i \in C_p$ respectively. We shall use $\varepsilon_L(c_i)$ to denote the set of edges connecting c_i and the random interleaver and $\varepsilon_R(c_i)$ the set of edges connecting c_i and V_p, i.e., $\varepsilon(c_i) = \varepsilon_L(c_i) \cup \varepsilon_R(c_i)$. Given a protograph structure on H_2, the problem of code design becomes one of finding good edge profiles for nodes in V_1 and the edges in $\cup_{c_i \in C_p} \varepsilon_L(c_i)$. In this paper, we present efficient techniques for solving this problem. Moreover we analyze the puncturing performance of the proposed codes using EXIT chart.

IV. EXIT Function Computation for the Structured Part

The usual approach for finding the EXIT function of a constituent code component has been to resort to Monte Carlo simulations. One generates random LLR's for the a priori inputs and channel inputs (if any) to the code component according to the distribution determined by I_A and the channel parameter, then performs decoding, and finally computes I_E (corresponding to I_A) based on the distribution of the extrinsic outputs. This process needs to be repeated for the range of I_A in order to obtain the whole EXIT function curve. Code design based on EXIT charts requires the EXIT functions to be smooth for accurate curve fitting, which implies that a large number of Monte Carlo simulations are needed which is rather time-consuming.

In this section we present a fast and accurate method of computing EXIT functions of structured code components of LDPC codes, such as the structured part of E^2RC codes and that of IRA codes, without resorting to Monte Carlo simulations. The method is explained in the context of the binary-input AWGN (BIAWGN) channel and can be adapted to the BEC and other channels in a straightforward manner.

For convenience, we use the notation $E_R = \cup_{v_i \in V_p} \varepsilon(v_i)$ and $E_L = \cup_{c_i \in C_p} \varepsilon_L(c_i)$. Note that $\cup_{v_i \in V_p} \varepsilon(v_i) = \cup_{c_i \in C_p} \varepsilon_R(c_i)$. Suppose that the a priori inputs carried on $e \in E_L$ have average mutual information $I_{A,in}$ and that $v \in V_p$ has channel inputs parameterized by $\sigma^2_{ch,v}$. We are interested in finding I_E, the average mutual information associated with the extrinsic outputs carried on $e \in E_L$ after iterative decoding. For an edge e connected to node $v_i(c_i)$, we shall use the

notation $I_{A,e}^{v_i}$(likewise $I_{A,e}^{c_i}$) to denote the mutual information describing the a priori inputs on it and $I_{E,e}^{v_i}$(likewise $I_{E,e}^{c_i}$) the mutual information describing the extrinsic outputs on it. We set up the following system of equations for the given structured code component. For $e \in E_R$ and $v_i \in V_p$, we have

$$I_{E,e}^{v_i} = J\left(\sqrt{\sum_{e' \in \varepsilon(v_i)\backslash\{e\}} [J^{-1}(I_{A,e'}^{v_i})]^2 + \sigma_{ch,v_i}^2}\right) \quad (3)$$

Similarly, for $e \in E_R$ and $c_i \in C_p$,

$$I_{E,e}^{c_i} = 1 - J\left(\left(\sum_{e' \in \varepsilon_L(c_i)} [J^{-1}(1 - I_{A,in})]^2 + \sum_{e' \in \varepsilon_R(c_i)\backslash\{e\}} [J^{-1}(1 - I_{A,e'}^{c_i})]^2\right)^{\frac{1}{2}}\right) \quad (4)$$

and for $e \in E_L$ and $c_i \in C_p$,

$$I_{E,e}^{c_i} = 1 - J\left(\left(\sum_{e' \in \varepsilon_L(c_i)\backslash\{e\}} [J^{-1}(1 - I_{A,in})]^2 + \sum_{e' \in \varepsilon_R(c_i)} [J^{-1}(1 - I_{A,e'}^{c_i})]^2\right)^{\frac{1}{2}}\right) \quad (5)$$

For each edge $e \in E_R$, there are two equations in the form of (3) and (4) respectively and two unknown variables $I_{E,e}^{v_i}$(or $I_{A,e}^{c_j}$), $I_{E,e}^{c_j}$(or $I_{A,e}^{v_i}$) associated with it; while for each edge $e \in E_L$, there is one equation in the form (5) and one unknown variable $I_{E,e}^{c_i}$ associated with it. We want to compute $I_{E,e}^{c_i}$ for $e \in E_L$. There are totally $2|E_R| + |E_L|$ equations and the same number of unknown variables involved in this system of equations. Note that the specific expressions for this system of equations are totally dependent on the structure of the code component. The main idea behind our method for computing the EXIT function is to find the solution to this system of equations for a given value of $I_{A,in}$ and channel parameter. We now present an intuitive method for solving this system of equations, which works in an iterative manner by applying the sequence of updates described in equations (3), (4) and (5). The details are given below.

1) *Problem Instance.* Given a structured code component, solve the system of equations described in (3), (4) and (5) above. The unknown variables involved in this system of equations are $I_{E,e}^{v_i}$(or $I_{A,e}^{c_j}$), $I_{E,e}^{c_j}$(or $I_{A,e}^{v_i}$) for all $e \in E_R$ and $I_{E,e}^{c_i}$ for all $e \in E_L$. The known variables are $I_{A,e}^{c_i} = I_{A,in}$ for all $e \in E_L$, and the channel parameter σ_n^2 from which σ_{ch,v_i}^2 can be determined for each v_i.

2) *Initialization.* Initialize all unknown variables to be 0. Set a small value of $\epsilon_{thresh} = 10^{-6}$.

3) *Iterative Updates.*

(a) *Check node update.* For all $e \in E_R$, compute $I_{E,e}^{c_i}$ using equation (4). Check to see whether the norm of the difference between this newly computed set of $I_{E,e}^{c_i}$ and the previously computed ones is smaller than ϵ_{thresh}. If yes, then terminate; otherwise, set $I_{A,e}^{v_j} = I_{E,e}^{c_i}$ if v_j and c_i are connected by e.

(b) *Variable node update.* For all $e \in E_R$, compute $I_{E,e}^{v_i}$ using equation (3). Set $I_{A,e}^{c_j} = I_{E,e}^{v_i}$ if c_j and v_i are connected by e. Go to step 3(a).

4) *Compute $I_{E,e}^{c_i}$ for $e \in E_L$.* For all $e \in E_L$, compute $I_{E,e}^{c_i}$ using equation (5). The average of these $I_{E,e}^{c_i}$ is denoted by I_E and $(I_{A,in}, I_E)$ is a point on the EXIT function.

The method can be adapted for computing EXIT functions over other channels by using appropriate update equations. Moreover, it can be used to compute the EXIT function of the structured part of other codes that have a succinct protograph representation such as IRA codes.

We demonstrate the effectiveness of our method by comparing the EXIT functions computed by our method and by Monte Carlo simulation based method for four cases, the structured part of E^2RC codes and that of IRA codes on BEC and BIAWGN channels respectively. The structured part of E^2RC code has a protograph structure of size 128. All check nodes have degree 8. As shown in Table I, the average error between the EXIT functions computed using our method and via Monte Carlo simulations is less than 2.7×10^{-3}. To obtain a smooth curve from Monte Carlo simulations, we used around 10^6 a priori inputs for each I_A. We expect the average error to reduce even more when more Monte Carlo simulations are used.

TABLE I
COMPARISON DETAILS FOR FOUR CASES

BEC: erasure probability= 0.5; 10^7 randomly generated a priori inputs used in simulations				
	E^2RC		IRA	
Method	Ours	Simulation	Ours	Simulation
Number of $\{(I_A, I_E)\}$ pairs	10^4	10^4	10^4	10^4
Computing time (s)	1.4	18745	0.5	167353
Average error	0.000114	-	0.000048	-
AWGN: noise variance= 0.95775;10^6 randomly generated a priori inputs used in simulations				
	E^2RC		IRA	
Method	Ours	Simulation	Ours	Simulation
Number of $\{(I_A, I_E)\}$ pairs	10^4	10^4	10^4	10^4
Computing time (s)	3.7	24596	0.6	175886
Average Error	0.00276	-	0.00268	-

V. Semi-Structured E^2RC Code Design

In the context of EXIT chart analysis, we consider the unstructured part and the structured part of E^2RC codes shown in Fig. 1 as two constituent code components. This code division for EXIT chart analysis is justified by the random edge interleaver between the two code components.

Our objective is to find good degree distributions for the variable nodes in V_1 and for edges in E_L, i.e. the left check degree distribution. In our work, we only consider concentrated or near-concentrated total check degrees. We have found experimentally that these tend to give the best performance.

A. Code Design by Linear Program

Suppose we are given channel parameter σ_n^2, the structured part including left check degrees. We want to find the degree distribution $\{\lambda_{d_v}, v \in V_1\}$ so as to minimize the gap between code rate R and channel capacity C (corresponding to σ_n^2). The EXIT function of the structured part, denoted by $I_{E,S}(I_{A,S})$, can be found based on our discussion in Section IV. For a given $\{\lambda_{d_v}, v \in V_1\}$, the EXIT function of the

unstructured part can be expressed as

$$I_{E,unS}(I_{A,unS}, \sigma_{ch}^2)$$

$$= \sum_{d_v} \lambda_{d_v} J(\sqrt{(d_v - 1)[J^{-1}(I_{A,unS})]^2 + \sigma_{ch,v}^2}). \quad (6)$$

The code design or optimization problem is formulated as:

$$\begin{aligned} \text{minimize:} \quad & C - R \\ \text{subject to:} \quad & 1.\ \textstyle\sum_{d_v=1}^{d_{v,max}} \lambda_{d_v} = 1, \lambda_{d_v} \geq 0, \\ & 2.\ I_{E,unS}(I_{A,unS}) > I_{A,S}(I_{E,S}), \\ & \quad \text{for } I_{A,unS} = I_{E,S} \in [0, 1). \end{aligned}$$

where $d_{v,max}$ is the maximum degree that is allowed for V_1. The second constraint is the zero-error constraint. Minimizing $C - R$ is equivalent to maximizing R, which is further equivalent to minimizing the average degree of nodes in V_1 (given by $1/\sum_{d_v=1}^{d_{v,max}} \frac{\lambda_{d_v}}{d_v}$) or maximizing $\sum_{d_v=1}^{d_{v,max}} \frac{\lambda_{d_v}}{d_v}$. For a given value of $I_{A,unS}$, $I_{E,unS}$ is linear in $\{\lambda_{d_v}, v \in V_1\}$. Therefore by a fine enough discretization, we can express the above optimization as a linear program. In practice, to set up the second constraint, we need to find the inverse map $I_{A,S}(I_{E,S})$ by using linear interpolation. We have found that a large number (say 10^4) of $(I_{A,S}, I_{E,S})$ pairs for the function $I_{E,S}(I_{A,S})$ are necessary to ensure accuracy of the solution to the optimization problem. Moreover usually we are interested in optimizing codes at a specific code rate, which requires solving the above optimization at closely spaced channel parameter levels below the Shannon limit. This necessitates the fast speed of our method for computing $I_{E,S}(I_{A,S})$ above.

B. Code Design Examples

We design a semi-structured E^2RC codes with concentrated check degree 8 and $d_{v,max} = 20$. The H_2 part has a protograph structure of size $M = 32$. The optimized code (referred to as code 1) is given by $\lambda_3 = 0.305825, \lambda_7 = 0.213474, \lambda_8 = 0.181737, \lambda_{20} = 0.298964$ for $v \in V_1$ and it has an asymptotic gap of 0.217 dB to capacity at rate 0.5. The simulation results of code 1 of block length 16384 bits are given in Fig. 3. Also given in Fig. 3 are the simulation results of the E^2RC code that is constructed according to the degree distribution specified in [6] (referred to as original E^2RC). It shows that our code achieves slightly better performance at rates near mother code rate but suffers a little at higher code rates.

To verify our design, we use the following terminology in this paper. The predicted threshold refers to the decoding threshold from asymptotic code performance analysis and the measured threshold refers to the channel parameter where the code achieves BER $= 10^{-4}$ in simulations. For our code 1 of length 16384 bits, the gap between the measured threshold and the predicted one at rate 0.5 is only 0.4 dB.

C. Puncturing Performance Analysis and Joint Optimization of Semi-Structured E^2RC Codes

The puncturing performance of a given code is specified in terms of its decoding thresholds at all punctured code rates. The given semi-structured E^2RC codes are specified by $\lambda_{d_v}, v \in V_1$ and the knowledge of the protograph structure of the structured part. For a given channel parameter, we can compute the two EXIT functions using (6) and the approach of Section IV. Note that when computing EXIT functions at different puncturing code rates, we follow the designed puncturing pattern of E^2RC codes. The decoding threshold at a given code rate is determined by finding the channel parameter where the two EXIT curves (computed under the puncturing pattern at that rate) just begin to separate.

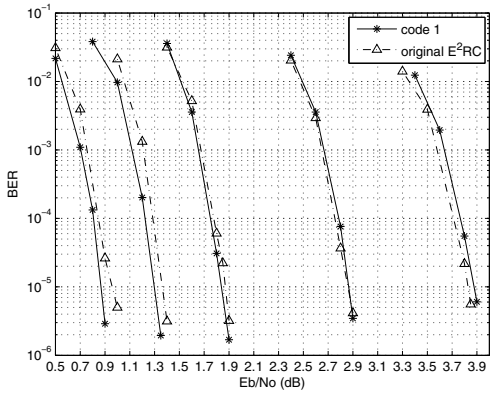

Fig. 3. Comparison between code 1 and original E^2RC code at code rates $0.5, 0.5714, 0.6667, 0.8$ and 0.8889 from left to right.

Our puncturing performance analysis of code 1 suggests that it has asymptotic decoding thresholds of around $0.40, 0.85, 1.40, 2.45, 3.44$ dB at rates $\frac{8}{16}, \frac{8}{14}, \frac{8}{12}, \frac{8}{10}$ and $\frac{8}{9}$ respectively. The measured thresholds at these rates for the code of block length 16384 bits based on the results in Fig. 3 are around $0.80, 1.22, 1.75, 2.78, 3.76$ dB, which are consistent with the predicted ones with gaps uniformly round 0.35 dB.

Joint Optimization of Semi-Structured E^2RC Codes

By exploring the E^2RC structure and its designed puncturing pattern, we can design rate-compatible punctured codes that are optimized at any specified code rate(s). Let $\mathbf{R}$ be a specified set of code rates where we want to optimize the code. Let $\sigma(g, R_i)$ denote the channel noise parameter that is at a gap of g from the channel parameter corresponding to the Shannon limit at rate R_i. Let $I_{A,S}(I_{E,S}, \sigma(g, R_i))$ denote the plot of $I_{A,S}$ vs. $I_{E,S}$ under the puncturing pattern corresponding to rate R_i, at the channel parameter $\sigma(g, R_i)$. The notation $I_{A,unS}(I_{A,unS}, \sigma(g, R_i))$ will be used analogously. We can formulate the joint optimization problem as minimizing the maximum gap to capacity at all rates in $\mathbf{R}$ as follows.

Joint optimization algorithm

for $g = g_{min} : g_{max}$

Solve the following linear program optimization problem:

$$\text{maximize:} \quad \sum_{d_v=1}^{d_{v,max}} \frac{\lambda_{d_v}}{d_v}$$

$$\begin{aligned} \text{subject to:} \ & 1.\ \sum_{d_v=1}^{d_{v,max}} \lambda_{d_v} = 1, \lambda_{d_v} \geq 0, \\ & 2.\ I_{E,unS}(I_{A,unS}, \sigma(g, R_i)) > I_{A,S}(I_{E,S}, \sigma(g, R_i)), \\ & \quad \text{for } I_{A,unS} = I_{E,S} \in [0, 1), \text{ for all } R_i \in \mathbf{R}. \end{aligned}$$

if mother code rate corresponding to $\{\lambda_{d_v}\}$ is acceptable.
break;
return $\{\lambda_{d_v}\}$ and g.
endif
endfor

Note that the second set of constraints is the zero-error constraint at all the required code rates. We can obtain all required EXIT functions relatively quickly using our approach outlined previously. To obtain optimized $\{\lambda_{d_v}\}$, we basically keep increasing g until we get the desired code rate. The code specified by $\{\lambda_{d_v}\}$ is guaranteed to have asymptotic performance gap to capacity no larger than g at all rates in $\mathbf{R}$.

We designed a semi-structured E^2RC code that was jointly optimized across the rate range $\frac{8}{16} \sim \frac{8}{9}$, where $M = 32$, all check nodes have degree 8 and $d_{v,\max} = 20$. The code is specified by $\lambda_3 = 0.309090, \lambda_6 = 0.278794, \lambda_{20} = 0.412116$. Fig. 4 gives the simulation results for the code of block length 16384 bits (listed as code 2). Also plotted are the simulation results in Fig. 3 for code 1 and original E^2RC code from Section V-B. The puncturing performance analysis suggests that code 1 has asymptotic performance gaps around $0.21, 0.32, 0.34, 0.41, 0.405$ dB to capacity at rates $\frac{8}{16}, \frac{8}{14}, \frac{8}{12}, \frac{8}{10}$ and $\frac{8}{9}$ respectively while code 2 has much more uniform gaps of around $0.29, 0.30, 0.25, 0.29, 0.295$ dB to capacity at these rates. The simulation results in Fig. 4 also suggest uniformly better performance of code 2 compared to code 1 across the range of rates. Moreover, the measured thresholds of the two codes at all rates in simulations are in good agreement with the predicted thresholds. At all code rates, the gap between the measured threshold and the predicted threshold is around 0.35 dB. Finally, we note that code 2 achieves better or at least the same performance as original E^2RC code at all rates.

The methods described in this section apply more generally to codes that have a structured component with a protograph representation, such as IRA codes. In fact, we applied our method to the design of IRA codes as well and obtained jointly optimized codes with performance gaps around $0.27, 0.30, 0.195, 0.27, 0.29$ dB to capacity at rates $\frac{8}{16}, \frac{8}{14}, \frac{8}{12}, \frac{8}{10}$ and $\frac{8}{9}$ respectively. Though not presented here, the simulation results of the IRA code are almost identical to the jointly optimized semi-structured E^2RC code above.

To the best of our knowledge, a joint optimization algorithm for designing rate-compatible punctured codes that minimizes the gap to capacity simultaneously across all code rates has not been considered previously in the literature.

VI. Conclusions

E^2RC codes were proposed as a promising class of rate-compatible codes. In this paper, we presented systematic design and analysis of semi-structured E^2RC codes using EXIT chart. We proposed a new analytical method of computing the EXIT functions of structured parts of E^2RC codes without resorting to Monte Carlo simulations. The design of capacity-approaching E^2RC code was accomplished by linear programming. By exploring the E^2RC structure and its designed puncturing pattern, we present the method for puncturing performance analysis and propose the joint optimization algorithm for designing rate-compatible punctured codes that are simultaneously optimized at any specified set of code rates. Our jointly optimized codes with $d_{v,max} = 20$ have performance gaps to capacity no larger than 0.3 dB across the entire rate range.

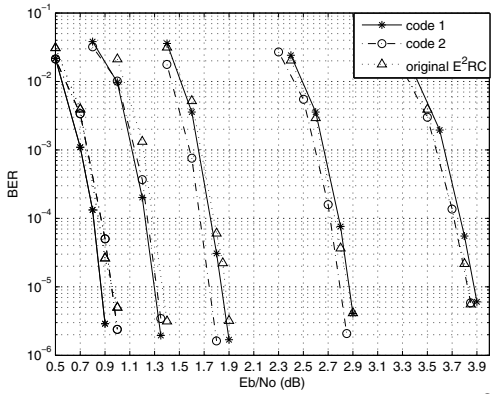

Fig. 4. Comparison between code 1, code 2 and original E^2RC code at code rates $0.5, 0.5714, 0.6667, 0.8$ and 0.8889 from left to right.

References

[1] R. G. Gallager, *Low Density Parity Check Codes*. MIT press, Cambridge, MA, 1963.

[2] J. Ha, J. Kim, and S. W. McLaughlin, "Rate-Compatible Puncturing of Low-Density Parity-Check Codes," *IEEE Trans. Inform Theory*, vol. 50, no. 11, pp. 2824 – 2836, Nov. 2004.

[3] ——, "Puncturing for Finite Length Low-Density Parity-Check Codes," *IEEE Int. Symp. Inform. Theory, Chicago*, p. 152, Jun. 2004.

[4] J. Ha, J. Kim, D. Klinc, and S. W. McLaughlin, "Rate-Compatible Punctured Low-Density Parity-Check Codes with Short Block Lengths," *IEEE Trans. Inform. Theory*, vol. 52, no. 2, pp. 728 – 738, Feb. 2006.

[5] G. Yue, X. Wang, and M. Madihian, "Design of Rate-Compatible Irregular Repeat Accumulate Codes," *IEEE Trans. Commun.*, vol. 55, no. 6, pp. 1153 – 1163, Jun. 2007.

[6] J. Kim, A. Ramamoorthy, and S. W. McLaughlin, "Design of Efficiently-Encodable Rate-Compatible Irregular LDPC Codes," *IEEE Trans. Commun. (to appear) available at http://arxiv.org/abs/0705.0543*.

[7] M. R. Yazdani and A. H. Banihashemi, "On Construction of Rate-Compatible Low-Density Parity-Check Codes," *IEEE Comm. Letters*, vol. 8, no. 3, pp. 159 – 161, Mar. 2004.

[8] S. ten Brink, "Convergence Behavior of Iteratively Decoded Parallel Concatenated Codes," *IEEE Trans. Commun.*, vol. 49, no. 10, pp. 1727 – 1737, Oct. 2001.

[9] S. ten Brink and G. Kramer, "Design of Repeat-Accumulate Codes for Iterative Detection and Decoding," *IEEE Trans. Signal Processing*, vol. 51, no. 11, pp. 2764 – 2772, Nov. 2003.

[10] S. ten Brink, G. Kramer, and A. Ashikhmin, "Design of Low-Density Parity-Check Codes for Modulation and Detection," *IEEE Trans. Commun.*, vol. 52, no. 4, pp. 670 – 678, Apr. 2004.

[11] A. Ashikhmin, G. Kramer, and S. ten Brink, "Extrinsic Information Transfer Functions: Model and Erasure Channel Properties," *IEEE Trans. Inform. Theory*, vol. 50, no. 11, pp. 2657 – 2673, Nov. 2004.

[12] E. Sharon, A. Ashikhmin, and S. Litsyn, "Analysis of Low-Density Parity-Check Codes Based on EXIT Functions," *IEEE Trans. Communications*, vol. 54, no. 8, pp. 1407 – 1414, Aug. 2006.