Some new Algebraic constructions of Codes from Graphs which are good Expanders*

Christine Kelley, Joachim Rosenthal, and Deepak Sridhara
Department of Mathematics,
University of Notre Dame,
Notre Dame, IN 46556.
email: {ckelley1, rosen, dsridhar}@nd.edu

Abstract

The design of LDPC codes based on a class of expander graphs is investigated. Graph products, such as the zig-zag product [9], of smaller expander graphs have been shown to yield larger expanders. LDPC codes are designed based on the zig-zag product graph of two component Cayley graphs. The results for specific cases simulated reveal that the resulting LDPC codes compare well with other random LDPC codes at short block lengths – suggesting that product graphs form yet another avenue to pursue in the design of codes over graphs.

1 Introduction

Recent years have seen a widespread activity in the area of codes over graphs. LDPC codes have been used to demonstrate near capacity performance over many different kinds of channels [10]. As these are codes primarily defined over graphs, the design of *good* graphs becomes a key criterion for the design of *good* codes. It has been shown in [12] that graphs having good expansion properties are good candidates for LDPC code designs. Among recent developments, Reingold et al [9] show that by using small component graphs that are known to be expanders, it is possible to design larger graphs that are also expanders. Their technique is the so called *zig-zag* product of the two component graphs. In this paper, we examine the expansion properties of the zig-zag product in relation to the design of LDPC codes. We also examine other graph products and design LDPC codes using the product graphs. 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 [13]. By choosing component graphs with relatively small degree, we obtain product graphs that are relatively sparse. Our preliminary findings in this direction indicate that LDPC codes based on zig-zag product graphs perform comparably to random LDPC codes for short block lengths, and to obtain a good LDPC code, the vertices of the zig-zag product graph must be fortified with strong (i.e., good minimum distance) sub-code constraints. This is necessary also to achieve good performance with graph based message passing decoders.

The paper is organized as follows. Section 2 discusses preliminaries, such as the notion of expansion and the eigenvalue bound for graphs. In Section 3, different types of graph products are described. The properties of the product graphs, such as expansion, diameter, and girth are examined in Section 4. Here, we also discuss their use as potential candidates for designing codes over graphs. Section 5 describes some preliminary LDPC code constructions based on the product graphs. An example illustrating the zig-zag product of two Cayley component graphs is presented. Section 6 compares the performance of LDPC codes designed over the zig-zag product graphs with that of randomly constructed LDPC codes under belief propagation decoding. Section 7 summarizes the results and concludes the paper.

*This research is supported in part by NSF Grants DMS-00-72383 and CCR-02-05310 and the Notre Chair in Applied Mathematics. The first author was also supported by a fellowship from the Center of Applied Mathematics at the University of Notre Dame.

2 Preliminaries

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 [12], the second largest eigenvalue of the adjacency matrix A must be as small as possible compared to the index [14]. 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 $\tilde{A}$ has an index of 1. In this paper, we will follow the definition provided in [2, 8] for a graph to be an *expander*. A sequence of graphs is said to be an expander family if for every 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. The best possible expansion based on the eigenvalue bound is achieved by Ramanujan graphs that have $\lambda(G) \leq \frac{2\sqrt{d-1}}{d}$ [7]. 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 $1 - \lambda(G)$.

3 Graph Products

This section describes different ways of forming graph products. In each case, the expansion of the product graph with respect to the expansion of the component graphs is examined.

3.1 Basic Graph Products

Standard graph products include the Cartesian product, tensor product, strong product, and lexicographic product [3]. Let G_1 and G_2 be the component graphs. Then each of these products yields a graph with $V(G_1) \times V(G_2)$ as the vertices, and edge relations based on edges in the components. However, since these products yield graphs which are not sparse, we omit further detail based on their lack of potential for LDPC codes (see section 4.4).

3.2 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.) 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 $G = G_1 \textcircled{R} G_2$ 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 $f(\lambda_1, \lambda_2) = \lambda_1 + \lambda_2 + \lambda_2^2$ [9, Theorem 6.4]. Note that the degree of the replacement product graph depends only on the degree of the smaller component graph G_2.

3.2.1 Connections with semi-direct product of groups

We now consider the case when the two component graphs are Cayley graphs [11]. 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 replacement 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. If S_a is the union of k orbits, i.e., the orbit 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 $|G_a||G_b|$ [6].

3.3 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. Then the zig-zag product of G_1 and G_2, as introduced in [6, 9], is a graph G defined as follows:

- 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 $(v[k[i]], k[i])$, where $v[k[i]]$ is the $k[i]^{th}$ neighbor of v in G_1 and correspondingly, v is the $k[i]^{th}$ neighbor of $v[k[i]]$ in G_1.
  - a final step "*zag*" on the small graph G_2 is made from vertex $(v[k[i]], k[i])$ to vertex $(v[k[i]], k[i][j])$, where $k[i][j]$ is the j^{th} neighbor of $k[i]$ 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 $i, j \in \{1, \ldots, d_2\}$.

It is shown in [9] that the zig-zag product graph $G = G_1 \textcircled{z} G_2$ 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.

As earlier, if we use Cayley graphs as the components for the product graph, 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 [9]. 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 $k|S_b|^2$.

3.3.1 Case for unbalanced bipartite graphs

Extending the above 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. Then the zig-zag product graph, which we also will denote by $G = G_1 \textcircled{z} G_2$, 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:

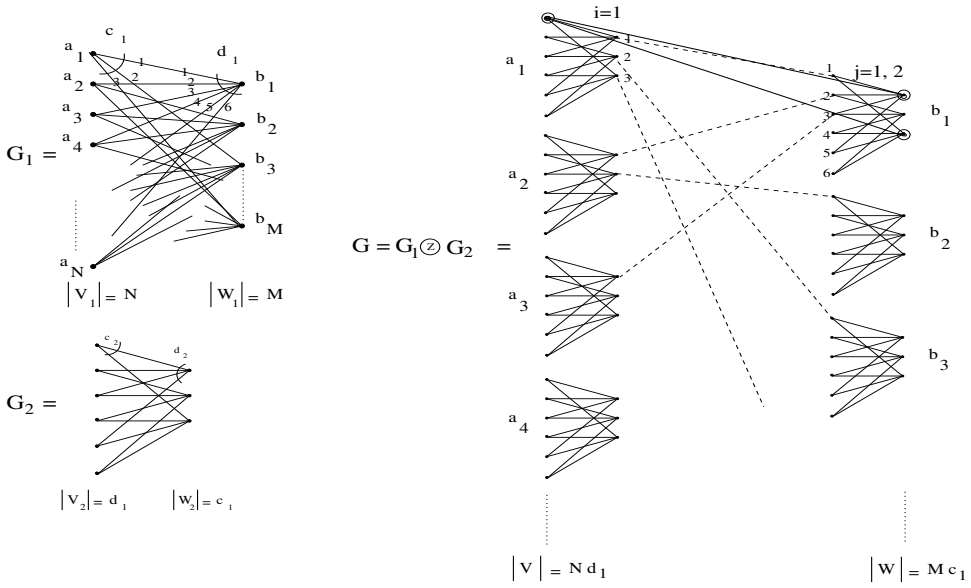

Figure 1: Zig-Zag product of two unbalanced bipartite graphs.

- 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 1.) 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 $(v, k) \to (v, k[i])$, for $i \in \{1, \ldots, c_2\}$.
  - A deterministic step from left to right on G_1 $(v, k[i]) \to (v[k[i]], \ell)$, 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.
  - A small step "zag" from left to right in the local copy of G_2. This is a step $(v[k[i]], \ell) \to (v[k[i]], \ell[j])$, where the final vertex is in W, for $j \in \{1, \ldots, c_2\}$.

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

We will show that $\lambda(G_1 \textcircled{z} G_2) \leq \lambda_1 + \lambda_2 + \lambda_2^2$. The expansion of the unbalanced zig-zag product graph is similar to that of the original zig-zag product graph [9]. (The case of balanced bipartite graphs has been dealt in [6].) Note that unlike in the original zig-zag product construction [9], 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.

4 Properties of the graphs

4.1 Degree

A small degree is desirable for designing LDPC codes over graphs. We have seen that the replacement product has a vertex degree that is of the same order as that of the smaller component graph, while the zig-zag product graph has a degree that is the square of the vertex degree of the smaller component graph. These parameters help determine our choice of graphs in the design of LDPC codes.

4.2 Expansion

The expansion coefficients in the previous section show that the replacement product graphs and zig-zag product graphs will be expanders if both the component graphs are. The results in [9] show that the replacement product graph isn't as good an expander as the zig-zag product graph. We state the following result (without proof) that shows the zig-zag product of two unbalanced bipartite component graphs to be an expander if the component graphs are.

Lemma 4.1 *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 $G_1 Ⓩ G_2$ is a (c_2^2, d_2^2)-regular bipartite on $(N \cdot d_1, M \cdot c_1)$ vertices with $\lambda = \lambda(G_1 Ⓩ G_2) \leq \lambda_1 + \lambda_2 + \lambda_2^2$.*

4.3 Diameter and Girth

We now look at the diameter and girth of the graph products considered in Section 3. Let us assume that the component graphs G_1 and G_2 have girths g_1 and g_2, respectively and diameters t_1 and t_2, respectively.

Lemma 4.2 *The girth and diameter of the replacement product graph $G = G_1 Ⓡ G_2$ are given by: (a) girth $\min\{g_2, g_1\} \leq g \leq \min\{g_2, g_1 + t_2\}$, and (b) diameter $t \leq t_1 + t_2$.*

Lemma 4.3 *The girth and diameter of the zig-zag product graph $G = G_1 Ⓩ G_2$ are given by: (a) girth $g = 4$, and (b) diameter $t \leq t_1 + 2t_2$.*

4.4 Remarks

For designing codes over graphs, a graph with good expansion, relatively small degree, small diameter, and large girth, is desired. Despite their expansion, the standard graph products mentioned earlier are not good candidates for LDPC code construction. The resulting graphs have relatively large degree and therefore are not sparse – a drawback for graph based message passing decoding. In addition, each has girth $g = 4$. The replacement product graph, while having a relatively small degree and promising girth, has relatively inferior expansion. The zig-zag product graph falls somewhere in between – its degree isn't too large and its expansion isn't too bad. However, the girth of the zig-zag product graph is always four. This means that the codes based on the zig-zag product fare poorly when the size of the graph increases. (For any randomly designed graph, the girth grows almost logarithmically with the size of the graph.)

5 LDPC codes: Construction and Properties

In this section, we use the zig-zag 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 [13] and [5].

We further restrict the choice of the component graphs for our zig-zag product to be appropriate Cayley graphs so that we can work directly with the group structure of the Cayley graphs. The following examples illustrate the code construction technique:

EXAMPLE 1: [2] 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, $C(B, \{\pm 1\})$. 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[1] $d_g \leq k|S_B|^2 = 4k$. 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 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.)

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.

Another entirely different approach for the design of asymptotically good codes is suggested in [2]. If the elements $a_1, \ldots, a_k$ are appropriately chosen and the group B acts on each of them producing an orbit of elements for each a_i, then these vectors could be arranged as rows of a circulant matrix (For the example above, the set of elements obtained when B acts on a_i is all cyclic shifts of a_i). Hence, a block of k circulant matrices $[C_1, C_2, \ldots, C_k]$ is obtained from the action of B on $a_1, a_2, \ldots, a_k$. The authors in [2] establish that the result of multiplying any non-zero vector $\mathbf{x} \in \mathbb{F}_2^p$ with $[C_1, \ldots, C_k]$ is a vector $\mathbf{c}$ that contains at least δ fraction of ones and δ fraction of zeros for some $\delta > 0$. This is a probabilistic result that holds with high probability when the choice of a_i's is completely random. Suppose we consider the block of circulants as the generator matrix G_{en} of a binary linear block code, then the result in [2] says for any non-zero message vector $\mathbf{x}$, $\mathbf{c} = \mathbf{x}G_{en}$ has a relative Hamming weight of at least δ, meaning the relative minimum distance of the code described by G_{en} is at least $\delta > 0$. Therefore, by selecting a large prime p, we can get a long block length binary linear code of rate $1/k$ with minimum distance linear in the block length of the code. However, it is not clear at this point whether it is possible to obtain a sparse parity check matrix representation for such a code. A sparse parity check representation would then guarantee efficient graph based iterative decoding with the benefit of good minimum distance imposed by the construction.

EXAMPLE 2: [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

[1]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 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 with high probability the Cayley graph $C(A, \{a_1^B, \ldots, a_k^B\})$ is an expander. Further, the zig-zag product of the two Cayley graphs is the Cayley graph $C(A \rtimes B, S)$ (as in Example 1) on $|A||B| = 2^{p+1}(p^3 - p)$ vertices. However, this Cayley graph will be a directed Cayley graph since the generating set S is not symmetric. Hence, we modify our graph construction a little 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 3: Codes from zig-zag product of unbalanced bipartite 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 $N_{LD} = N d_1 c_2^2$ and the rate is $r \geq \frac{N d_1 c_2^2 - (N d_1 (c_2^2 - k_1) + M c_1 (d_2^2 - k_2))}{N d_1 c_2^2} = \frac{k_1}{c_2^2} + \frac{k_2}{d_2^2} - 1$ (since $N d_1 c_2^2 = M c_1 d_2^2$ 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.

In Section 6, specific cases of these three examples are designed and the performance of the resulting LDPC codes are compared with that of other randomly constructed LDPC codes.

5.1 Minimum distance Bounds

Sipser and Spielman in [12] and Janwa and Lal [4] lower bound the minimum distance of codes based on expander graphs. Let us assume that the vertices of a d-regular graph are interpreted as sub-code constraints of a $[d, k, \epsilon d]$ linear block code and the edges are interpreted as the code bits of the LDPC code as above. If the graph is an expander with eigenvalue $\lambda(G) = \lambda$ and if $\epsilon > \lambda$, then the relative minimum distance[2] is at least $\epsilon \frac{(\epsilon - \lambda)}{(1 - \lambda)}$. This is a decreasing function in λ for $0 \leq \lambda \leq \epsilon$. Therefore, for a LDPC code based on the $(N \cdot d_1, d_2^2, \lambda < f(\lambda_1, \lambda_2))$- zig-zag product graph, if the sub-code constraint used is of a $[d_2^2, k, \epsilon d_2^2]$ linear block code, then the relative minimum distance of the LDPC code is lower bounded by

$$d_{min} \geq \epsilon \frac{(\epsilon - f(\lambda_1, \lambda_2))}{(1 - f(\lambda_1, \lambda_2))}$$

Janwa and Lal extend this argument to the unbalanced bipartite case. Let the two sets of vertices of a (c, d)-regular bipartite graph on the vertex sets V_1 ($|V_1| = N$)and W_1 ($|W_1| = M$) be interpreted as sub-code constraints of a $[c, k_1, \epsilon_1 c]$ and a $[d, k_2, \epsilon_2 d]$ linear block codes, respectively, and the edges be interpreted as code bits of a LDPC code. If the bipartite graph G has $\lambda(G) = \lambda$ and if $d\epsilon_2 \geq c\epsilon_1 > \lambda\sqrt{cd}/2$, then the relative minimum distance of the LDPC code is at least

$$d_{min} \geq \{\epsilon_1 \epsilon_2 - \frac{\lambda}{2}(\epsilon_1 \sqrt{\frac{c}{d}} + \epsilon_2 \sqrt{\frac{d}{c}})\}$$

[2] i.e., relative to the block length of the LDPC code.

Using Lemma 4.1 and the above result, the relative minimum distance of LDPC codes based on Example 3 can be lower bounded by $d_{min} \geq \{\epsilon_1\epsilon_2 - \frac{\lambda(G_1 \textcircled{Z} G_2)}{2}(\epsilon_1 \frac{c_2}{d_2} + \epsilon_2 \frac{d_2}{c_2})\}$. (Here, we use sub-codes $[c_2^2, k_1, \epsilon_1 c_2^2]$, $[d_2^2, k_2, \epsilon_2 d_2^2]$ and assume that $d_2^2\epsilon_2 \geq c_2^2\epsilon_1 > \frac{\lambda(G_1 \textcircled{Z} G_2)}{2}$.)

6 Results

The performance of the LDPC code designs based on zig-zag 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 belief propagation (BP) algorithm. Since the LDPC codes based on the zig-zag product graph 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 [15].) 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 2 shows the performance with BP decoding of a LDPC code that is designed based on Example 1 in Section 5. 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 in this 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 225280 LDPC code. At this block length however, the LDPC based on the zig-zag product graph is found to perform better than the LDPC code based on a random degree 20 graph primarily at small SNRs, but due to the poor girth of the zig-zag product graph[3] (and hence, in the LDPC constraint graph), its performance at high SNRs is inferior to that of the random LDPC design.

Figure 3 shows the performance of LDPC codes that are designed based on Example 2. 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 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, a longer block length 122880 LDPC code is obtained. As in the previous case, this code performs poorly in contrast to its random counterpart. Once again, we attribute this behavior to the poor girth of the zig-zag product graph.

Figure 4 shows the performance of LDPC codes designed based on the zig-zag product of two unbalanced bipartite graphs as in Example 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

[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.

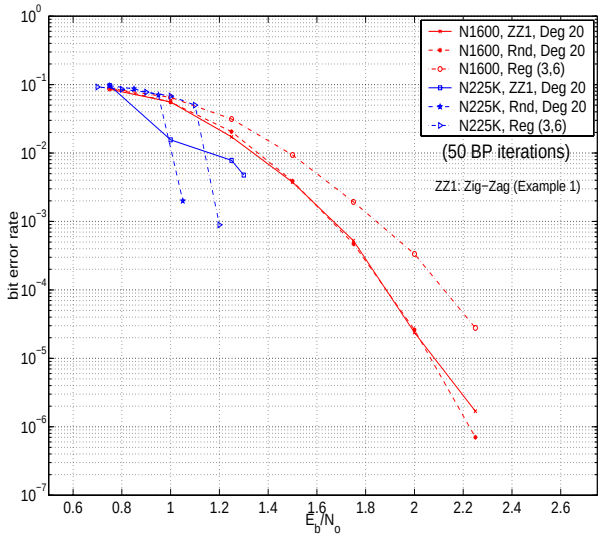

Figure 2: LDPC codes from zig-zag product graphs based on Example 1.

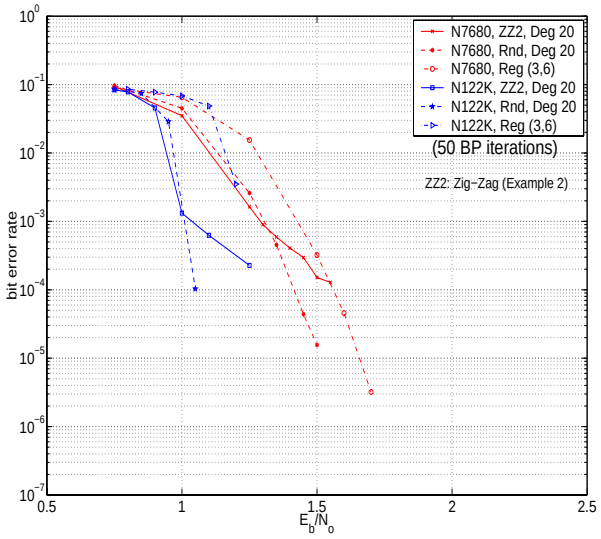

Figure 3: LDPC codes from zig-zag product graphs based on Example 2.

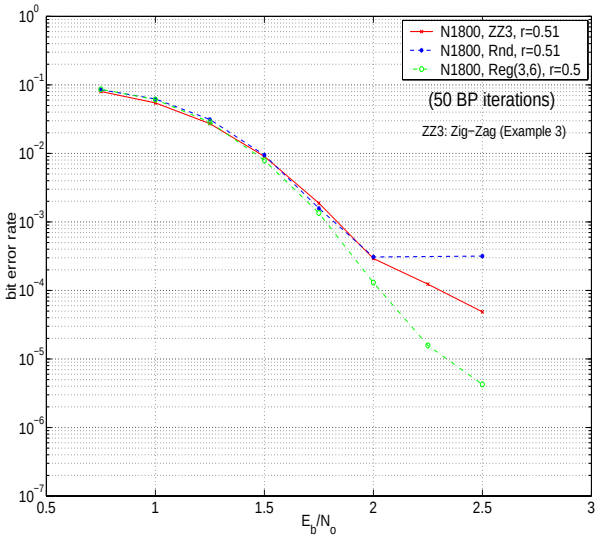

Figure 4: Performance of LDPC codes from zig-zag product graphs based on Example 3.

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 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.

7 Conclusions

In this paper we discussed the properties of the replacement product and the zig-zag product, and investigated their potential for LDPC code constructions. Included in this was a straightforward generalization of the zig-zag product for the case of unbalanced

bipartite graphs. Our preliminary results indicate that LDPC codes based on the zig-zag product of two good component graphs yields decent performance at short block lengths, but the overall performance is severely limited by the poor girth and cycle distribution of the zig-zag product graph. A modification in the zig-zag product steps may be necessary to alleviate this problem. The code design may be improved by a more judicious choice of component graphs. These construction techniques may also be applied to the replacement product graphs. Given that the replacement product graph has better girth than the zig-zag product, we suspect that LDPC codes based on the replacement product would perform better.

References

[1] N. Alon, *Eigenvalues and expanders*, Combinatorica, 6 (1986), pp. 83–96. 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), IEEE Computer Soc., Los Alamitos, CA, 2001, pp. 630–637.

[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 (2003), pp. 335–347.

[5] 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, Nov. 2000.

[6] 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/.

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

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

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

[10] T. Richardson and R. Urbanke, *The capacity of low-density parity check codes under message-passing decoding*, IEEE Trans. Inform. Theory, 47 (2001), pp. 599–618.

[11] 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, 2000, pp. 248–257.

[12] M. Sipser and D. A. Spielman, *Expander codes*, IEEE Trans. Inform. Theory, 42 (1996), pp. 1710–1722.

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

[14] ——, *Explicit concentrators from generalized N-gons*, SIAM J. Algebraic Discrete Methods, 5 (1984), pp. 287–293.

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