Zig-zag and replacement product graphs and LDPC codes

Kelley, C ; Sridhara, D ; Rosenthal, J

DOI: https://doi.org/10.3934/amc.2008.2.347

Zig-zag and Replacement Product Graphs and LDPC Codes

Christine A. Kelley
The Fields Institute
222 College Street
Toronto, ON M5T 3J1, Canada.
email: ckelley@fields.utoronto.ca

Deepak Sridhara and Joachim Rosenthal
Institut für Mathematik
Universität Zürich
CH-8057, Zürich, Switzerland.
email: {sridhara, rosen}@math.unizh.ch

Abstract

The performance of codes defined from graphs depends on the expansion property of the underlying graph in a crucial way. Graph products, such as the zig-zag product [13] and replacement product provide new infinite families of constant degree expander graphs. The paper investigates the use of zig-zag and replacement product graphs for the construction of codes on graphs [16]. The original zig-zag construction is adapted in order to arrive at a family of iteratively constructed bi-partite graphs. It is shown that this family forms an infinite family of bipartite expander graphs.

Index Terms

Codes on graphs, LDPC codes, expander graphs, zig-zag product, replacement product of a graph.

I. Introduction

Expander graphs are of fundamental interest in mathematics and engineering and have several applications in computer science, complexity theory, designing communication networks, and coding theory [8], [1], [17]. In a remarkable paper [13] Reingold, Vadhan, and Wigderson introduced an iterative construction which leads to infinite families of constant degree expander graphs. The iterative construction is based on the *zig-zag graph product* introduced by the authors in the same paper. The zig-zag product of two regular graphs is a new graph whose degree is equal to the degree of the second graph and whose expansion property depends on the expansion properties of the two component graphs. In particular if both component graphs are good expanders, then their zig-zag product is a good expander as well. Similar things can be said about the replacement product.

Since the work of Sipser and Spielman [15] it has been well known that the performance of codes defined on graphs depends on the expansion property of the underlying graph in a crucial way. Several authors have provided constructions of codes from graphs whose underlying graphs are good expanders. In general, a graph that is a good expander is particularly suited for the message-passing decoder that is used

This work was supported in part by the NSF Grant No. CCR-ITR-02-05310 and the Swiss NSF Grant No. 113251.

to decode LDPC codes, in that it allows for messages to be dispersed to all nodes in the graph as quickly as possible. Furthermore, graphs with good expansion yield LDPC codes with good minimum distance and pseudocodeword weights [6], [4], [7], [14], [15].

Probably the most prominent example of expander graphs are the class of Ramanujan graphs which are characterized by the property that the second eigenvalue of the adjacency matrix is minimal inside the class of k-regular graphs on n vertices. This family of 'maximal expander graphs' was independently constructed by Lubotzky, Phillips and Sarnak [10] and by Margulis [11]. The description of these graphs and their analysis rely on deep results from mathematics using tools from graph theory, number theory, and representation theory of groups [9]. Codes from Ramanujan graphs were constructed and studied by several authors [7], [14], [18].

Ramanujan graphs have the drawback, that they exist only for a limited set of parameters. In contrast to this the zig-zag product and the replacement product can be performed on a large variety of component graphs. The iterative construction has also a lot of engineering appeal as it allows one to construct larger graphs from smaller graphs as one desires. This was the starting point of our research reported in [5].

In this paper we examine the expansion properties of the zig-zag product and the replacement product in relation to the design of LDPC codes. We also introduce variants of the zig-zag scheme that allow for the component graphs to be unbalanced bipartite graphs. We then propose an iterative construction of the new variant and also an iterative construction of the replacement product to generate a family of expanders. In our code construction, the vertices of the product graph are interpreted as sub-code constraints of a suitable linear block code and the edges are interpreted as the code bits of the LDPC code, as originally suggested by Tanner in [16]. By choosing component graphs with relatively small degree, we obtain product graphs that are relatively sparse. Examples of each product and resulting LDPC codes are given to illustrate the results of this paper. Some of the examples use Cayley graphs as components, and the resulting product graph is also a Cayley graph with the underlying group being the semi-direct product of the component groups, and the new generating set being a function of the generating sets of the components [2]. Simulation results reveal that LDPC codes based on zig-zag and replacement product graphs perform comparably to, if not better than, random LDPC codes of comparable block lengths and rate. The vertices of the product graph must be fortified with strong (i.e., good minimum distance) sub-code constraints, in order to achieve good performance with message-passing (or, iterative) decoding.

The paper is organized as follows. Section 2 discusses preliminaries on the formal definition of expansion for a d-regular graph and the best one can achieve in terms of expansion. Furthermore, expansion for a general graph is discussed. Section 3 describes the original zig-zag product and replacement product. Futhermore, a variant of the zig-zag product that allows for unbalanced bipartite component graphs is introduced. The properties of the product graphs, such as expansion, diameter, and girth are also discussed in this section. Section 4 presents examples of the original and the unbalanced bipartite zig-zag products, the replacement product, and LDPC codes obtained from these examples. Simulation results of the LDPC codes constructed are presented in Section 5. Section 6 introduces an iterative construction for an unbalanced bipartite zig-zag

product and the replacement product to generate families of expanders with constant degree. For completion, the iterative construction for the original zig-zag product from [13] is also described. The expansion for the iterative families are also discussed. Section 7 summarizes the results and concludes the paper.

II. PRELIMINARIES

In this section, we introduce some preliminaries on the expansion of graphs.

Definition 2.1: A d-regular graph G on N vertices is said to be a (N, d, λ)-graph if the second largest eigenvalue of the normalized adjacency matrix $\tilde{A}$ representing G is λ.

It is now almost common knowledge that for a graph to be a good expander [15], the second largest eigenvalue of the adjacency matrix A must be as small as possible compared to the largest eigenvalue [17]. For a d-regular graph G, the index of the adjacency matrix A is d.

Hence, by *normalizing* the entries of A by the factor d, the normalized matrix $\tilde{A} = \frac{1}{d}A$ has the largest eigenvalue equal to 1. In this paper, we will follow the definition provided in [2], [12] for a graph to be an *expander*. A sequence of graphs is said to be an expander family if for every (connected) graph G in the family, the second largest eigenvalue $\lambda(G)$ of the normalized adjacency matrix $\tilde{A}$ is bounded below some constant $\lambda_q < 1$. Or in other words, there is an $\epsilon > 0$ such that for every graph G in the family, $\lambda(G) < 1 - \epsilon$. In particular, a graph belonging to an expander family is called an expander graph. For d-regular (connected) graphs, the best possible expansion based on the eigenvalue bound is achieved by Ramanujan graphs that have $\lambda(G) \leq \frac{2\sqrt{d-1}}{d}$ [10]. Alon and Boppana have shown that for a d-regular graph G, as the number of vertices n in G tends to infinity, $\lambda(G) \geq \frac{2\sqrt{d-1}}{d}$ [1]; therefore, Ramanujan graphs are optimal in terms of the eigenvalue gap $1 - \lambda(G)$.

The definition of expansion to d-regular graphs can be similarly extended to (c, d)-regular bipartite graphs as defined below and also to general irregular graphs.

Definition 2.2: A graph $G = (X, Y; E)$ is (c, d)-*regular bipartite* if the set of vertices in G can be partitioned into two disjoint sets X and Y such that all vertices in X (called *left* vertices) have degree c and all vertices in Y (called *right* vertices) have degree d and each edge $e \in E$ of G is incident with one vertex in X and one vertex in Y, i.e., $e = (x, y), x \in X, y \in Y$.

Definition 2.3: A (c, d)-regular bipartite graph G on N left vertices and M right vertices is said to be a (N, M, c, d, λ)-graph if the second largest eigenvalue of the normalized adjacency matrix $\tilde{A}$ representing G is λ.

The largest eigenvalue of a (c, d)-regular graph is $\sqrt{cd}$. Once again, normalizing the adjacency matrix of a (c, d)-regular bipartite graph by its largest eigenvalue $\sqrt{cd}$, we have that the (connected) graph is a good expander if second largest eigenvalue of its normalized adjacency matrix is bounded away from 1 and is as small as possible.

To normalize the entries of an irregular graph G defined by the adjacency matrix $A = (a_{ij})$, we scale each $(i, j)^{th}$ entry in A by $\frac{1}{r_i c_j}$, where r_i and c_j are the i^{th} row weight and j^{th} column weight, respectively, in A.

It is easy to show that the resulting normalized adjacency matrix has its largest eigenvalue equal to one. The definition of an expander for an irregular graph G can be defined analogously.

III. Graph Products

In designing codes over graphs, graphs with good expansion, relatively small degree, small diameter, and large girth are desired. Product graphs give a nice avenue for code construction, in that taking the product of small graphs suitable for coding can yield larger graphs (and therefore, codes) that preserve these desired properties. Standard graph products, however, such as the Cartesian product, tensor product, lexicographic product, and strong product, all yield graphs with large degree and girth $g = 4$ [3], and so are not good candidates for LDPC code construction. (Sparsity is essential for efficient graph-based message-passing decoding.)

In this section we describe the zig-zag product of [8], [13], introduce a variation of the zig-zag product that holds for bi-regular (unbalanced) bipartite graphs, and review the replacement product. In each case, the expansion of the product graph with respect to the expansion of the component graphs is examined. When the graph is regular-bipartite, this bi-regular product yields the product in [8], [13]. An iteration method for these products will be given in a later section.

A. Zig-zag product

Let G_1 be a (N_1, d_1, λ_1)-graph and let G_2 be a (d_1, d_2, λ_2) graph. Randomly number the edges around each vertex of G_1 by $\{1, \ldots, d_1\}$, and each vertex of G_2 by $\{1, \ldots, d_2\}$.Then the zig-zag product of G_1 and G_2, as introduced in [8], [13], is a graph G defined as follows[1]:

- the vertices of G are represented as ordered pairs (v, k), where $v \in \{1, 2, \ldots, N_1\}$ and $k \in \{1, 2, \ldots, d_1\}$. That is, every vertex in G_1 is replaced by a cloud of vertices of G_2.
- the edges of G are formed by making two steps on the small graph and one step on the big graph as follows:
  - a step "*zig*" on the small graph G_2 is made from vertex (v, k) to vertex $(v, k[i])$, where $k[i]$ denotes the i^{th} neighbor of k in G_2, for $i \in \{1, 2, \ldots, d_2\}$.
  - a deterministic step on the large graph G_1 is made from vertex $(v, k[i])$ to vertex $(v[k[i]], \ell)$, where $v[k[i]]$ is the $k[i]^{th}$ neighbor of v in G_1 and correspondingly, v is the ℓ^{th} neighbor of $v[k[i]]$ in G_1.
  - a final step "*zag*" on the small graph G_2 is made from vertex $(v[k[i]], \ell)$ to vertex $(v[k[i]], \ell[j])$, where $\ell[j]$ is the j^{th} neighbor of ℓ in G_2, for $j \in \{1, 2, \ldots, d_2\}$.

  Therefore, there is an edge between vertices (v, k) and $(v[k[i]], k[i][j])$ for $i, j \in \{1, \ldots, d_2\}$.

[1] This is actually the second presentation of the zig-zag product given in [13]; the original description required $\ell = k[i]$ in step 2 of the product, i.e. each endpoint of an edge had to have the same label.

It is shown in [13] that the zig-zag product graph $G = G_1 (z) G_2$ is a $(N_1 \cdot d_1, d_2^2, \lambda)$-graph with $\lambda < \lambda_1 + \lambda_2 + \lambda_2^2$, and further, that $\lambda < 1$ if $\lambda_1 < 1$ and $\lambda_2 < 1$. Therefore, the degree of the zig-zag product graph depends only on the smaller component graph whereas the expansion property depends on the expansion of both the component graphs, i.e., it is a good expander if the two component graphs are good expanders.

Lemma 3.1: Let G_1 and G_2 have girth g_1 and g_2, respectively. Then the zig-zag product graph $G = G_1 (z) G_2$ has girth $g = 4$.

Proof: We show that any pair of vertices at distance 2 in G_2 are involved in a 4-cycle in G. Consider two vertices (v_1, k_1) and (v_1, k_2) in the same cloud of G that lie at distance 2 apart in G_2. Let (v_1, k_3) be their common neighbor. In step 1 of the zig-zag product, an edge will start from (v_1, k_1) and (v_1, k_2) to (v_1, k_3). Note that the deterministic step will then continue the edge from (v_1, k_3) to a specified vertex $(\tilde{v}, \tilde{k})$ in another cloud. Therefore, with step 3, the actual edges in G will go from (v_1, k_1) to the neighbors of $(\tilde{v}, \tilde{k})$, and from (v_1, k_2) to the neighbors of $(\tilde{v}, \tilde{k})$. Therefore, (v_1, k_1) and (v_1, k_2) are involved in a 4-cycle provided $(\tilde{v}, \tilde{k})$ does not have degree 1. Since it is assumed G_2 is a connected graph with more than 2 vertices, there is a pair of vertices such that the resulting $(\tilde{v}, \tilde{k})$ has degree > 1 in G_2. ■

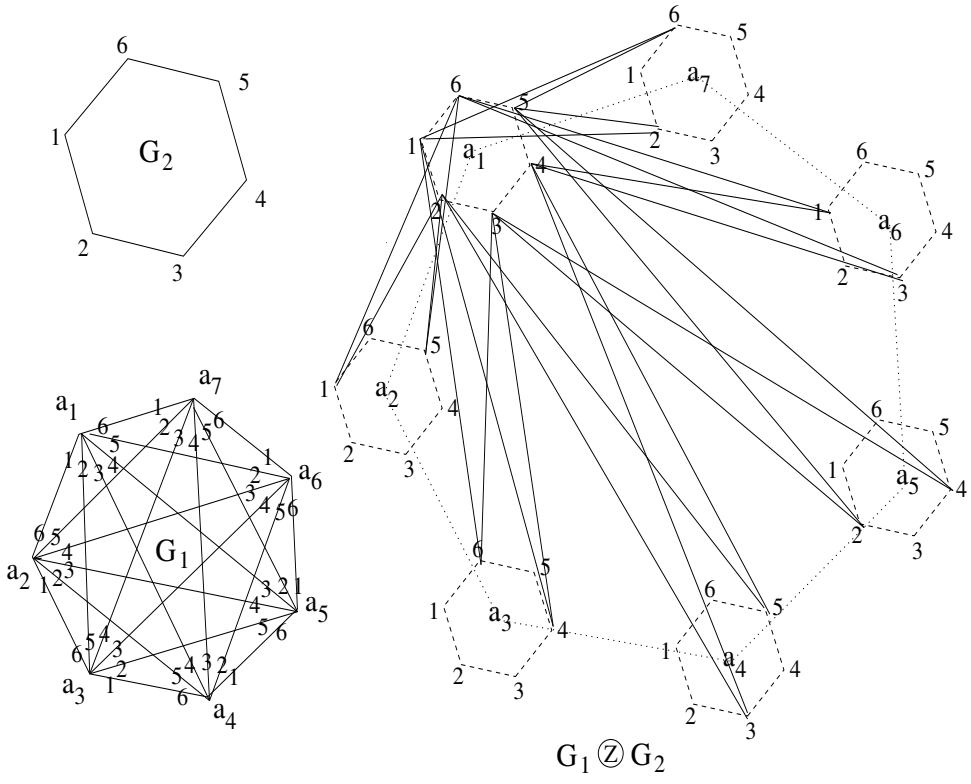


Fig. 1. Zig-Zag product of two graphs.

We now consider the case when the two component graphs are Cayley graphs [14]. Suppose $G_1 = C(G_a, S_a)$ is the Cayley graph formed from the group G_a with S_a as its generating set. This means that G_1 has the elements of G_a as vertices and there is an edge from the vertex representing $g \in G_a$ to the vertex representing

$h \in G_a$ if for some $s \in S_a$, $g * s = h$, where '$*$' denotes the group operation. If the generating set S_a is symmetric, i.e., if $a \in S_a$ implies $a^{-1} \in S_a$, then the Cayley graph is undirected.

Let the two components of our (zig-zag product) graph be Cayley graphs of the type $G_1 = C(G_a, S_a)$ and $G_2 = C(G_b, S_b)$ and further, let us assume that there is a well-defined group action by the group G_b on the elements of the group G_a.Then the product graph is again a Cayley graph. More specifically, if $G_1 = C(G_a, S_a)$ and $G_2 = C(G_b, S_b)$, and if S_a is the orbit of k elements $a_1, a_2, \ldots, a_k \in G_a$ under the action of G_b, then the generating set S for the Cayley (zig-zag product) graph is

$$S = \{(1_{G_a}, \beta)(a_i, 1_{G_b})(1_{G_a}, \beta')|\ \beta, \beta' \in S_b, i \in 1, \ldots, k\}.$$

It is easily verified that when $k = 1$, the Cayley graph $C(G_a \rtimes G_b, S)$ is the zig-zag product originally defined in [13]. The degree of this Cayley graph is at most $k|S_b|^2$ if we disallow multiple edges between vertices. When the group sizes G_a and G_b are large and the k distinct elements $a_1, a_2, \ldots, a_k \in G_a$ are chosen randomly, then the degree of the product graph is almost always $k|S_b|^2$.

B. Zig-zag product for unbalanced bipartite graphs

Extending the original zig-zag construction in a straightforward manner, we are now able to define the zig-zag product construction for the case when the two component graphs are unbalanced bipartite graphs, i.e., the two sets of vertices have different degrees. Let G_1 be a (c_1, d_1)-regular graph on the vertex sets V_1, W_1, where $|V_1| = N$ and $|W_1| = M$. Let G_2 be a (c_2, d_2)-regular graph on the vertex sets V_2, W_2, where $|V_2| = d_1$ and $|W_2| = c_1$. Let λ_1 and λ_2 denote the second largest eigenvalues of the normalized adjacency matrices of G_1 and G_2, respectively. Again, randomly number the edges around each vertex $\tilde{v}$ in G_1 and G_2 by $\{1, \ldots, deg(\tilde{v})\}$, where $deg(\tilde{v})$ is the degree of $\tilde{v}$. Then the zig-zag product graph, which we will denote by $G = G_1 (z) G_2$, is a (c_2^2, d_2^2)-regular bipartite graph on the vertex sets V, W with $|V| = N \cdot d_1$, $|W| = M \cdot c_1$, formed in the following manner:

- Every vertex $v \in V_1$ and $w \in W_1$ of G_1 is replaced by a copy of G_2. The cloud at a vertex $v \in V_1$ has vertices V_2 on the left and vertices W_2 on the right, with each vertex from W_2 corresponding to an edge from v in G_1. The cloud at a vertex $w \in W_1$ is similarly structured with each vertex in V_2 in the cloud corresponding to an edge of w in G_1. (See Figure 2.) Then the vertices from V are represented as ordered pairs (v, k), for $v \in \{1, \ldots, N\}$ and $k \in \{1, \ldots, d_1\}$, and the vertices from W are represented as ordered pairs (w, ℓ), for $w \in \{1, \ldots, M\}$ and $\ell \in \{1, \ldots, c_1\}$.
- A vertex $(v, k) \in V$ is connected to a vertex in W by making three steps in the product graph:
  - A small step "zig" from left to right in the local copy of G_2. This is a step $(v, k) \rightarrow (v, k[i])$, for $i \in \{1, \ldots, c_2\}$.
  - A deterministic step from left to right on G_1 $(v, k[i]) \rightarrow (v[k[i]], \ell)$, where $v[k[i]]$ is the $k[i]^{th}$ neighbor of v in G_1 and v is the ℓ^{th} neighbor of $v[k[i]]$ in G_1.

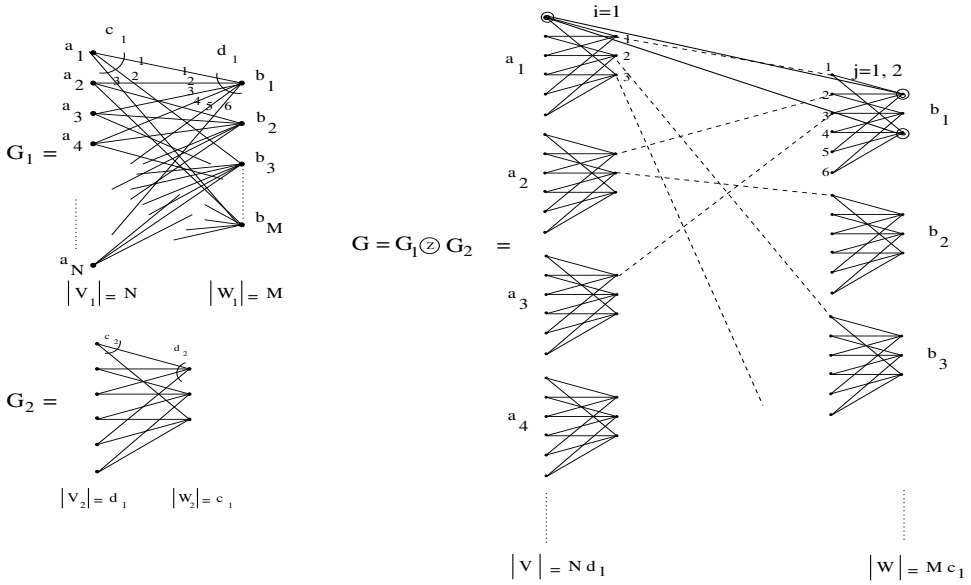


Fig. 2. Zig-Zag product of two unbalanced bipartite graphs.

– A small step "zag" from left to right in the local copy of G_2. This is a step $(v[k[i]], \ell) \rightarrow (v[k[i]], \ell[j])$, where the final vertex is in W, for $j \in \{1, \ldots, c_2\}$.

Therefore, there is an edge between (v, k) and $(v[k[i]], \ell[j])$.

Theorem 3.2: Let G_1 be a (c_1, d_1)-regular bipartite graph on (N, M) vertices with $\lambda(G_1) = \lambda_1$, and let G_2 be a (c_2, d_2)-regular bipartite graph on (d_1, c_1) vertices with $\lambda(G_2) = \lambda_2$. Then, the zig-zag product graph $G_1 \textcircled{z} G_2$ is a (c_2^2, d_2^2)-regular bipartite on $(N \cdot d_1, M \cdot c_1)$ vertices with $\lambda = \lambda(G_1 \textcircled{z} G_2) \leq \lambda_1 + \lambda_2 + \lambda_2^2$. Moreover, if $\lambda_1 < 1$ and $\lambda_2 < 1$, then $\lambda = \lambda(G_1 \textcircled{z} G_2) < 1$.

The proof of the expansion of the unbalanced zig-zag product graph is similar to that of the original zig-zag product graph [13]. (The case of balanced bipartite graphs has been dealt in [8] and is a special case of this generalized variation.) Note that unlike in the original zig-zag product construction [13], the vertex set of G does not include vertices from the set W_2 in any cloud of vertices from V_1, nor vertices from V_2 in any cloud of vertices from W_1. However, the girth of the unbalanced zig-zag product is also 4, and can be seen using a similar argument as in Lemma 3.1.

C. Replacement product

Let G_1 be a (N_1, d_1, λ_1)-graph and let G_2 be a (d_1, d_2, λ_2)-graph. (Observe that the number of vertices in G_2 is chosen to be equal to the degree of each vertex in G_1.) Randomly number the edges around each vertex of G_1 by $\{1, \ldots, d_1\}$, and each vertex of G_2 by $\{1, \ldots, d_2\}$. Then the replacement product of G_1 and G_2 is a graph G with the vertex set and edge set defined as follows: the vertices of G are represented as ordered two tuples (v, k), for $v \in \{1, 2, \ldots, N_1\}$ and $k \in \{1, 2, \ldots, d_1\}$. There is an edge between (v, k) and (v, ℓ) if there is an edge between k and ℓ in G_2; there is also an edge between (v, k) and (w, ℓ) if the k^{th} edge

incident on vertex v in G_1 is connected to vertex w and this edge is the ℓ^{th} edge incident on w in G_1. The replacement product graph $G = G_1 \circledR G_2$ is a $(N_1 \cdot d_1, d_2+1, \lambda)$-graph with $\lambda \leq (p+(1-p)f(\lambda_1,\lambda_2))^{1/3}$ for $p = d_2^2/(d_2+1)^3$, where $f(\lambda_1,\lambda_2) = \lambda_1+\lambda_2+\lambda_2^2$ [13, Theorem 6.4]. Note that the degree of the replacement product graph depends only on the degree of the smaller component graph G_2.

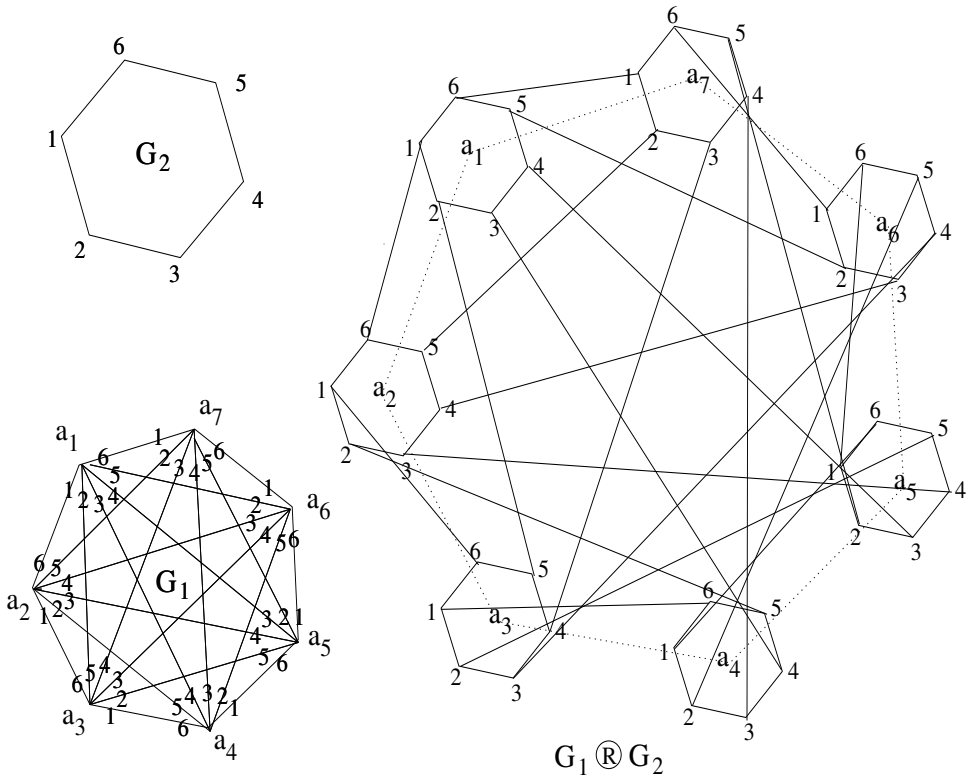


Fig. 3. Replacement product of two graphs.

Lemma 3.3: Let G_1 (resp., G_2) have girth g_1 and diameter t_1(resp., g_2, t_2). Then the girth g and diameter t of the replacement product graph $G = G_1 \circledR G_2$ are given by: (a) girth $\min\{g_2, 2g_1\} \leq g \leq \min\{g_2, g_1 t_2\}$, and (b) diameter $\max\{t_2, 2t_1\} \leq t \leq t_1 + t_2$.

Proof: (a) Observe that there are clearly cycles of length g_2 in G, as G_2 is a subgraph of G. Moreover, consider two vertices in G_1 on a cycle of length g_1. Their clouds are g_1 apart in G, so a smallest cycle between them would contain at most $g_1 t_2$ edges (in the worst case, t_2 steps would be needed within each cloud in the G_1-cycle). So $g \leq \min\{g_2, g_1 t_2\}$. For the lower bound, the smallest cycle possible involving vertices in different clouds has length $2g_1$, and would occur if in the cycle, only one step was needed on each cloud. Thus, $g \geq \min\{g_2, 2g_1\}$. (b) For the diameter, the furthest two vertices could be would occur if they belonged to clouds associated to vertices at distance t_1 apart in G_1, and the path between them in G would require at most t_2 steps on each cloud. Therefore, $t \leq t_1 t_2$. Similarly, the furthest distance between vertices in the same cloud is t_2, and the furthest distance between vertices in different clouds is at least $2t_1$, which would occur if they lie in clouds associated to vertices at distance t_1 apart in G_1, but only one step

was needed on each cloud on the path. So $t \geq \max\{t_2, 2t_1\}$. ■

As earlier, let the two components of the product graph be Cayley graphs of the type $G_1 = C(G_a, S_a)$ and $G_2 = C(G_b, S_b)$ and again assume that there is a well-defined group action by the group G_b on the elements of the group G_a. Then the replacement product graph is again a Cayley graph. If S_a is the union of k orbits, i.e., the orbits of $a_1, a_2, \ldots, a_k \in G_a$ under the action of G_b, then the replacement product graph is the Cayley graph of the semi-direct product group $G_a \rtimes G_b$ and has $(1_{G_a}, S_b) \bigcup \{(a_1, 1_{G_b}), \ldots, (a_k, 1_{G_b})\}$ as the generating set. The degree of this Cayley graph is $|S_b| + k$ and the size of its vertex set is $|G_a||G_b|$ [8].

IV. Zig-zag and replacement product LDPC codes

In this section, we use the zig-zag and replacement product graphs as building blocks for designing LDPC codes. The zig-zag product of regular graphs yields a regular graph which may or may not be bipartite, depending on the choice of the component graphs. Therefore, to translate the zig-zag product graph into a LDPC code, the vertices of the zig-zag product are interpreted as sub-code constraints of a suitable linear block code and the edges are interpreted as code bits of the LDPC code. This is akin to the procedure described in [16] and [7]. The same procedure is applied to the replacement product graphs.

We further restrict the choice of the component graphs for our products to be appropriate Cayley graphs so that we can work directly with the group structure of the Cayley graphs. The following examples, the first two using Cayley graphs from [2], illustrate the code construction technique:

Example 4.1: Let $A = \mathbb{F}_2^p$ be the Galois field of 2^p elements for a prime p, where the elements of A are represented as vectors of a p-dimensional vector space over $\mathbb{F}_2$. Let $B = \mathbb{Z}_p$ be the group of integers modulo p. (Further, let p be chosen such that the element 2 generates the multiplicative group $\mathbb{Z}_p^* = \mathbb{Z}_p - \{0\}$.) The group B acts on an element $\mathbf{x} = (x_0, x_1, \ldots, x_{p-1}) \in A$ by cyclically shifting its coordinates, i.e. $\phi_b(\mathbf{x}) = (x_b, x_{b+1}, \ldots, x_{b-1}),\ \forall b \in B$. Let us now choose k elements $a_1, a_2, \ldots, a_k$ randomly from A. The result in [2, Theorem 3.6] says that for a random choice of elements $a_1, a_2, \ldots, a_k$, the Cayley graph $C(A, \{a_1^B, a_2^B, \ldots, a_k^B\})$ is an expander with high probability. (Here, a_i^B is the orbit of a_i under the action of B.) The Cayley graph for the group B with the generators $\{\pm 1\}$ is the cyclic graph on p vertices, $C(B, \{\pm 1\})$.

(a) The zig-zag product of the two Cayley graphs is the Cayley graph

$$C(A \rtimes B, S = \{(0, \beta)(a_i, 0)(0, \beta')|\ \beta, \beta' = \pm 1, i = 1, 2, .., k\})$$

on $N = 2^p \cdot p$ vertices, where $A \rtimes B$ is the semi-direct product group and the group operation is $(a, b)(c, d) = (a + \phi_b(c), b + d)$, for $a, c \in A,\ b, d \in B$. This is a regular graph with degree[2] $d_g \leq k|S_B|^2 = 4k$. If we interpret the vertices of the graph as sub-code constraints of a $[d_g, k_g, d_m]$ linear block code and the edges of the graph as code bits of the LDPC code, then the block length N_{LD} of the LDPC code is $2^p \cdot p \cdot d_g/2$ and

[2]Depending on the choice of the a_i's, the number of distinct elements in S may be fewer than $k|S_B|^2$.

the rate of the LDPC code is

$$r \geq \frac{N_{LD} - N(d_g - k_g)}{N_{LD}} = 1 - \frac{2(d_g - k_g)}{d_g} = \frac{2k_g}{d_g} - 1.$$

(Observe that $r \geq 2r_1 - 1$, where r_1 is the rate of the sub-code.)

(b) The replacement product of the two Cayley graphs is the Cayley graph

$$C(A \rtimes B, S = (0, S_B) \cup \{(a_i, 0) | i = 1, 2, .., k\})$$

on $N = 2^p \cdot p$ vertices, where $A \rtimes B$ is the semi-direct product group and the group operation is $(a, b)(c, d) = (a + \phi_b(c), b + d)$, for $a, c \in A,\ b, d \in B$. This is a regular graph with degree $d_g = k + |S_B| = k + 2$. We interpret the vertices of the graph as sub-code constraints of a $[d_g, k_g, d_m]$ linear block code and the edges of the graph as code bits of the LDPC code, to obtain an LDPC code of block length $N_{LD} = 2^p \cdot p \cdot d_g/2$ and rate

$$r \geq \frac{N_{LD} - N(d_g - k_g)}{N_{LD}} = 1 - \frac{2(d_g - k_g)}{d_g} = \frac{2k_g}{d_g} - 1.$$

(Observe that $r \geq 2r_1 - 1$, where r_1 is the rate of the sub-code.) □

In some cases, to achieve a certain desired rate, we may have to use a mixture of sub-code constraints from two or more linear block codes. For example, to design a rate 1/2 LDPC code when d_g is odd, we may have to impose a combination of $[d_g, k_g, d_{m1}]$ and $[d_g, k_g + 1, d_{m2}]$ block code constraints, for an appropriate k_g, on the vertices of the graph.

Example 4.2: Let $B = SL_2(\mathbb{F}_p)$ be the group of all 2×2 matrices over $\mathbb{F}_p$ with determinant one. Let $S_B = \left\{ \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}, \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix} \right\}$ be the generating set for the Cayley graph $C(B, S_B)$. Further, let $\mathbb{P}_1 = \mathbb{F}_p \cup \{\infty\}$ be the projective line. The *Möbius* action of B on $\mathbb{P}_1$ is given by $\begin{pmatrix} a & b \\ c & d \end{pmatrix}(x) = \frac{ax+b}{cx+d}$. Let $A = \mathbb{F}_2^{\mathbb{P}_1}$ and let the action of B on the elements of A be the Möbius permutation of the coordinates as above. If we now choose k elements $a_1, a_2, \ldots, a_k$ randomly from A as in the previous example, then [2] again shows that with high probability, the Cayley graph $C(A, \{a_1^B, \ldots, a_k^B\})$ is an expander.

(a) The zig-zag product of the two Cayley graphs is the Cayley graph

$$C(A \rtimes B, S = \{(1_A, \beta)(a_i, 1_B)(1_A, \beta') |\ \beta, \beta' \in S_B, i = 1, 2, .., k\})$$

on $|A||B| = 2^{p+1}(p^3 - p)$ vertices. (Note that $1_B = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$ and $1_A = \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix}$.) However, this Cayley graph will be a directed Cayley graph since the generating set S is not symmetric. Hence, we modify our graph construction by taking two copies of the vertex set $A \rtimes B$. A vertex v from one copy is connected to vertex w in the other copy if there is a $s \in S$ such that $v * s = w$. The new product graph obtained has $2|A||B|$ vertices and every vertex has degree $d_g = |S|$; moreover, it is a balanced bipartite graph. An LDPC

code of block length $|A||B|d_g$ is obtained by interpreting the vertices of the graph as sub-code constraints of a $[d_g, k_g, d_m]$ linear block code, and the edges as code bits of the LDPC code. The rate of this code is

$$r \geq 1 - \frac{2(d_g - k_g)}{d_g} = \frac{2k_g}{d_g} - 1.$$

(b) The replacement product of the two Cayley graphs is the Cayley graph

$$C(A \rtimes B, S = (1_A, S_B) \cup \{(a_i, 1_B) | i = 1, 2, \ldots, k\}),$$

on $|A||B| = 2^{p+1}(p^3 - p)$ vertices. Here also, the Cayley graph will be a directed Cayley graph since the generating set S is not symmetric. Hence, we modify our graph construction by taking two copies of the vertex set $A \rtimes B$. A vertex v from one copy is connected to vertex w in the other copy if there is a $s \in S$ such that $v * s = w$. The new product graph obtained has $2|A||B|$ vertices and every vertex has degree $d_g = |S|$; moreover, it is a balanced bipartite graph. An LDPC code of block length $|A||B|d_g$ is obtained by interpreting the vertices of the graph as sub-code constraints of a $[d_g, k_g, d_m]$ linear block code, and the edges as code bits of the LDPC code. The rate of this code is

$$r \geq 1 - \frac{2(d_g - k_g)}{d_g} = \frac{2k_g}{d_g} - 1.$$

□

Example 4.3: Codes from unbalanced bipartite zig-zag product graphs.
Using a random construction, we design a (c_1, d_1)-regular bipartite graph G_1 on (N, M) vertices. Similarly, we design a (c_2, d_2)-regular bipartite graph G_2 on (d_1, c_1) vertices. The zig-zag product of G_1 and G_2 is a (c_2^2, d_2^2)-regular graph on $(N \cdot d_1, M \cdot c_1)$ vertices. An LDPC code is obtained as before by interpreting the degree c_2^2 vertices [resp. degree d_2^2 vertices] as sub-code constraints of a $C_{S1} = [c_2^2, k_1, d_{m1}]$ [resp. a $C_{S2} = [d_2^2, k_2, d_{m2}]$] linear block code and the edges of the product graph as code bits of the LDPC code. The block length of the LDPC code thus obtained is $N_{LD} = Nd_1c_2^2$ and the rate is

$$r \geq \frac{Nd_1c_2^2 - (Nd_1(c_2^2 - k_1) + Mc_1(d_2^2 - k_2))}{Nd_1c_2^2} = \frac{k_1}{c_2^2} + \frac{k_2}{d_2^2} - 1$$

(since $Nd_1c_2^2 = Mc_1d_2^2$ is the number of edges in the graph). Observe that $r \geq r_1 + r_2 - 1$, where r_1 and r_2 are the rates of the two sub-codes C_{S1} and C_{S2}, respectively. □

V. Performance of Zig-zag and Replacement Product LDPC Codes

The performance of the LDPC code designs based on zig-zag and replacement product graphs is examined for use over the additive white Gaussian noise (AWGN) channel. (Binary modulation is simulated and the bit error performance with respect to signal to noise ratio (SNR) E_b/N_o is determined.) The LDPC codes are decoded using the graph based iterative sum-product (SP) algorithm. Since LDPC codes based on product graphs use sub-code constraints, the decoding at the constraint nodes is accomplished using the BCJR algorithm

on a trellis representation of the appropriate sub-code. (A simple procedure to obtain the trellis representation of the sub-code based on its parity check matrix representation is discussed in [19].) It must be noted that as the number of states in the trellis representation and the block length of the sub-code increases, the decoding complexity correspondingly increases.

Figure 4 shows the performance of the zig-zag product LDPC codes based on Example 4.1, with sum-product decoding. For the parameters $p = 5$ and $k = 5$, five elements in $A = \mathbb{F}_2^p$ are chosen (randomly) to yield a set of generators for the Cayley graph of the semi-direct product group. The Cayley graph has 160 vertices, each of degree 20. The sub-code used for the zig-zag LDPC code design is a $[20, 15, 4]$ code and the resulting LDPC code has rate 1/2 and block length 1600. The figure also shows the performance of a LDPC code based on a randomly designed degree 20 regular graph on 160 vertices which also uses the same sub-code constraints as the former code. The two codes perform comparably, indicating that the expansion of the zig-zag product code compares well with that of a random graph of similar size and degree. Also shown in the figure is the performance of a $(3, 6)$ regular LDPC code, that uses no special sub-code constraints other than simple parity check constraints, having the same block length and rate. Clearly, using strong sub-code constraints improves the performance significantly, albeit at the cost of higher decoding complexity. The figure also shows another set of curves for a longer block length design. Choosing $p = 11$ and $k = 5$ and the $[20, 15, 4]$ sub-code constraints yields a rate 1/2 and block length 225,280 zig-zag product LDPC code. At this block length also, the LDPC based on the zig-zag product graph is found to perform comparably, if not, better than the LDPC code based on a random degree 20 graph. The zig-zag product graph has a poor girth[3] and this causes the performance of the zigzag LDPC code to be inferior to that of the random LDPC codes at high signal to noise ratios.

Figure 5 shows the performance of a replacement product LDPC code based on Example 4.1, with sum-product decoding. For the parameters $p = 11$ and $k = 13$, 13 elements in $A = \mathbb{F}_2^p$ are chosen (randomly) to yield a set of generators for the Cayley graph of the semi-direct product group. The Cayley graph has 22,528 vertices, each of degree 15. The sub-code used for the replacement product LDPC code design is a $[15, 11, 3]$ Hamming code and the resulting LDPC code has rate 0.4667 and block length 168,960. The figure also shows the performance of a LDPC code based on a randomly designed degree 15 regular graph on 22,528 vertices which also uses the same sub-code constraints as the former code. Here again, the two codes perform comparably, indicating that the expansion of the replacement product code compares well with that of a random graph of similar size and degree.

[3]Note that there is no growth in the girth of the zig-zag product graph as opposed to that for a randomly chosen graph, with increasing graph size.

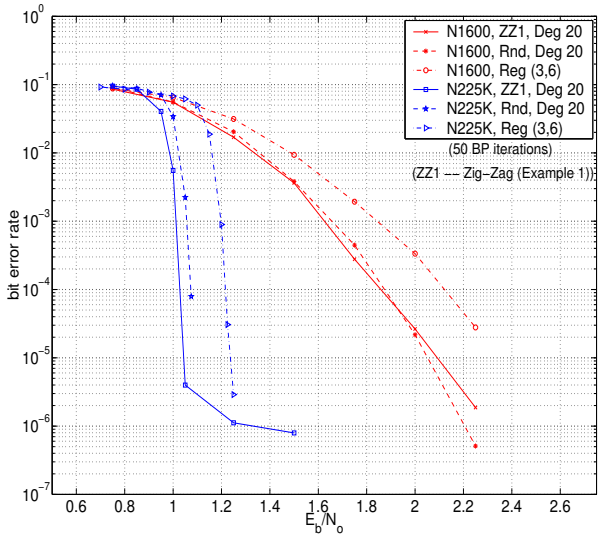


Fig. 4. LDPC codes from zig-zag product graphs based on Example 4.1.

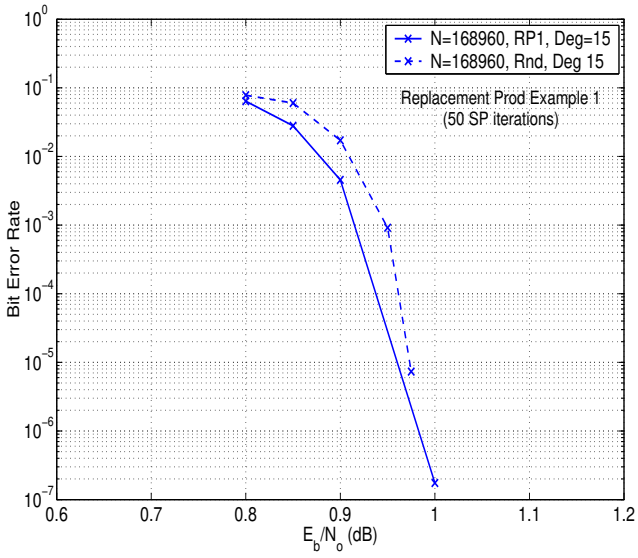


Fig. 5. LDPC code from replacement product based on Example 4.1.

Figure 6 shows the performance of zig-zag product LDPC codes based on Example 4.2, with sum-product decoding. Once again, this performance is compared with the analogous performance of a LDPC code based on a random graph using identical sub-code constraints and having the same block length and rate. These results are also compared with a $(3, 6)$ regular LDPC code that uses simple parity check constraints. For the parameters $p = 3$ and $k = 5$ in Example 4.2, a bipartite graph, based on the zig-zag product graph, on 768 vertices with degree 20 is obtained. Using the $[20, 15, 4]$ sub-code constraints as earlier, a block length 7680 rate 1/2 LDPC code is obtained. This code performs comparably with the random LDPC code that is based on a degree 20 randomly designed graph. Using the parameters $p = 5$ and $k = 4$ and a $[16, 12, 2]$ sub-code,

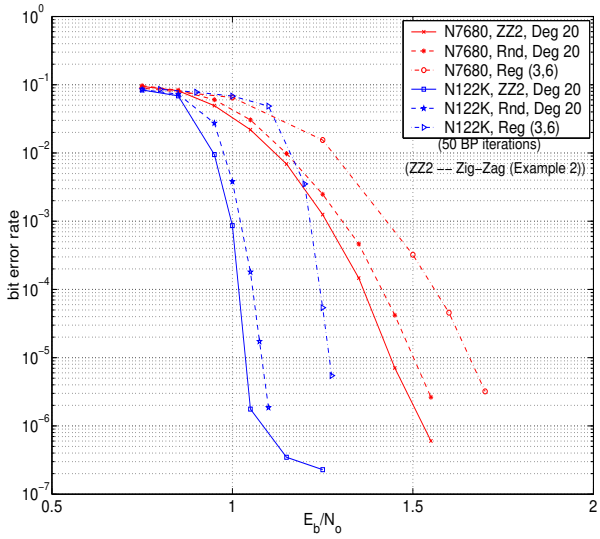


Fig. 6. LDPC codes from zig-zag product graphs based on Example 4.2.

a longer block length 122,880 LDPC code is obtained. As in the previous case, this code also performs comparably, if not, better than its random counterpart for low to medium signal-to-noise ratios (SNRs). Once again, we attribute its slightly inferior performance at high SNRs to the poor girth of the zig-zag product graph.

Figure 7 shows the performance of a replacement product LDPC code based on Example 4.2, with sum-product decoding. Once again, this performance is compared with the analogous performance of an LDPC code based on a random graph using identical sub-code constraints and having the same block length and rate. For the parameters $p = 5$ and $k = 13$ in Example 4.2, a bipartite graph, based on the replacement product graph, on $15,360$ vertices with degree 15 is obtained. Using the $[15, 11, 3]$ Hamming code as a sub-code in the replacement product graph, a block length 115,200 rate 0.4667 LDPC code is obtained. The performance of the replacement product LDPC code is inferior to that of the random code in this example due to the poor choice of the generators in the component Cayley graphs. We believe a more judicious choice would improve the performace considerably.

Figure 8 shows the performance of LDPC codes designed based on the zig-zag product of two unbalanced bipartite graphs as in Example 4.3. A $(6, 10)$-regular bipartite graph on $(20, 12)$ vertices is chosen as one of the component graphs and a $(3, 5)$-regular bipartite graph on $(10, 6)$ vertices is chosen as the other component. Their zig-zag product is a $(9, 25)$-regular bipartite graph on $(200, 72)$ vertices. Using sub-code constraints of two codes – a $[9, 6, 2]$ and a $[25, 21, 2]$ linear block code – a block length 1800 LDPC code of rate 0.5066

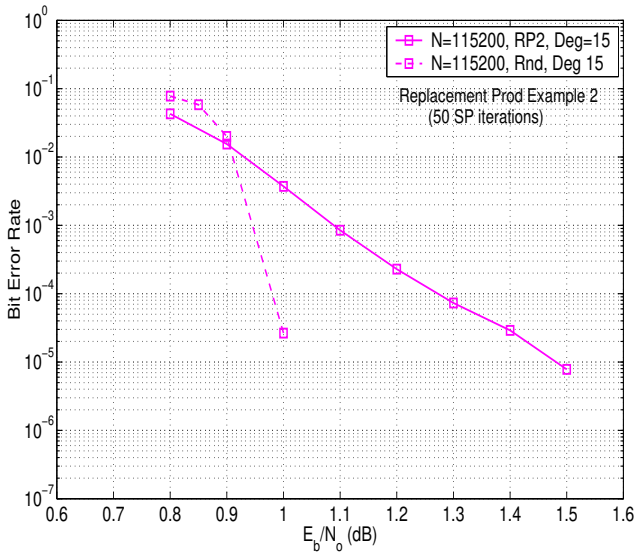


Fig. 7. LDPC code from replacement product based on Example 4.2.

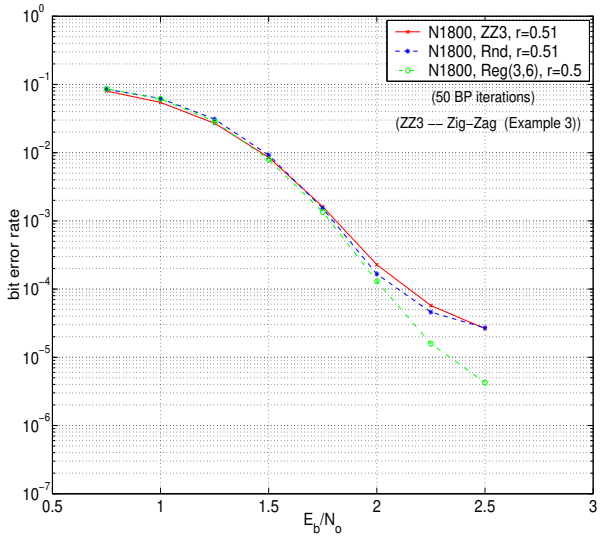


Fig. 8. LDPC code from the unbalanced bipartite zig-zag product graph based on Example 4.3.

is obtained. The performance of this code is compared with a LDPC code based on a random $(9, 25)$-regular bipartite graph using the same sub-code constraints, and also with a block length 1800 random $(3, 6)$ regular LDPC code. All three codes perform comparably, with the random $(3, 6)$ showing a small improvement over others at high SNRs. Given that the zigzag product graph is composed of two very small graphs, this result highlights the fact that good graphs may be designed using just simple component graphs.

VI. ITERATIVE CONSTRUCTION OF GENERALIZED PRODUCT GRAPHS

In this section, we introduce iterative families of expanders that address an important design problem in graph theory and that have several other practical engineering applications such as in designing communication networks, complexity theory, and derandomization techniques.

For code constructions, we would ideally use products that could be iterated to generate families of LDPC codes having a slow growth in the number of vertices (so as to get codes for many blocklengths), while maintaining a constant (small) degree. The iterative families described in this section have these characteristics, but unfortunately do not have parameters that make the codes practical. Designing such iterative constructions suitable for coding is a nice open problem.

First we review the iteration scheme of [13] for the original zig-zag product starting from a seed graph H. The existence of the seed graph H as well as explicit examples of suitable seed graphs for H are also discussed in [13]. We present new iterative constructions of a modified unbalanced bipartite zig-zag product and the replacement product thereafter.

A. Iterative construction of original zig-zag product graphs

We will need a squaring operation and the zig-zag operation in the iterative technique that is proposed next. Note that for a graph G, its square G^2 is a graph whose vertices are the same as in G and whose edges are paths of length two in G. Further, if G is a (N, D, λ) graph, then G^2 is a (N, D^2, λ^2) graph.

A graph H is used to serve as the basic building block for the iteration. Let H be any $(D^4, D, \frac{1}{5})$ graph. Then the iteration is defined by

$$G_1 = H^2 = (D^4, D^2, \frac{1}{25}).$$

$$G_{i+1} = G_i^2 \textcircled{z} H.$$

The above iterative construction indeed gives a family of expanders as presented in the following result:

Theorem 6.1: [13] For every i, G_i is an $(D^{4i}, D^2, \frac{2}{5})$ graph.

B. Iterative construction of unbalanced bipartite zig-zag product graphs

The unbalanced bipartite zig-zag product presented in Section III C cannot be used directly to obtain an iterative construction, due to constraints on the parameters[4]. Therefore, we slightly modify the zig-zag product by intoducing an additional step on the small component graph in the product construction. We note that the introduction of this additional step can only increase the expansion of the zig-zag product graph. However, this increase in expansion is at the cost of increasing the degree of the graph slightly. The new modified unbalanced bipartite zig-zag product is presented next, followed by an iterative construction that uses this product.

[4] The only parameters that were compatible were for the special case where the bipartite components were balanced.

1) Modified unbalanced bipartite zig-zag product: The two component graphs are unbalanced bipartite graphs, i.e., the two sets of vertices have different degrees. Let G_1 be a (c_1, d_1)-regular graph on the vertex sets V_1, W_1, where $|V_1| = N$ and $|W_1| = M$. Let G_2 be a (c_2, d_2)-regular graph on the vertex sets V_2, W_2, where $|V_2| = d_1$ and $|W_2| = c_1$. Let λ_1 and λ_2 denote the second largest eigenvalues of the normalized adjacency matrices of G_1 and G_2, respectively. Again, randomly number the edges around each vertex $\tilde{v}$ in G_1 and G_2 by $\{1, \ldots, deg(\tilde{v})\}$, where $deg(\tilde{v})$ is the degree of $\tilde{v}$. Then the zig-zag product graph, which we will denote by $G = G_1 (\textcircled{z}_M) G_2$, is a $(c_2^2 d_2, d_2^2 c_2)$-regular bipartite graph on the vertex sets V, W with $|V| = N \cdot d_1$, $|W| = M \cdot d_1$, formed in the following manner:

- Every vertex $v \in V_1$ and $w \in W_1$ of G_1 is replaced by a copy of G_2. The cloud at a vertex $v \in V_1$ has vertices V_2 on the left and vertices W_2 on the right, with each vertex from W_2 corresponding to an edge from v in G_1. The cloud at a vertex $w \in W_1$ is similarly structured with each vertex in V_2 in the cloud corresponding to an edge of w in G_1. (See Figure 2.) Then the vertices from V are represented as ordered pairs (v, k), for $v \in \{1, \ldots, N\}$ and $k \in \{1, \ldots, d_1\}$, and the vertices from W are represented as ordered pairs (w, ℓ), for $w \in \{1, \ldots, M\}$ and $\ell \in \{1, \ldots, c_1\}$.
- A vertex $(v, k) \in V$ is connected to a vertex in W by making four steps in the product graph. The first three steps are the same as in Section 2C. The fourth step is:
  - A second small step from right to left in the local copy of G_2. This is a step $(v[k[i]], \ell[j]) \rightarrow (v[k[i]], \ell[j][j'])$, where the final vertex is in W, for $j' \in \{1, \ldots, d_2\}$.

  Therefore, there is an edge between (v, k) and $(v[k[i]], \ell[j][j'])$.

Theorem 6.2: Let G_1 be a (c_1, d_1)-regular bipartite graph on (N, M) vertices with $\lambda(G_1) = \lambda_1$, and let G_2 be a (c_2, d_2)-regular bipartite graph on (d_1, c_1) vertices with $\lambda(G_2) = \lambda_2$. Then, the modified zig-zag product graph $G_1 (\textcircled{z}_M) G_2$ is a $(c_2^2 d_2, c_2 d_2^2)$-regular bipartite on $(N \cdot d_1, M \cdot d_1)$ vertices with $\lambda = \lambda(G_1 (\textcircled{z}_M) G_2) \leq \lambda_1 + \lambda_2 + \lambda_2^2$. Moreover, if $\lambda_1 < 1$ and $\lambda_2 < 1$, then $\lambda = \lambda(G_1 (\textcircled{z}_M) G_2) < 1$.

The proof is omitted but may be seen intuitively given the expansion of the original unbalanced zig-zag product (Theorem 3.2) in the following way. The new step is independent of the previous steps and is essentially a random step on an expander graph (G_2). Considering a distribution on the vertices (v, k) of $G = G_1 (\textcircled{z}_M) G_2$, if the distribution of k conditioned on v is close to uniform after step 3, then step 4 is redundant and no gain is made. If the distribution of k conditioned on v is not close to uniform after step 3, then step 4 will increase the entropy of k by the expansion of G_2.

2) Iterative construction: So the modified zigzag product of $G_1 = (N, M, c_1, d_1, \lambda_1)$ and $G_2 = (d_1, c_1, c_2, d_2, \lambda_2)$ is $G = G_1 (\textcircled{z}_M) G_2$, that is $(c_2^2 d_2, c_2 d_2^2)$-regular on $(N d_1, M d_1)$ vertices. For the iteration, let H be any $(N = c_2^4 d_2^5, M = c_2^5 d_2^4, c_2, d_2, \lambda)$ expander graph, Then, the iteration is defined by

$$G_1 = H^3 = (N, M, c_2^2 d_2, c_2 d_2^2, \lambda^3).$$

$$G_{i+1} = G_i^3 (\textcircled{z}_M) H.$$

We show that the above iterative technique yields a family of expanders in the following

Theorem 6.3: Let H be a $(c_2d_2^2, c_2^2d_2, c_2, d_2, \lambda)$ graph, where $\lambda \leq 0.296$. Let $G_1 = H^3$ and $G_{i+1} = G_i^3 \textcircled{z} H$. Then the i-th iterated zig-zag product graph G_i is a $((c_2^{4i}d_2^{5i}, c_2^{4i+1}d_2^{5i-1}, c_2^2d_2, c_2d_2^2, \lambda')$ graph, where $\lambda' \leq 0.55$.

Proof: Let n_i and m_i be the number of left vertices and right vertices in G_i, respectively. Since $G_i = G_{i-1}^3 \textcircled{z} H$, we have $n_i = n_{i-1}(c_2^4d_2^5)$ and $m_i = m_{i-1}(c_2^4d_2^5)$. Since $n_1 = c_2^4d_2^5$ and $m_1 = c_2^5d_2^4$, it follows from the above recursion that $n_i = c_2^{4i}d_2^{5i}$ and $m_i = c_2^{4i+1}d_2^{5i-1}$. Note that G_i is always $(c_2^2d_2, c_2d_2^2)$-regular.
Let λ_i be the normalized second eigenvalue of G_i. Using the result from Theorem 6.2, we have

$$\lambda_i \leq \lambda_{i-1}^3 + \lambda + \lambda^2.$$

Further note that $\lambda_1 = \lambda^3$. Observe that even for $\lambda_i = \lambda_{i-1}^3 + \lambda + \lambda^2$, the series converges $\lambda_i \to 0.5499$ when $\lambda \leq 0.296$. Hence, for each iteration i, $\lambda_i \leq 0.55$, thereby yielding a family of expanders. ■

C. Iterative construction of replacement product graphs

The replacement product of $G_1 = (N, d_1, \lambda_1)$ and $G_2 = (d_1, d_2, \lambda_2)$, denoted by $G_1 \textcircled{R} G_2$, is an $(Nd_1, d_2 + 1, \lambda)$ graph. In [13], it is shown that the expansion of the replacement product graph is given by

$$\lambda \leq (p + (1-p)f(\lambda_1, \lambda_2))^{\frac{1}{3}}, \tag{1}$$

$$\text{where } p = \frac{d_2^2}{(d_2+1)^3} \text{ and } f(\lambda_1, \lambda_2) = \frac{1}{2}(1-\lambda_2^2)\lambda_1 + \frac{1}{2}\sqrt{(1-\lambda_2^2)^2\lambda_1^2 + 4\lambda_2^2}.$$

To obtain an iterative construction, we choose two graphs $G_1 = (N, (d+1), \lambda_1)$ and $H = ((d+1)^4, d, \lambda_2)$.

The iteration is defined by

$$G_{i+1} = (G_i)^4 \textcircled{R} H.$$

We show that the above iterative construction results in a family of expanders.

Theorem 6.4: Let G_1 be a $(N, (d+1), \lambda_1)$ graph and let H be a $((d+1)^4, d, \lambda_2)$ graph, where $\lambda_1 \leq 0.2, \lambda_2 \leq 0.2$ and $d \geq 6$. Let $G_{i+1} = (G_i)^4 \textcircled{R} H$. Then the i-th iterated replacement product graph G_i is a $(N(d+1)^{4(i-1)}, d+1, \lambda)$ graph, where $\lambda \leq 0.86$.

Proof: Let n_i be the number of vertices in G_i. Then $n_i = n_{i-1}(d+1)^4$. Since $n_1 = N$, it follows that $n_i = N(d+1)^{4i-4}$. It is clear that the degree of G_i is one more than the degree of H, and thus, G_i is $(d+1)$-regular. Let λ_i be the normalized second eigenvalue of G_i. Using the result from Equation (1), we have

$$\lambda_i \leq [p + (1-p)f(\lambda_{i-1}^4, \lambda_2)]^{\frac{1}{3}},$$

where $p = \frac{d^2}{(d+1)^3}$ and f is as above. Using numerical methods with Matlab, it was verified that λ_i coverges to 0.8574 when $\lambda_1 \leq 0.2, \lambda_2 \leq 0.2$, and $d \geq 6$. Hence, for each iteration i, $\lambda_i < 0.86$, thereby yielding a family of expanders. ■

Note that if the above iteration was defined to be $G_{i+1} = (G_i)^2 \textcircled{R} H$, then for no choice of λ_1, λ_2, or d, would the resulting iterative family be expanders.

VII. CONCLUSIONS

In this paper we generalized the zig-zag product resulting in an unbalanced bi-partite graph. We proved that the resulting graph is an expander graph as long as the component graphs are expanders as well. Using the bi-regular zig-zag product and the replacement product we introduced new families of iteratively constructed expander graphs with constant degrees.

We investigated the performance of LDPC codes obtained from zig-zag and replacement product graphs. The resulting product LDPC codes perform comparably if not better than random LDPC codes. In contrast to random LDPC codes one has a very compact description of these codes. We conclude that codes from product graphs provide a nice avenue for code constructions.

APPENDIX

Theorem 3.2 *Proof:* Let M_G denote the adjacency matrix of G. For convenience, we also let G_2 denote the $d_1 \times c_1$ matrix that describes the connections between the nodes in V_2 to the nodes in W_2 for the graph G_2, and G_1 denote the $N \times M$ matrix that describes the connections between the nodes in V_1 and the nodes in W_1 for the graph G_1. This means that the adjacency matrix for the graph G_2 is given by $M_2 = \begin{bmatrix} 0 & G_2 \\ G_2^T & 0 \end{bmatrix}$, and the adjacency matrix for the graph G_1 is given by $M_1 = \begin{bmatrix} 0 & G_1 \\ G_1^T & 0 \end{bmatrix}$.

The adjacency matrix for the zig-zag product graph G is given by

$$M_G = \begin{bmatrix} 0 & (G_2 \otimes I_n)\tilde{A}_2(G_2^T \otimes I_m) \\ (G_2^T \otimes I_m)\tilde{A}_2(G_2 \otimes I_n) & 0 \end{bmatrix},$$

where $\tilde{A}_2$ is a permutation matrix of size $Nc_1 \times Nc_1$ that describes the zig-zag product connections.

The largest eigenvalue of M_G is $c_2 d_2$ and the corresponding eigenvector is $v_0 = [1 \ \cdots \ 1 \ r \ \cdots \ r]^T$, where the first Nd_1 components are equal to 1 and the remaining Mc_1 components are equal to $r = \frac{d_2}{c_2} = \frac{d_1}{c_1}$.

Let 1_x denote a column vector of length x with all entries equal to 1. Then, the largest eigenvalue of M_1 is $\sqrt{c_1 d_1}$ and the corresponding eigenvector is $w_0 = \begin{bmatrix} 1_N \\ r_1 1_M \end{bmatrix}$, where $r_1 = \sqrt{\frac{d_1}{c_1}} = \sqrt{r}$. Similarly, the largest eigenvalue of M_2 is $\sqrt{c_2 d_2}$ and the corresponding eigenvector is $u_0 = \begin{bmatrix} 1_{d_1} \\ r_2 1_{c_1} \end{bmatrix}$, where $r_2 = \sqrt{\frac{d_2}{c_2}} = r_1 = \sqrt{r}$.

Let $\lambda_2 = \max_{u \perp u_0} \frac{<M_2 u, u>}{<u, u>} = \max_{u \perp u_0} \frac{2(u_a^T G_2 u_b)}{\|u_a\|^2 + \|u_b\|^2}$, where $u = \begin{bmatrix} u_a \\ u_b \end{bmatrix}$. That is u_a is a column vector of length d_1 corresponding to the vertices in V_2 and u_b is a column vector of length c_1 corresponding to the vertices in W_2. We choose u such that $u \perp u_0$. Furthermore, u can be written as two vectors $u^{\parallel}$ and $u^{\perp}$ where $u^{\parallel}$ is a vector that is parallel to the constant (non-zero) vector and $u^{\perp}$ is a vector that is perpendicular (or, orthogonal) to the constant (non-zero) vector. That is $u = u^{\parallel} + u^{\perp}$.

(Note that, by definition, λ_2 corresponds to the second largest eigenvalue of M_2 and the corresponding eigenvector u that maximizes λ_2 in the above is orthogonal to the u_0, the eigenvector corresponding to the eigenvalue $\sqrt{c_2 d_2}$.)

Similarly, let $\lambda_1 = \max_{w \perp w_0} \frac{<M_1 w, w>}{<w, w>} = \max_{w \perp w_0} \frac{2(w_a^T G_1 w_b)}{\|w_a\|^2 + \|w_b\|^2}$, where $w = \begin{bmatrix} w_a \\ w_b \end{bmatrix}$. (Here, w_a is a column vector of length N and w_b is a column vector of length M. w can also be broken down as $w = w^{\|} + w^{\perp}$ as above. By definition, λ_1 is the second largest eigenvalue of M_1.)

The eigenvector of M_G corresponding to the largest eigenvalue $c_2 d_2$ is $v_0 = \begin{bmatrix} 1_{Nd_1} \\ r 1_{Mc_1} \end{bmatrix}$. Let $\alpha = \begin{bmatrix} \alpha_a \\ \alpha_b \end{bmatrix}$ be an eigenvector of M_G, where α_a has length Nd_1 and α_2 has length Mc_1. Let e_m be a basis vector with component value 1 at the mth entry and component value 0 elsewhere. Then, the vectors α_a and α_b can be written as $\alpha_a = \sum_{n \in [N]} \alpha_{a_n} \otimes e_n$, and $\alpha_b = \sum_{m \in [M]} \alpha_{b_m} \otimes e_m$, where $\otimes$ denotes the Kronecker product, α_{a_n} is the vector α_a restricted to the components corresponding to the vertices in the n^{th} vertex cloud, $n \in [N]$, of the graph G, and α_{b_m}, for $m \in [M]$, is defined similarly.

Then the second largest eigenvalue of M_G is $\lambda = \max_{\alpha \perp v_0} \frac{2\alpha_a^T (K) \alpha_b}{<\alpha, \alpha>}$, where $K = (G_2 \otimes I_N) \tilde{A}_2 (G_2 \otimes I_M)$.

Let $s = |2(\alpha_a^T K \alpha_b)|$. Splitting α_{a_n} (and, α_{b_m}) into parallel and perpendicular parts, $\alpha_{a_n} = \alpha_{a_n}^{\|} + \alpha_{a_n}^{\perp}$, we can write

$$s = 2\Big(\sum_{n \in [N]} ((\alpha_{a_n}^{\|})^T G_2 \otimes e_n) + \sum_{n \in [N]} (\alpha_{a_n}^{\perp})^T G_2 \otimes e_m \Big) \tilde{A}_2 \Big(\sum_{m \in [M]} G_2 \alpha_{b_m}^{\|} \otimes e_m + \sum_{m \in [M]} g_2 \alpha_{b_m}^{\perp} \otimes e_m \Big)$$

We want to show that $s \leq f(\lambda_1, \lambda_2)(\| \alpha_a \|^2 + \| \alpha_b \|^2)$, where $f(\lambda_1, \lambda_2)$ is some positive-valued function such that $f(\lambda_1, \lambda_2) \leq \lambda_1 + \lambda_2 + \lambda_2^2$. Note that $\alpha_a \in \mathbb{R}^{Nd_1}$, and $\alpha_b \in \mathbb{R}^{Mc_1}$.

Observe the following:

1) $\lambda_2 = \max_{u \perp \begin{bmatrix} 1_{d_1} \\ \sqrt{r} 1_{c_1} \end{bmatrix}} \frac{2u_a^T g_2 u_b}{\|u_a\|^2 + \|u_b\|^2}$, where $u = \begin{bmatrix} u_a \\ u_b \end{bmatrix}$.
2) $\lambda_1 = \max_{u \perp \begin{bmatrix} 1_N \\ \sqrt{r} 1_M \end{bmatrix}} \frac{2w_a^T g_1 w_b}{\|w_a\|^2 + \|w_b\|^2}$, where $w = \begin{bmatrix} w_a \\ w_b \end{bmatrix}$.
3) $\alpha_a = \begin{bmatrix} \alpha_{a_1} \\ \vdots \\ \alpha_{a_N} \end{bmatrix}$, where $\alpha_{a_i} \in \mathbb{R}^{d_1}, i \in [N]$, $\alpha_b = \begin{bmatrix} \alpha_{b_1} \\ \vdots \\ \alpha_{b_M} \end{bmatrix}$,
   where $\alpha_{b_i} \in \mathbb{R}^{c_1}, i \in [M]$, and $\alpha = \begin{bmatrix} \alpha_a \\ \alpha_b \end{bmatrix} \in \mathbb{R}^{Nd_1 + Mc_1}$.
4) From the definition of λ_2, we have

$$\| \begin{bmatrix} 0 & G_2 \\ G_2^T & 0 \end{bmatrix} \begin{bmatrix} \alpha_{a_n} \\ 0 \end{bmatrix} \| \leq \lambda_2 | \begin{bmatrix} \alpha_{a_n} \\ 0 \end{bmatrix} |$$

$$\Rightarrow \| G_2^T \alpha_{a_n} \| \leq \lambda_2 \| \alpha_{a_n}^{\perp} \| \text{ and } \| G_2^T \alpha_{a_n} \| \leq \lambda_2 \| \alpha_{a_n} \| .$$

This implies that $\begin{bmatrix} \alpha_{a_n} \\ 0 \end{bmatrix} \perp \begin{bmatrix} 1_{d_1} \\ \sqrt{r} 1_{c_1} \end{bmatrix}$.

5) Since $\alpha_{a_n}^{\perp}$ is orthogonal to the constant vector, we have $\alpha_{a_n}^{\perp} \perp 1_{d_1}$.

Rewriting, we have that

$$s = 2\Big(\sum_{n\in[N]} (\alpha_{a_n}^{\parallel})^T G_2 \otimes e_n \Big) \tilde{A}_2 \Big(\sum_{m\in[M]} G_2 \alpha_{b_m}^{\parallel} \otimes e_m \Big) + 2\Big(\sum_{n\in[N]} (\alpha_{a_n}^{\parallel})^T G_2 \otimes e_n \Big) \tilde{A}_2 \Big(\sum_{m\in[M]} G_2 \alpha_{b_m}^{\perp} \otimes e_m \Big)$$
$$+2\Big(\sum_{n\in[N]} (\alpha_{a_n}^{\perp})^T G_2 \otimes e_n \Big) \tilde{A}_2 \Big(\sum_{m\in[M]} G_2 \alpha_{b_m}^{\parallel} \otimes e_m \Big) + 2\Big(\sum_{n\in[N]} (\alpha_{a_n}^{\perp})^T G_2 \otimes e_n \Big) \tilde{A}_2 \Big(\sum_{m\in[M]} G_2 \alpha_{b_m}^{\perp} \otimes e_m \Big).$$

So $s = s_1 + s_2 + s_3 + s_4$, where

$$s_1 = 2\Big(\sum_{n\in[N]} (\alpha_{a_n}^{\parallel})^T G_2 \otimes e_n \Big) \tilde{A}_2 \Big(\sum_{m\in[M]} G_2 \alpha_{b_m}^{\parallel} \otimes e_m \Big)$$

$$s_2 = 2\Big(\sum_{n\in[N]} (\alpha_{a_n}^{\parallel})^T G_2 \otimes e_n \Big) \tilde{A}_2 \Big(\sum_{m\in[M]} G_2 \alpha_{b_m}^{\perp} \otimes e_m \Big)$$

$$s_3 = 2\Big(\sum_{n\in[N]} (\alpha_{a_n}^{\perp})^T G_2 \otimes e_n \Big) \tilde{A}_2 \Big(\sum_{m\in[M]} G_2 \alpha_{b_m}^{\parallel} \otimes e_m \Big), \text{ and}$$

$$s_4 = 2\Big(\sum_{n\in[N]} (\alpha_{a_n}^{\perp})^T G_2 \otimes e_n \Big) \tilde{A}_2 \Big(\sum_{m\in[M]} G_2 \alpha_{b_m}^{\perp} \otimes e_m \Big).$$

We will bound each part of s separately:

1) Since $\tilde{A}_2$ is a permutation matrix, we have

$$s_4 \leq 2 \| \sum_{n\in[N]} (\alpha_{a_n}^{\perp})^T G_2 \otimes e_n \| \| \sum_{m\in[M]} G_2 \alpha_{b_m}^{\perp} \otimes e_m \|$$

Since $\| (\alpha_{a_n}^{\perp})^T G_2 \| \leq \lambda_2 \| \alpha_{a_n}^{\perp} \|$ by definition of λ_2 (and similarly, $\| G_2 \alpha_{b_m}^{\perp} \| \leq \lambda_2 \| \alpha_{b_n}^{\perp} \|$), we have

$$s_4 \leq 2 \| \sum_{n\in[N]} \lambda_2 \alpha_{a_n}^{\perp} \otimes e_n \| \| \lambda_2 \sum_{m\in[M]} \alpha_{b_m}^{\perp} \otimes e_m \|$$
$$= 2\lambda_2^2 \| \alpha_a^{\perp} \| \| \alpha_b^{\perp} \| \leq \lambda_2^2 (\| \alpha_a^{\perp} \|^2 + \| \alpha_b^{\perp} \|^2) = \lambda_2^2 (\| \alpha^{\perp} \|^2)$$

2) Since $\tilde{A}_2$ is a permutation matrix, we have

$$s_3 \leq 2 \| \sum_{n\in[N]} (\alpha_{a_n}^{\perp})^T G_2 \otimes e_n \| \| \sum_{m\in[M]} G_2 \alpha_{b_m}^{\parallel} \otimes e_m \|$$

Using the argument from the previous step and since $\| G_2\alpha^{\|}_{b_m} \| \leq \| \alpha^{\|}_{b_m} \|$, we have

$$s_3 \leq 2\lambda_2 \| \alpha^{\perp}_a \| \| \alpha^{\|}_b \|$$

3) Similarly, we can show that

$$s_2 \leq 2\lambda_2 \| \alpha^{\|}_a \| \| \alpha^{\perp}_b \|$$

4) To upper bound s_1, define a new vector $C(\alpha'_a)$ for every vector α'_a such that its mth component is

$$(C(\alpha'_a))_m := \frac{1}{d_1}\sum_{a=1}^{d_1} \alpha_{a'_m}, \text{ for } m \in [M]$$

This implies that $\alpha'^{\|}_a = C(\alpha'_a) \otimes \frac{1d_1}{d_1}$.

Similarly, define a new vector $C'(\alpha'_b)$ for every α'_b as

$$(C'(\alpha'_b))_n := \frac{1}{c_1}\sum_{b=1}^{c_1} \alpha'_{b_n}, \text{ for } n \in [N]$$

This implies that $\alpha'^{\|}_b = C'(\alpha'_b) \otimes \frac{1c_1}{c_1}$.

That is, the functions $C(\cdot)$ and $C'(\cdot)$ computes the average value of the components in each vertex cloud of the zig-zag product graph G.

Therefore, we have $C'\tilde{A}_2(e_m \otimes \frac{1d_1}{d_1}) = G_1 e_m$. Note that $\alpha^{\|}_{b_m} G_2 = \alpha^{\|}_{\tilde{a}_m}$, where $\tilde{a}_m$ in the subscript refers to the left vertices on the right of the zig-zag product graph that are used for the construction but do not belong to the vertex set of the zig-zag product graph. Rewriting s_1, we have

$$s_1 = 2(\sum_{n\in[N]} (\alpha^{\|}_{a_n})^T G_2 \otimes e_n)^T \tilde{A}_2 (\sum_{m\in[M]} \alpha^{\|}_{b_m} G_2 \otimes e_m) = 2(\sum_{n\in[N]} \alpha^{\|}_{\tilde{b}_n} \otimes e_n)^T \tilde{A}_2 (\sum_{m\in[M]} \alpha^{\|}_{\tilde{a}_m} \otimes e_m).$$

This is because, $(\alpha^{\|}_{a_n})^T G_2 = \alpha^{\|}_{\tilde{b}_n}$, the components of the right vertices of the G_2 clouds on the left of G, and $G_2(\alpha^{\|}_{b_m}) = \alpha^{\|}_{\tilde{a}_m}$, the components corresponding to the left vertices of the G_2 clouds on the right of G. (That is, since G_2 denotes the connections between the left vertices and right vertices, multiplying with G_2 takes $(\alpha^{\|}_{a_n})^T$ to $(\alpha^{\|}_{\tilde{b}_n})$ and $(\alpha^{\|}_{b_m})$ to $(\alpha^{\|}_{\tilde{a}_m})$.)

But $\alpha^{\|}_a = (C(\alpha_a)) \otimes \frac{1d_1}{d_1}$, $\alpha^{\|}_{\tilde{b}} = (C'(\alpha_{\tilde{b}}) \otimes \frac{1c_1}{c_1}$. Hence,

$$s_1 = 2(C'(\alpha_{\tilde{b}}) \otimes \frac{1c_1}{c_1})^T \tilde{A}_2 (C(\alpha_{\tilde{a}}) \otimes \frac{1d_1}{d_1})$$

$$\Rightarrow s_1 = 2(C(\alpha_{\tilde{b}}))^T G_1 (C(\alpha_{\tilde{a}}))/(c_1 d_1) = \frac{2(\frac{1}{\sqrt{c_1}}C'(\alpha_{\tilde{b}}))^T G_1 (\frac{1}{\sqrt{d_1}}C(\alpha_{\tilde{a}}))}{\sqrt{c_1}\sqrt{d_1}}$$

But observe that $\begin{bmatrix} \frac{1}{\sqrt{c_1}}C'(\alpha_{\tilde{b}}) \\ \frac{1}{\sqrt{d_1}}C(\alpha_{\tilde{a}}) \end{bmatrix}$ is orthogonal to the vector $\begin{bmatrix} 1_N \\ \sqrt{r}1_M \end{bmatrix}$. This is because

$$< \begin{bmatrix} \frac{1}{\sqrt{c_1}}C'(\alpha_{\tilde{b}}) \\ \frac{1}{\sqrt{d_1}}C(\alpha_{\tilde{a}}) \end{bmatrix}, \begin{bmatrix} 1_N \\ \sqrt{r}1_M \end{bmatrix} >= \frac{1}{\sqrt{c_1}}\sum_{b=1}^{c_1}\sum_{n=1}^{N} \alpha_{\tilde{b}_n} + \frac{1}{\sqrt{d_1}}\frac{\sqrt{d_1}}{\sqrt{c_1}}\sum_{a=1}^{d_1}\sum_{m=1}^{M} \alpha_{\tilde{a}_m} \quad (**)$$

However, $\sum_{b=1}^{c_1}\sum_{n=1}^{N}\alpha_{b_n} = c_2\sum_{a=1}^{d_1}\sum_{n=1}^{N}\alpha_{a_n}$ and $\sum_{a=1}^{d_1}\sum_{m=1}^{M}\alpha_{a_m} = d_2\sum_{b=1}^{c_1}\sum_{m=1}^{M}\alpha_{b_m}$. Since $\begin{bmatrix}\alpha_a\\ \alpha_b\end{bmatrix}$ was chosen to be orthogonal to $\begin{bmatrix}1_{Nd_1}\\ r1_{Mc_1}\end{bmatrix}$, it is easy to verify that the sum in $(**)$ is zero.

Hence, from the definition of λ_1, we have

$$s_1 = \frac{2(\frac{1}{\sqrt{c_1}}C'(\alpha_{\tilde{b}}))^T G_1(\frac{1}{\sqrt{d_1}}C(\alpha_{\tilde{a}}))}{c_1 d_1} \leq \frac{\lambda_1}{c_1 d_1}(\frac{1}{c_1}\parallel C'(\alpha_{\tilde{b}})\parallel^2 + \frac{1}{d_1}\parallel C(\alpha_{\tilde{a}})\parallel^2) \quad (*)$$

It is easy to verify that the RHS in $(*)$ can be upper bounded as

$$s_1 \leq RHS(*) \leq \lambda_1(\parallel \alpha_a^{\parallel}\parallel^2 + \parallel \alpha_b^{\parallel}\parallel^2) = \lambda_1(\parallel \alpha^{\parallel}\parallel^2)$$

Combining the upper bounds on s_1, s_2, s_3, s_4, we have

$$s = s_1 + s_2 + s_3 + s_4$$

$$s \leq \lambda_1(\parallel \alpha^{\parallel}\parallel^2) + 2\lambda_2(\parallel \alpha_a^{\perp}\parallel\parallel \alpha_b^{\parallel}\parallel + \parallel \alpha_a^{\parallel}\parallel\parallel \alpha_b^{\perp}\parallel) + \lambda_2^2(\parallel \alpha^{\perp}\parallel^2)$$

However, observe that

$$2\lambda_2(\parallel \alpha_a^{\perp}\parallel\parallel \alpha_b^{\parallel}\parallel + \parallel \alpha_a^{\parallel}\parallel\parallel \alpha_b^{\perp}\parallel) \leq$$

$$\lambda_2(\parallel \alpha_a^{\parallel}\parallel^2 + \parallel \alpha_a^{\perp}\parallel^2 + \parallel \alpha_b^{\parallel}\parallel^2 + \parallel \alpha_b^{\perp}\parallel^2) = \lambda_2(\parallel \alpha\parallel^2).$$

Further, $\parallel \alpha^{\parallel}\parallel^2 \leq \parallel \alpha\parallel^2$ and $\parallel \alpha^{\perp}\parallel^2 \leq \parallel \alpha\parallel^2$.

Thus, we have

$$s \leq (\lambda_1 + \lambda_2 + \lambda_2^2)(\parallel \alpha\parallel^2)$$

The second largest eigenvalue of M_G is defined as $\lambda = \max_{\alpha\perp v_0}\frac{s}{<\alpha,\alpha>}$, where $s = 2\alpha_a^T(G_2\otimes I_N)\tilde{A}_2(G_2\otimes I_M)\alpha_b$. Using the upper bound on s, we get $\lambda \leq \frac{(\lambda_1+\lambda_2+\lambda_2^2)(\|\alpha\|^2)}{\|\alpha\|^2} = \lambda_1+\lambda_2+\lambda_2^2$.

The only remaining step is to show that if $\lambda_1 < 1$ and $\lambda_2 < 1$, then $\lambda < 1$. Suppose $\lambda_1 < 1, \lambda_2 < 1$ and suppose $\parallel \alpha^{\perp}\parallel \leq \frac{1-\lambda_1}{3\lambda_2}\parallel \alpha\parallel$. Then, we can upper bound s as follows

$$s \leq \lambda_1\parallel \alpha^{\parallel}\parallel^2 + 2\lambda_2\parallel \alpha^{\parallel}\parallel\parallel \alpha^{\perp}\parallel + \lambda_2^2\parallel \alpha^{\perp}\parallel^2$$

$$\leq \parallel \alpha\parallel^2 + \frac{2(1-\lambda_1)}{3}\parallel \alpha\parallel^2 + \frac{(1-\lambda_1)^2}{9}\parallel \alpha\parallel^2 = (1-\frac{1-\lambda_1}{3})^2\parallel \alpha\parallel^2 \leq \parallel \alpha\parallel^2$$

Suppose $\parallel \alpha^{\perp}\parallel > \frac{1-\lambda_1}{3\lambda_2}\parallel \alpha\parallel$.

Then, notice that $s = 2(\alpha_a^{\parallel} + \alpha_a^{\perp})(\sum_n G_2^T\otimes e_n)A_2(\sum_m G_2\otimes e_m)(\alpha_b^{\parallel} + \alpha_b^{\perp})$. The RHS can be written as $2(\alpha_{\tilde{b}}^{\parallel} + \sum_n \alpha_{a_n}^{\perp}(G_2^T)\otimes e_n)A_2(\alpha_{\tilde{a}}^{\parallel} + \sum_m G_2\alpha_{b_m}^{\perp}\otimes e_m$. However, $\sum_n \alpha_{a_n}^{\perp}(G_2^T)\otimes e_n$ is orthogonal to $\alpha_{\tilde{a}}^{\parallel}$ and $\sum_m G_2\alpha_{b_m}^{\perp}\otimes e_m$ is orthogonal to $\alpha_{\tilde{b}}^{\parallel}$. Thus,

$$s = 2(\alpha_{\tilde{b}}^{\parallel})A_2(\alpha_{\tilde{a}}^{\parallel}) + 2(\sum_n \alpha_{a_n}^{\perp}(G_2^T)\otimes e_n)A_2(\sum_m G_2\alpha_{b_n}^{\perp}\otimes e_m)$$

From the previous arguments, we have

$$s \leq \lambda_1(\parallel \alpha^{\parallel}\parallel^2) + \lambda_2^2(\parallel \alpha^{\perp}\parallel^2)$$

$$= \lambda_1(\| \alpha \|^2 - \| \alpha^{\perp} \|^2) + \lambda_2^2 \| \alpha^{\perp} \|^2$$

$$\leq (\| \alpha \|^2 - \| \alpha^{\perp} \|^2) + \lambda_2^2 \| \alpha^{\perp} \|^2 = (1 - \frac{(1-\lambda_1)^2(1-\lambda_2^2)}{9}) \| \alpha \|^2 \leq \| \alpha \|^2$$

This completes the proof. ■

REFERENCES

[1] N. Alon. Eigenvalues and expanders. *Combinatorica*, 6(2):83–96, 1986. Theory of computing (Singer Island, Fla., 1984).

[2] N. Alon, A. Lubotzky, and A. Wigderson. Semi-direct product in groups and zig-zag product in graphs: connections and applications (extended abstract). In *42nd IEEE Symposium on Foundations of Computer Science (Las Vegas, NV, 2001)*, pages 630–637. IEEE Computer Soc., Los Alamitos, CA, 2001.

[3] W. Imrich and S. Klavžar. *Product graphs*. Wiley-Interscience Series in Discrete Mathematics and Optimization. Wiley-Interscience, New York, 2000. Structure and recognition, With a foreword by Peter Winkler.

[4] H. Janwa and A. K. Lal. On Tanner codes: minimum distance and decoding. *Appl. Algebra Engrg. Comm. Comput.*, 13(5):335–347, 2003.

[5] C. Kelley, J. Rosenthal, and D. Sridhara. Some new algebraic constructions of codes from graphs which are good expanders. In *Proc. of the 41-st Allerton Conference on Communication, Control, and Computing*, pages 1280–1289, 2003.

[6] C. Kelley and D. Sridhara. Eigenvalue bounds on the pseudocodeword weight of expander codes. Submitted to the Journal of Advances in Mathematics of Communication, September 2006.

[7] J. Lafferty and D. Rockmore. Codes and iterative decoding on algebraic expander graphs. In the Proceedings of ISITA 2000, Honolulu, Hawaii, available at http://www-2.cs.cmu.edu/afs/cs.cmu.edu/user/lafferty/www/pubs.html, November 2000.

[8] N. Linial and A. Wigderson. Expander graphs and their applications. Lecture notes of a course given at the Hebrew University, 2003. Available under http://www.math.ias.edu/∼avi/TALKS/.

[9] A. Lubotzky. *Discrete Groups, Expanding Graphs and Invariant Measures*. Birkhäuser Verlag, Basel, 1994. With an appendix by Jonathan D. Rogawski.

[10] A. Lubotzky, R. Phillips, and P. Sarnak. Ramanujan graphs. *Combinatorica*, 8(3):261–277, 1988.

[11] G. A. Margulis. Explicit group-theoretic constructions of combinatorial schemes and their applications in the construction of expanders and concentrators. *Problems Inform. Transmission*, 24(1):39–46, 1988. Translation from Problemy Peredachi Informatsii.

[12] R. Meshulam and A. Wigderson. Expanders in group algebras. *Combinatorica*, 2003. To appear.

[13] O. Reingold, S. Vadhan, and A. Wigderson. Entropy waves, the zig-zag graph product, and new constant-degree expanders. *Ann. of Math. (2)*, 155(1):157–187, 2002.

[14] J. Rosenthal and P. O. Vontobel. Constructions of LDPC codes using Ramanujan graphs and ideas from Margulis. In *Proc. of the 38-th Allerton Conference on Communication, Control, and Computing*, pages 248–257, 2000.

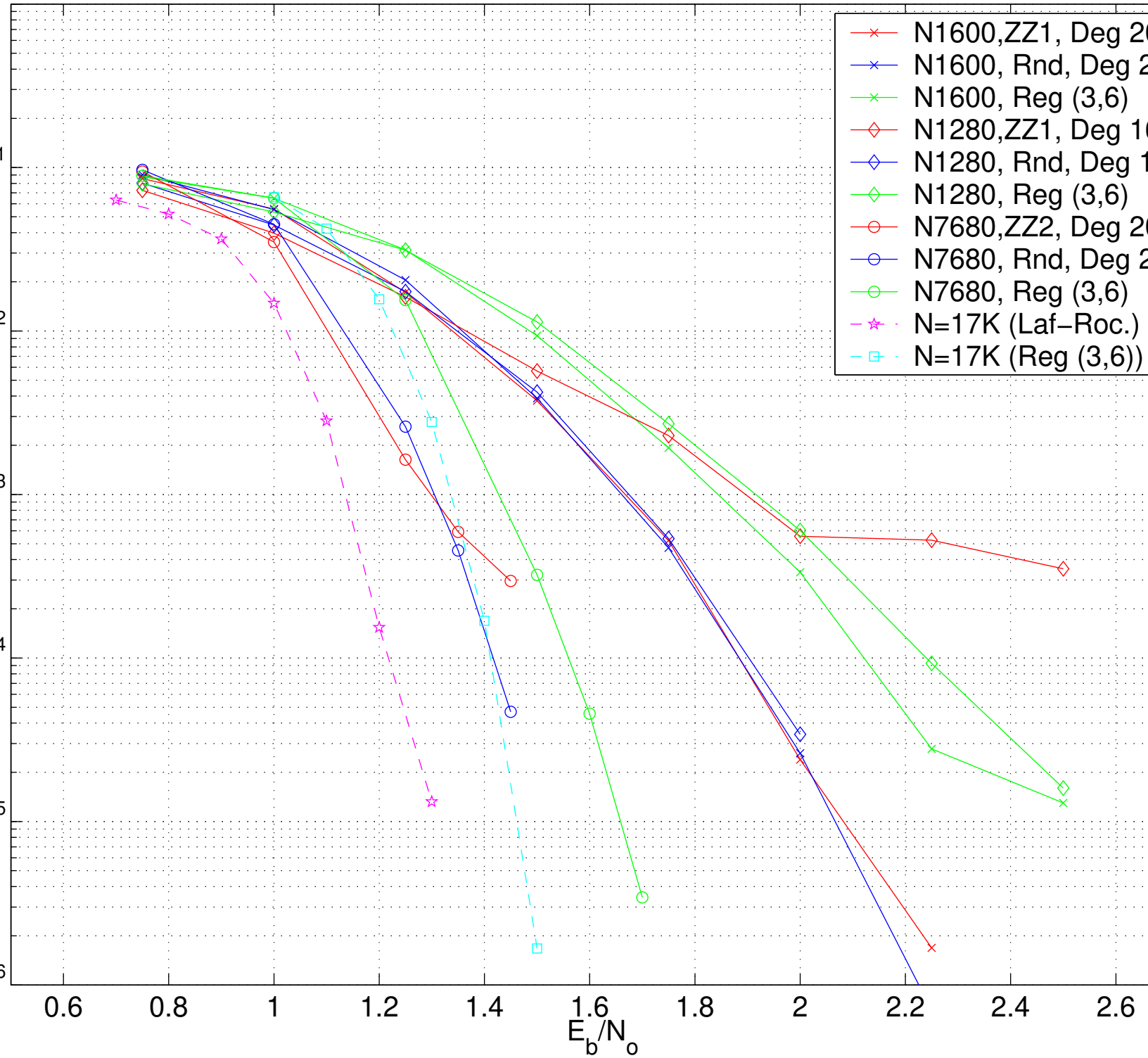
[15] M. Sipser and D. A. Spielman. Expander codes. *IEEE Trans. Inform. Theory*, 42(6, part 1):1710–1722, 1996.

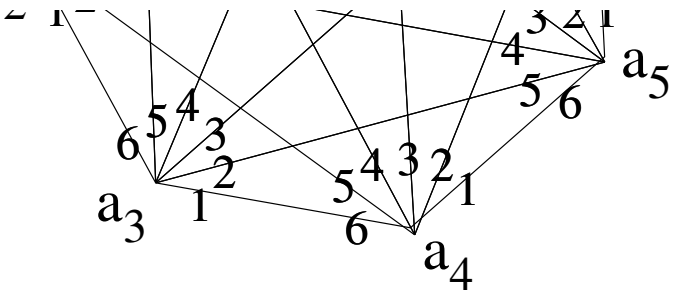
[16] R. M. Tanner. A recursive approach to low complexity codes. *IEEE Trans. Inform. Theory*, 27(5):533–547, 1981.

[17] R. M. Tanner. Explicit concentrators from generalized N-gons. *SIAM J. Algebraic Discrete Methods*, 5(3):287–293, 1984.

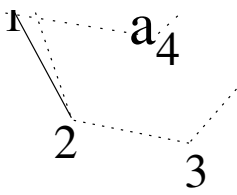
[18] J.-P. Tillich and G. Zémor. Optimal cycle codes constructed from Ramanujan graphs. *SIAM J. Discrete Math.*, 10(3):447–459, 1997.

[19] J. K. Wolf. Efficient maximum likelihood decoding of linear block codes using a trellis. *IEEE Trans. Inform. Theory*, IT-24(1):76–80, 1978.

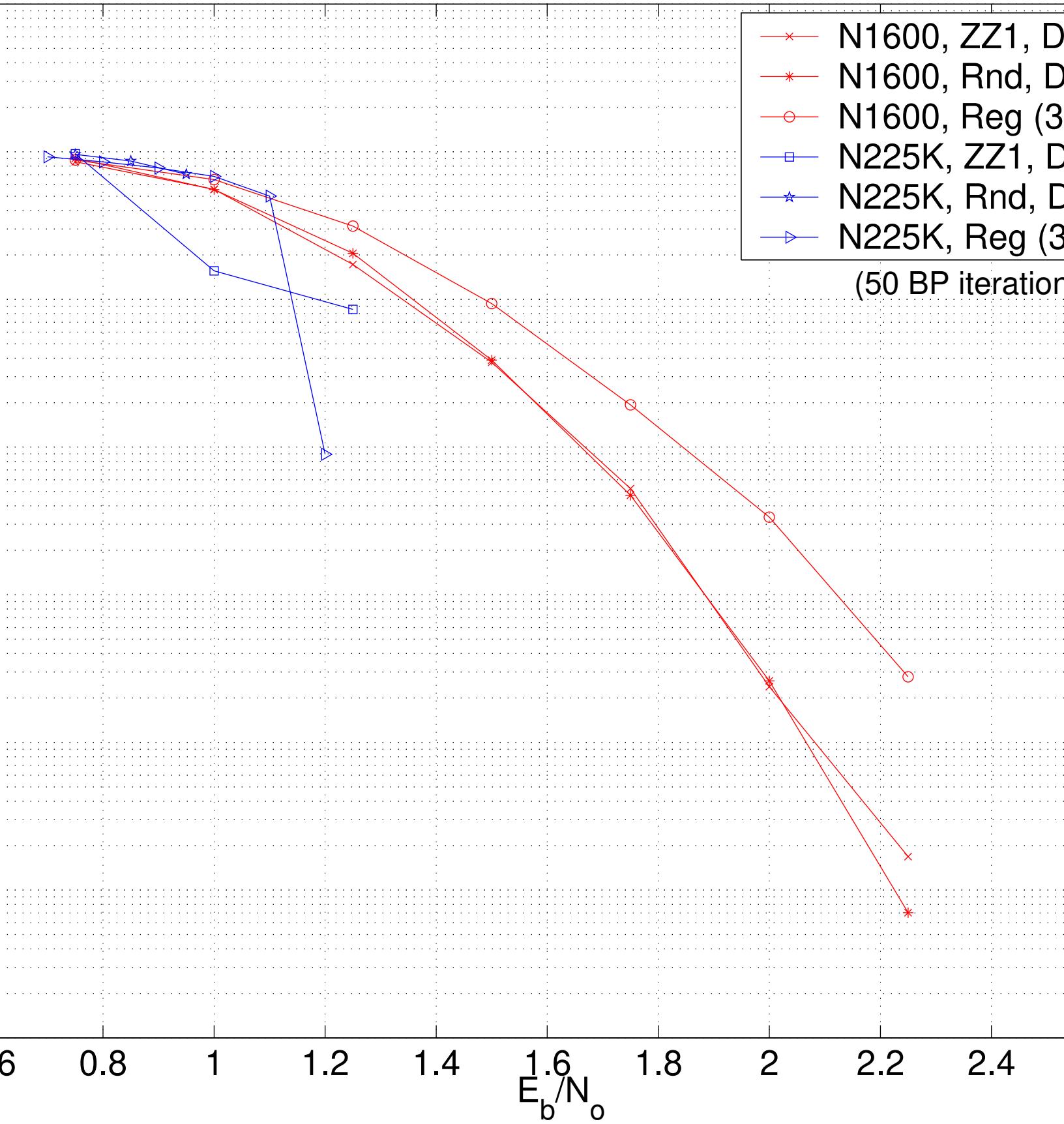

N1600,ZZ1, Deg 2
N1600, Rnd, Deg 2
N1600, Reg (3,6)
N1280,ZZ1, Deg 1
N1280, Rnd, Deg 1
N1280, Reg (3,6)
N7680,ZZ2, Deg 2
N7680, Rnd, Deg 2
N7680, Reg (3,6)
N=17K (Laf−Roc.)
N=17K (Reg (3,6))
0.6
0.8
1
1.2
1.4
1.6
1.8
2
2.2
2.4
2.6
E_b/N_o

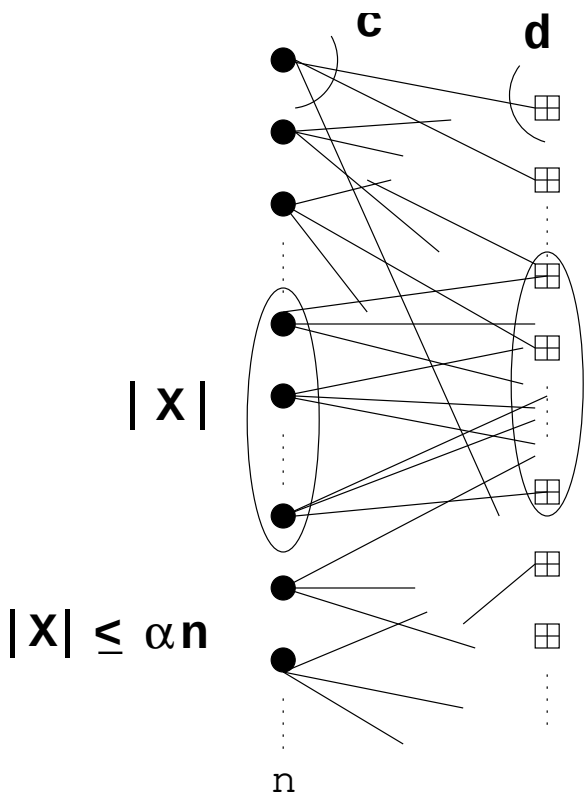

a_3
a_4
a_5


2
3


a_4
2
3

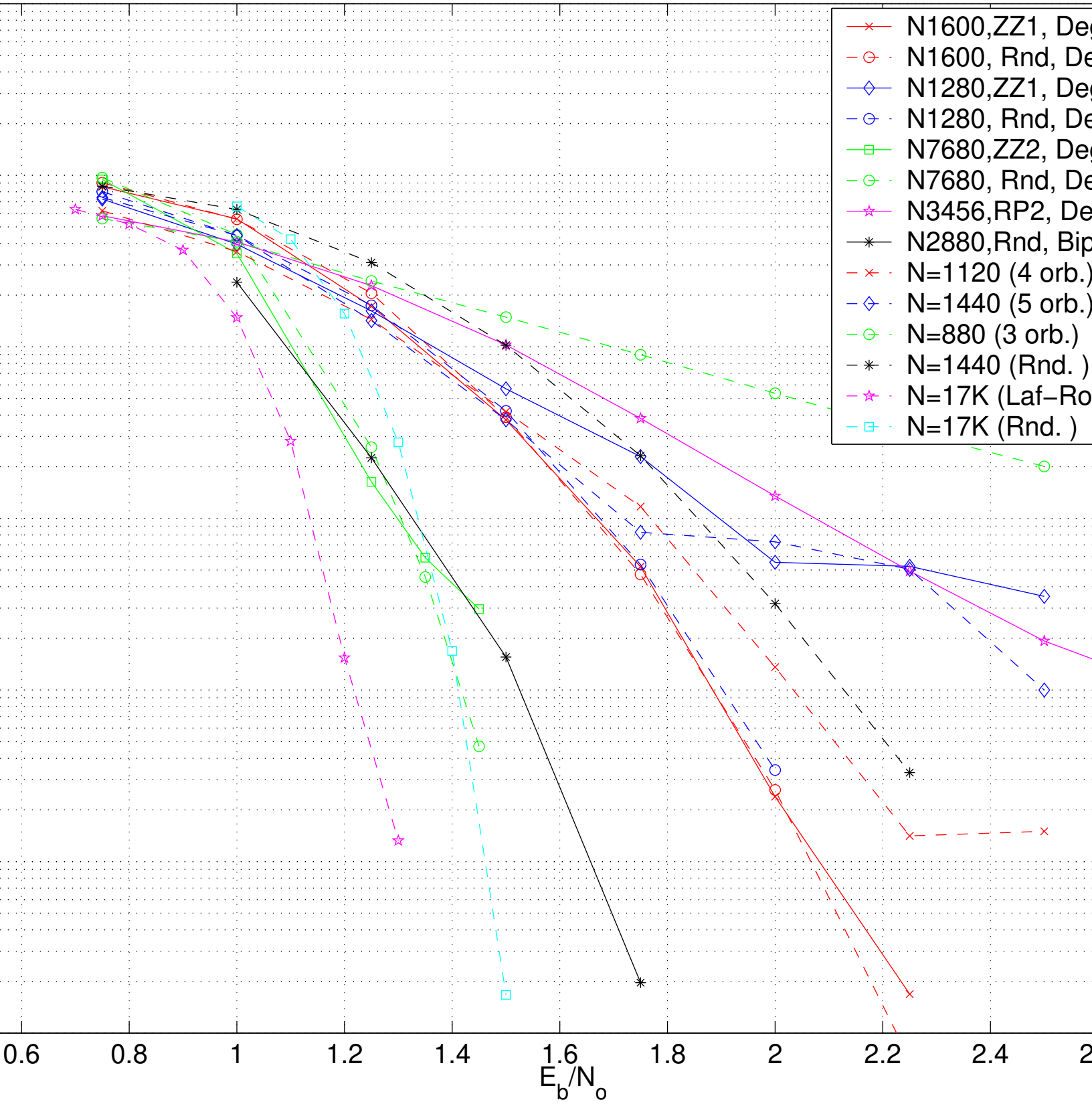
$G_1 \otimes G_2$

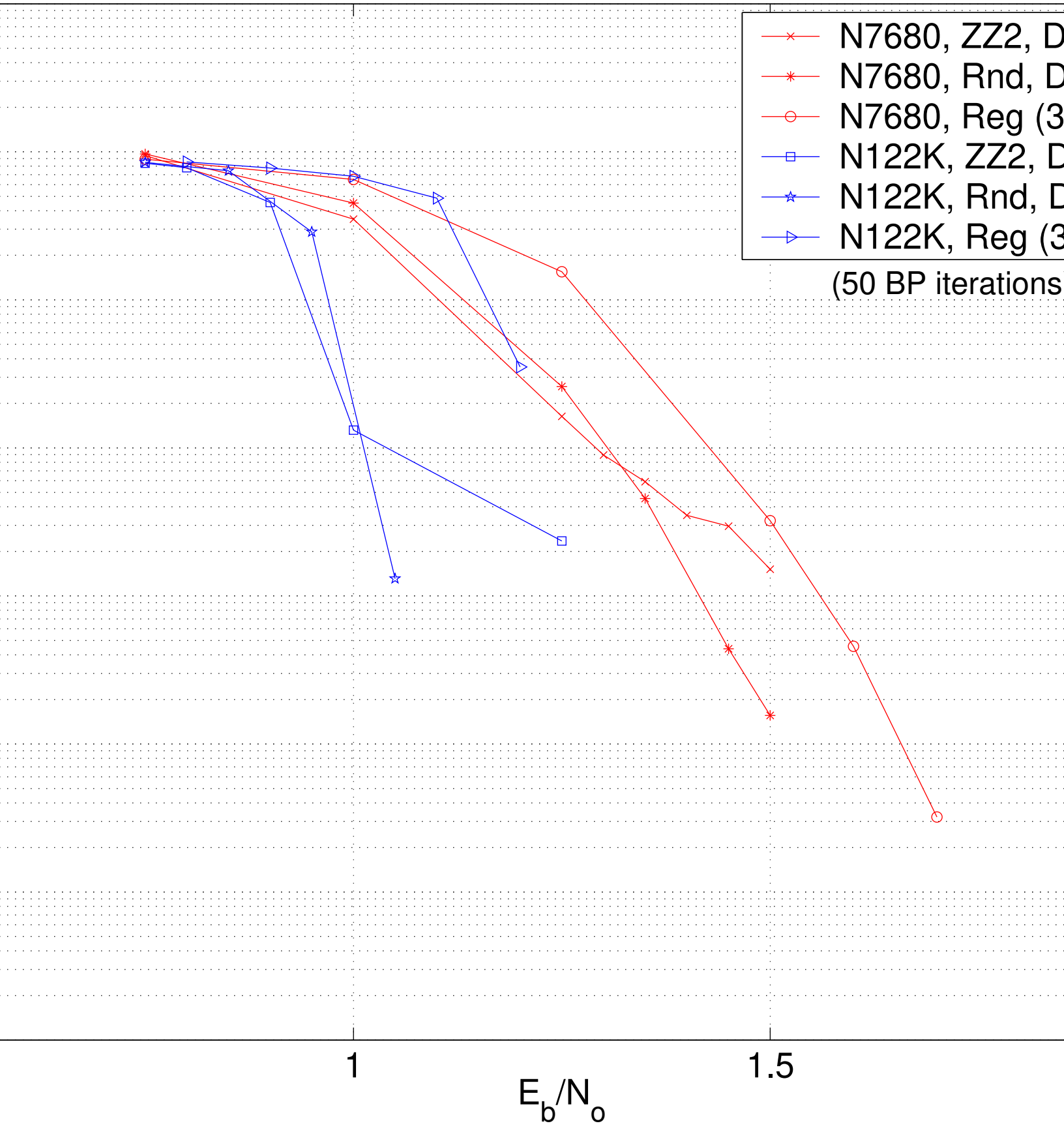

N1600, ZZ1, D
N1600, Rnd, D
N1600, Reg (3
N225K, ZZ1, D
N225K, Rnd, D
N225K, Reg (3
(50 BP iteration
6
0.8
1
1.2
1.4
1.6
1.8
2
2.2
2.4
E_b/N_o

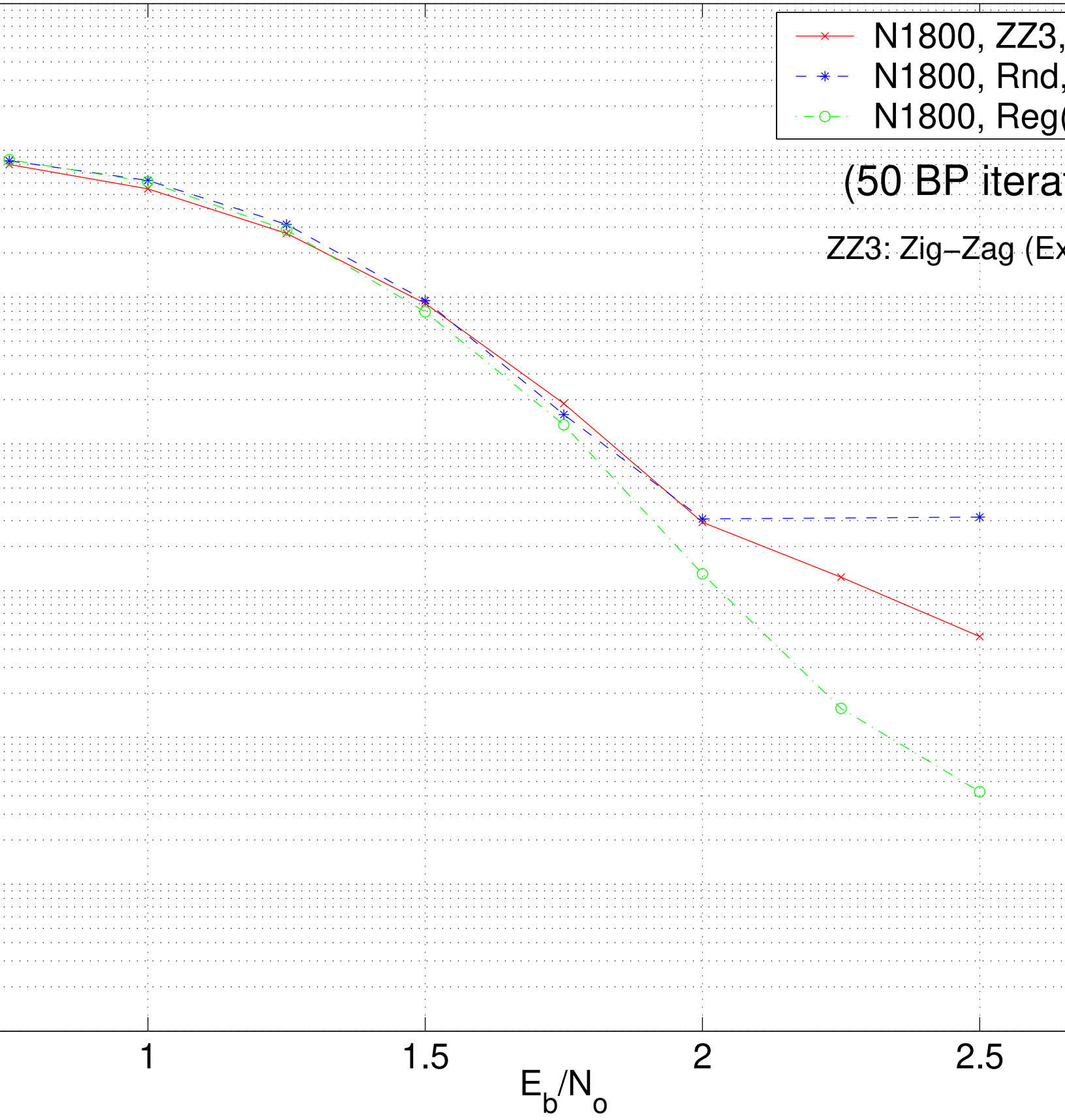


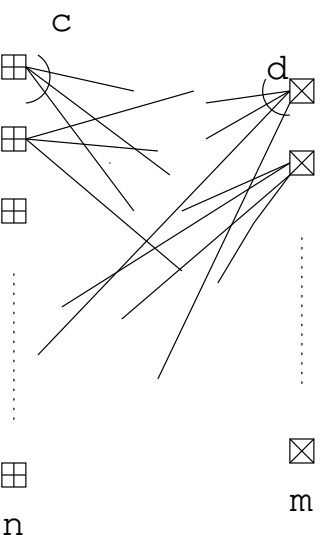
⊞ generalized constraint
([d, rd, εd] constraint)

$|\Gamma(X)| \geq \delta c |X|$


N1600,ZZ1, De
N1600, Rnd, De
N1280,ZZ1, De
N1280, Rnd, De
N7680,ZZ2, De
N7680, Rnd, De
N3456,RP2, De
N2880,Rnd, Bip
N=1120 (4 orb.)
N=1440 (5 orb.)
N=880 (3 orb.)
N=1440 (Rnd.)
N=17K (Laf−Ro
N=17K (Rnd.)
0.6
0.8
1
1.2
1.4
1.6
1.8
2
2.2
2.4
E_b/N_o


N7680, ZZ2, D
N7680, Rnd, D
N7680, Reg (3
N122K, ZZ2, D
N122K, Rnd, D
N122K, Reg (3
(50 BP iterations
1
1.5
E_b/N_o

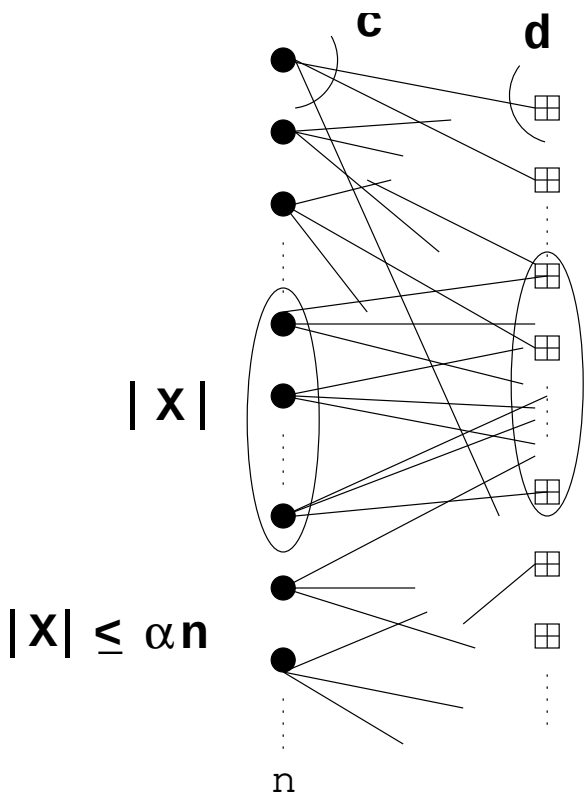

N1800, ZZ3,
N1800, Rnd,
N1800, Reg(
(50 BP itera
ZZ3: Zig–Zag (Ex
1
1.5
2
2.5
E_b/N_o



⊞ [c, r_1c, ε_1c] constraint

⊠ [d, r_2d, ε_2d] constraint

N = nc = md



⊞ parity-check constraint
([d,d-1, 2] constraint)

$|\Gamma(X)| \geq \delta c |X|$