Advances in Mathematics of Communications
Volume 2, No. 4, 2008, 347–372
doi:10.3934/amc.2008.2.347

ZIG-ZAG AND REPLACEMENT PRODUCT GRAPHS AND LDPC CODES

Christine A. Kelley
Department of Mathematics
University of Nebraska-Lincoln
Lincoln, NE 68588, USA

Deepak Sridhara
1251 Waterfront Place
Seagate Technology
Pittsburgh, PA 15222, USA

Joachim Rosenthal
Institut für Mathematik
Universität Zürich
Zürich, CH-8057, Switzerland

(Communicated by Mike O'Sullivan)

Abstract. It is known that the expansion property of a graph influences the performance of the corresponding code when decoded using iterative algorithms. Certain graph products may be used to obtain larger expander graphs from smaller ones. In particular, the zig-zag product and replacement product may be used to construct infinite families of constant degree expander graphs. This paper investigates the use of zig-zag and replacement product graphs for the construction of codes on graphs. A modification of the zig-zag product is also introduced, which can operate on two unbalanced biregular bipartite graphs, and a proof of the expansion property of this modified zig-zag product is presented.

1. 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 [3, 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 noncommutative *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 square of 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

2000 *Mathematics Subject Classification:* Primary: 58F15, 58F17; Secondary: 53C35.
Key words and phrases: Codes on graphs, LDPC codes, expander graphs, zig-zag product, replacement product of a graph.
This work was supported in part by the NSF Grant No. CCR-ITR-02-05310 and the Swiss NSF Grant No. 113251, and conducted in part when the first author was at The Fields Institute, Toronto, and the Ohio State University, and when the second and third authors were at the University of Zürich.

is a good expander as well. Similar facts apply to the replacement product that results in a slightly smaller expansion in comparison to the zig-zag product.

Since the work of Sipser and Spielman [15] it has become well known that the performance of codes defined on graphs is heavily influenced by the expansion property of the graphs. Several authors have provided constructions of graph-based codes 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 to decode low density parity check (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 [5, 7, 8, 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 [8, 14, 18].

Ramanujan graphs have the drawback that they exist only for a limited set of parameters. In contrast, the zig-zag product and the replacement product can be performed on a large variety of component graphs. The iterative construction also has 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 [6].

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. 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]. Codes obtained in this way will be referred to as *generalized LDPC* (GLDPC) *codes*. 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) subcode constraints, in order to achieve good performance with graph-based 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. The girth and diameter of these products is discussed.

Section 4 contains a new zig-zag product construction of unbalanced bipartite graphs. The main result is Theorem 1 which essentially states that the constructed bipartite graph is a good expander graph if the two component graphs are good expanders. The proof of this theorem is provided in the appendix.

Section 5 is concerned with applications to coding theory. The section contains several code constructions using the original and the unbalanced bipartite zig-zag products and the replacement product. Simulation results of the LDPC codes constructed in Section 5 are presented in Section 6. The simulation results of the zig-zag and replacement product LDPC codes are comparable to, if not better than, randomly constructed LDPC codes of similar parameters. Section 7 introduces an iteration scheme for both the replacement product and a 4-step version of the unbalanced zig-zag graph 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 properties of the iterative families are also discussed. Section 8 summarizes the results and concludes the paper.

2. Preliminaries

In this section, we review the basic graph theory notions used in this paper.

Let $G = (V, E)$ be a graph with vertex set V and edge set E. The number of edges involved in a path or cycle in G is called the *length* of the path or cycle. The *girth* of G is the length of the shortest cycle in G. If $x, y \in V$ are two vertices in G, then the *distance* from x to y is defined to be the length of the shortest path from x to y. If no such path exists from x to y, then we say the distance from x to y is infinity. The *diameter* of G is the maximum distance among all pairs of vertices of G.

Intuitively, a graph has good expansion if any small enough set of vertices in the graph has a large enough set of vertices connected to it. 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 (in absolute value) must be as small as possible compared to the largest eigenvalue [17]. For a d-regular graph G, the largest eigenvalue of A is d. Hence, by *normalizing* the entries of A by the factor d, the normalized matrix, $\frac{1}{d}A$, has the largest eigenvalue equal to 1.

Definition 1. Let G be a d-regular graph on N vertices. Denote by $\lambda(G)$ the second largest eigenvalue (in absolute value) of the normalized adjacency matrix representing G. G is said to be a (N, d, λ)-*graph* if $\lambda(G) \leq \lambda$.

In this paper, we will follow the definition provided in [2, 12] for a graph to be an *expander*.

Definition 2. A sequence of graphs is said to be an *expander family* if for every (connected) graph G in the family, the second largest eigenvalue $\lambda(G)$ is bounded above by some constant $\kappa < 1$. In other words, there is an $\epsilon > 0$ such that for every graph G in the family, $\lambda(G) < 1 - \epsilon$. A graph belonging to an expander family is called an *expander graph*.

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]. 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]. Hence, 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. Using the definition of expansion for bipartite graphs in [17, 5], we have the following:

Definition 3. 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 4. 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 (in absolute value) of the normalized adjacency matrix of 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 the 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.

3. 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 degrees. Although sparsity is not as essential for generalized LDPC codes, large degrees significantly increase the complexity of the decoder.

In this section we describe the zig-zag product of [3, 13], introduce a variation of the zig-zag product that holds for bi-regular unbalanced bipartite graphs (that is, (c, d) regular 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 (that is, $c = d$), this bi-regular product yields the product in [3, 13]. In addition to preserving expansion, these products are notable in that the resulting product graphs have degrees dependent on only one of the component graphs, and therefore can be chosen to yield graphs suitable for coding.

3.1. Zig-zag product. Let G_1 be a $(N_1, d_1, \lambda^{(1)})$-graph and let G_2 be a $(d_1, d_2, \lambda^{(2)})$-graph. Randomly number the edges around each vertex of G_1 by $\{1, \ldots, d_1\}$, and randomly number the vertices of G_2 by $\{1, \ldots, d_1\}$.Then the zig-zag product

*See Definition 3.

$G = G_1 ⓩ G_2$ of G_1 and G_2, as introduced in [3, 13], is a $(N_1 \cdot d_1, d_2^2, \lambda)$-graph defined as follows[†]:

- 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.
- 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 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 $(v[k[i]], \ell[j])$ for $i, j \in \{1, \ldots, d_2\}$.

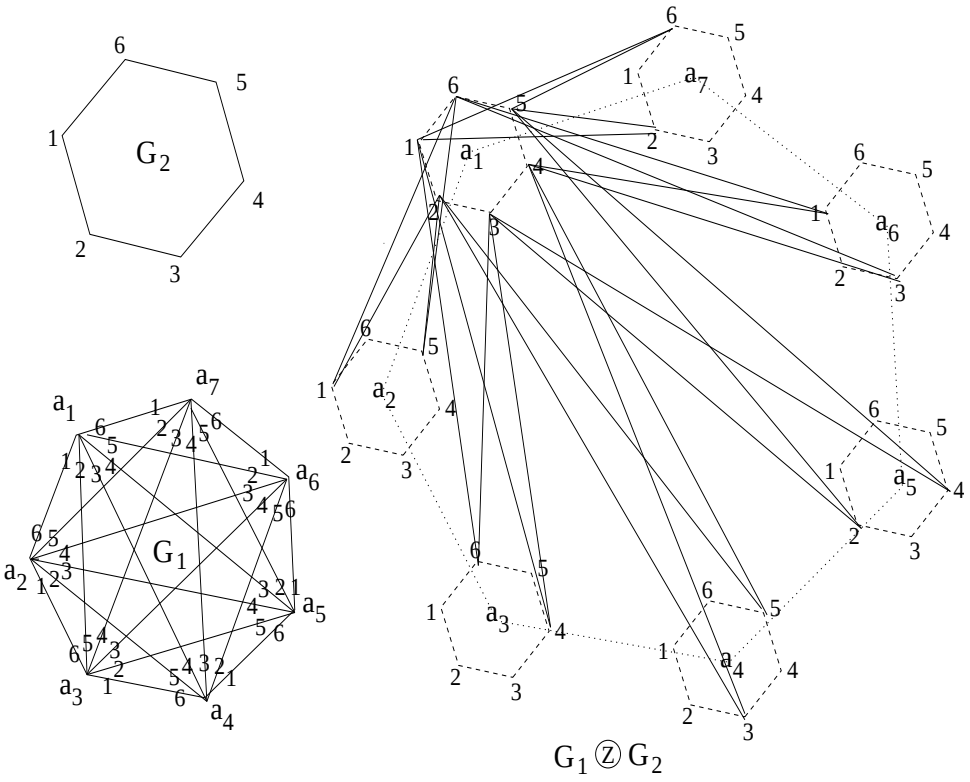

FIGURE 1. Zig-zag product of two graphs.

[†] 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.

Example 1. Consider the zig-zag product graph $G = G_1 ⓩ G_2$ depicted in Figure 1. The edge from $(1,3)$ ("cloud a_1, vertex 3") to $(5,2)$ ("cloud a_5, vertex 2") is obtained by the following 3 steps:

$$(1,3) \to (1,4) \to (5,3) \to (5,2).$$

The first step from 3 to 4 in cloud a_1 takes $(1,3)$ to $(1,4)$. The second step from a_1 to a_5 in G_1 takes $(1,4)$ to $(5,3)$, since a_5 is the 4th neighbor of a_1 and a_1 is the 3rd neighbor of a_5 in the labeling around the vertices of G_1. The final step in cloud a_5 takes $(5,3)$ to $(5,2)$. Similarly, vertex $(1,3)$ also connects to $(5,4)$,$(3,4)$, and $(3,6)$ by these steps:

$$(1,3) \to (1,4) \to (5,3) \to (5,4),$$
$$(1,3) \to (1,2) \to (3,5) \to (3,4),$$
$$(1,3) \to (1,2) \to (3,5) \to (3,6),$$

so the degree of vertex $(1,3)$ is $2^2 = 4$ as expected.

It is shown in [13] that the zig-zag product graph $G = G_1 ⓩ G_2$ 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 1. *Let G_1 and G_2 have girth g_1 and g_2, respectively. Then the zig-zag product graph $G = G_1 ⓩ G_2$ has girth $g = 4$.*

Proof. We show that any pair of vertices at distance two 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 two 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 second 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. □

We now consider the case when the two component graphs are Cayley graphs [2, 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.

Using the construction in [2], 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. For $g \in G_a$ and $h \in G_b$, let g^h denote the action of h on g. 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

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

The group having as elements the ordered pairs $\{(g,h)|g \in G_a, h \in G_b\}$, and group operation defined by

$$(g', h')(g, h) = (g' g^{h'^{-1}}, h'h)$$

is called the *semi-direct product* of G_a and G_b, and is denoted by $G_a \rtimes G_b$. 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 $k|S_b|^2$.

3.2. Replacement product. Let G_1 be a $(N_1, d_1, \lambda^{(1)})$-graph and let G_2 be a $(d_1, d_2, \lambda^{(2)})$-graph. Randomly number the edges around each vertex of G_1 by $\{1, \ldots, d_1\}$, and each vertex of G_2 by $\{1, \ldots, d_1\}$. Then the replacement product $G = G_1 Ⓡ G_2$ of G_1 and G_2 has 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. Note that the degree of the replacement product graph depends only on the degree of the smaller component graph G_2. The replacement product graph $G = G_1 Ⓡ G_2$ 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].

Example 2. Consider the replacement product graph $G = G_1 Ⓡ G_2$ shown in Figure 2. The vertex $(1, 6)$ ("cloud a_1, vertex 6") has degree $d_2 + 1 = 3$. The edges from $(1, 6)$ to $(1, 1)$ and $(1, 5)$ result since in G_2, vertex 6 connects to vertices 1 and 5. The edge from $(1, 6)$ to $(7, 1)$ result since in G_1, a_1 is the first neighbor of a_7 and a_7 is the sixth neighbor of a_1 in the labeling of G_1. Similarly, $(1, 5)$ connects to $(1, 6)$ and $(1, 4)$ due to the original connections in G_2, and $(1, 5)$ has an edge to $(6, 2)$ since a_1 is the second neighbor of a_6 and a_6 is the fifth neighbor of a_2.

Lemma 2. *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 $G = G_1 Ⓡ G_2$ satisfy: (a) $\min\{g_2, 2g_1\} \leq g \leq \min\{g_2, g_1 t_2\}$, and (b) $\max\{t_2, 2t_1\} \leq t \leq t_1 + t_2$.*

Proof. (a) Observe that there are 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

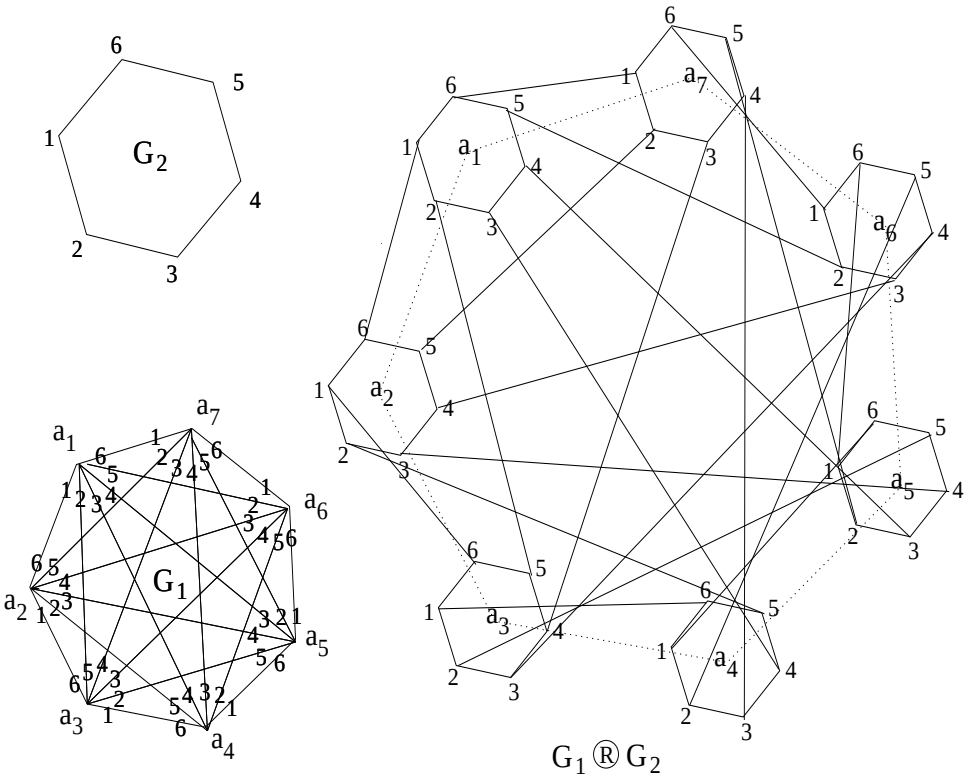

FIGURE 2. Replacement product of two graphs.

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 $S = (1_{G_a}, S_b) \bigcup \{(a_1, 1_{G_b}), \ldots, (a_k, 1_{G_b})\}$ as the generating set. The degree of this Cayley graph is $|S_b| + k$ and the size of its vertex set is $|G_a||G_b|$ [3]. (Here again, it is easily verified that when $k = 1$, the Cayley graph $C(G_a \rtimes G_b, S)$ is the replacement product originally defined in [3].)

4. Zig-zag product for unbalanced bipartite graphs

For the purpose of coding theory it would be very interesting to have a product construction of good unbalanced bipartite expanders. In this Section we adapt the original zig-zag construction in a natural manner. The main result (Theorem 1) will show that this construction results in a bipartite expander graph if the component bipartite graphs are expander graphs.

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 $\lambda^{(1)}$ and $\lambda^{(2)}$ denote the second largest eigenvalues in absolute value 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 $G = G_1 \textcircled{z} G_2$, 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:

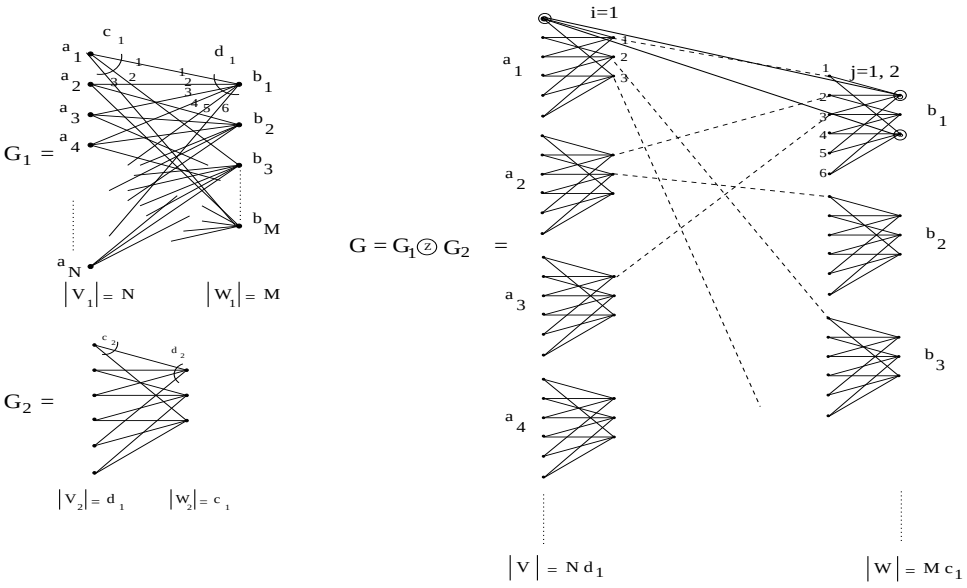

FIGURE 3. Zig-zag product of two unbalanced bipartite graphs.

- 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 3.) 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 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 $(v[k[i]], \ell[j])$.

There is a subtle difference in the zig-zag product construction described for the unbalanced bipartite component graphs when compared to the original construction in [13]. The difference lies in that the vertex set of G does not include vertices from

the set W_2 in every cloud of vertices from V_1 and similarly, the vertex set of G does not include vertices from V_2 in every cloud of vertices from W_1.

The following theorem describes the major properties of the constructed unbalanced bipartite graph.

Theorem 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 $G = G_1 (z)_p G_2$ is a (c_2^2, d_2^2)-regular bipartite on $(N \cdot d_1, M \cdot c_1)$ vertices with $\lambda = \lambda(G) \leq \lambda^{(1)} + \lambda^{(2)} + (\lambda^{(2)})^2$. Moreover, if $\lambda^{(1)} < 1$ and $\lambda^{(2)} < 1$, then $\lambda = \lambda(G) < 1$.*

The proof of the above theorem on the expansion of the unbalanced zig-zag product graph is deferred to the appendix. Several of the key ideas in the following proof are already present in the original zig-zag product graph paper [13]. Some modifications for balanced bipartite graphs has been dealt in [3]. 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 this can be seen using a similar argument as in Lemma 1.

5. Zig-zag and replacement product LDPC codes

In this section, we design LDPC codes based on expander graphs arising from the zig-zag and replacement products. 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 [8], and the resulting code is less affected by the small girth in the zig-zag product[‡]. 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 3. 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, $C(B, \{\pm 1\})$.

[‡]Note that the girth of the resulting Tanner graph describing the code is twice that of the underlying expander graph. Thus, girth four in the zig-zag product results in girth eight in the resulting Tanner graph describing the generalized LDPC code.

(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[§] $d_g \leq k|S_B|^2 = 4k$.

(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 $d_g = k + |S_B| = k + 2$.

For both (a) and (b), 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_{LDPC} of the LDPC code is $2^p \cdot p \cdot d_g/2$ and the rate of the LDPC code is

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

(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. 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 $1_A = \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix}$.)

(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.

In both (a) and (b), 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 an $s \in S$ such that $v * s = w$. The new product

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

graph obtained has $2|A||B|$ vertices and every vertex has degree $d_g = |S|$, the size of the generating set in each case; 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 5. *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 $[c_2^2, k_1, d_{m_1}]$ (resp., a $[d_2^2, k_2, d_{m_2}]$) 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 $N_{LDPC} = Nd_1c_2^2$ 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, respectively.

For all of the examples discussed above, the corresponding LDPC codes have a very succinct description and representation for implementation purposes. This is a considerable advantage over randomly constructed graphs and codes. Furthermore, as will be discussed in a later section, an entire family of zig-zag and replacement product graphs can be obtained by an iterative construction that uses a small graph as a seed graph. Thus, specifying this seed graph is sufficient to describe any graph in the resulting family. A certain level of expansion and thereby a certain level of minimum distance [15] is guaranteed for the zig-zag and replacement product graphs in contrast to the randomly designed codes.

6. 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 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 with sum-product decoding of the zig-zag product LDPC codes based on Example 3. 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 an LDPC code based on a randomly designed 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.

Figure 4 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 code based on the zig-zag product graph is found to perform comparably to, if not better than, the LDPC code based on a random degree 20 graph. The zig-zag product graph has a poor girth[¶] 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. However, by scaling the log-likelihood ratio (LLR) messages that are sent from the constraint nodes to the variable nodes during iterative decoding, the error-floor performance can be further improved.

Figure 5 shows the performance with sum-product decoding of a replacement product LDPC code based on Example 3. 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 an LDPC code based on a randomly designed 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.

Figure 6 shows the performance with sum-product decoding of zig-zag product LDPC codes based on Example 4. 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. 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, a bipartite 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 122,880 LDPC code is obtained. As in the previous case, this code also performs comparably to, 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.

[¶]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.

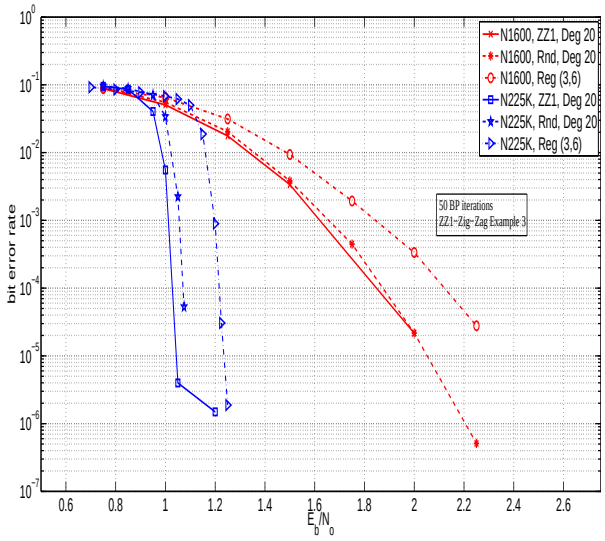

FIGURE 4. LDPC codes from zig-zag product graphs based on Example 3.

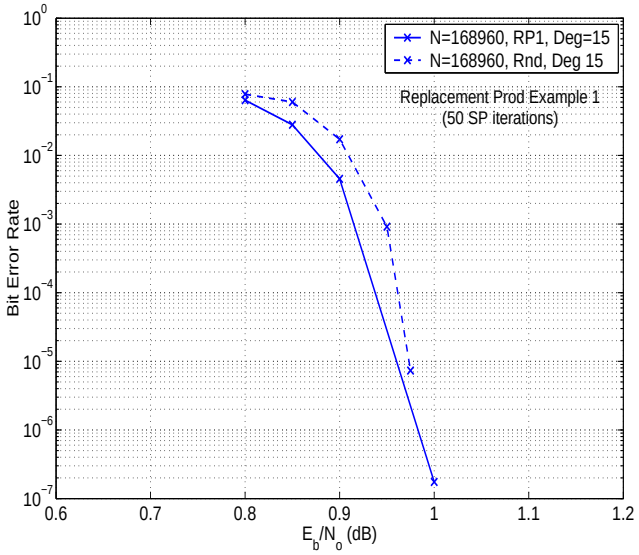

FIGURE 5. LDPC code from replacement product based on Example 3.

Figure 7 shows the performance with sum-product decoding of a replacement product LDPC code based on Example 4. 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, a graph on $15,360$ vertices with degree 14 is obtained. Using a $[14, 10, 3]$ code as a sub-code in the replacement product graph, a block length 107,520 rate 0.4285 LDPC code is obtained. The performance

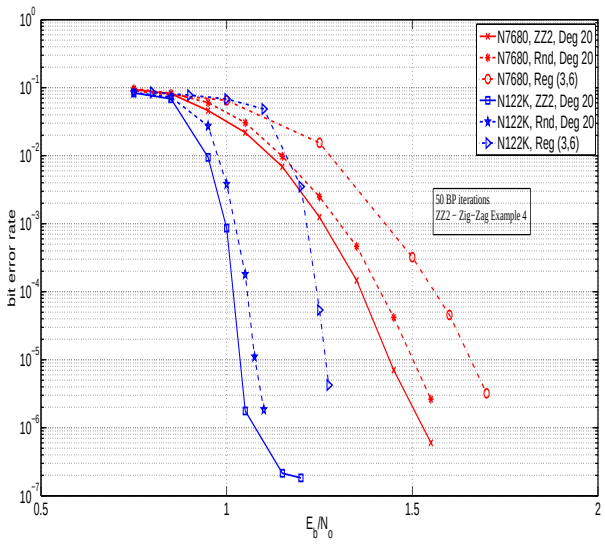

FIGURE 6. LDPC codes from zig-zag product graphs based on Example 4.

of the replacement product LDPC code is better than that of the random code in this example for low to moderate SNRs. However, the replacement product LDPC code has an error floor at high SNRs due to the short cycles in the graph.

Figure 8 shows the performance of LDPC codes designed based on the zig-zag product of two unbalanced bipartite graphs as in Example 5. 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 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 $(3, 6)$-regular LDPC code showing a small improvement over the 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. 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.

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 block lengths), 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. Still,

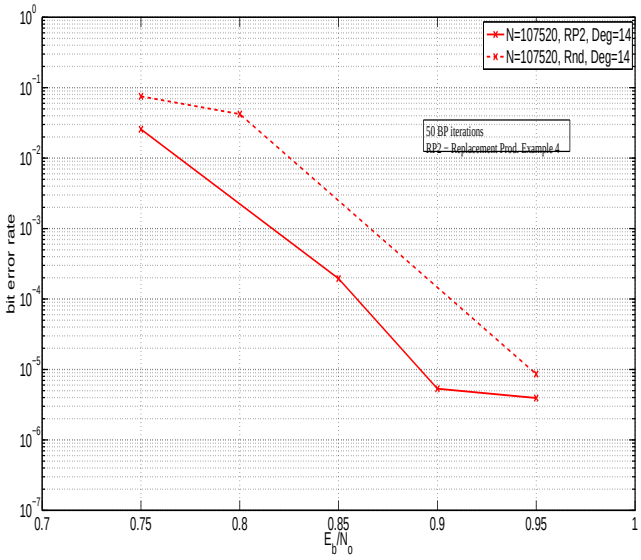

FIGURE 7. LDPC code from replacement product based on Example 4.

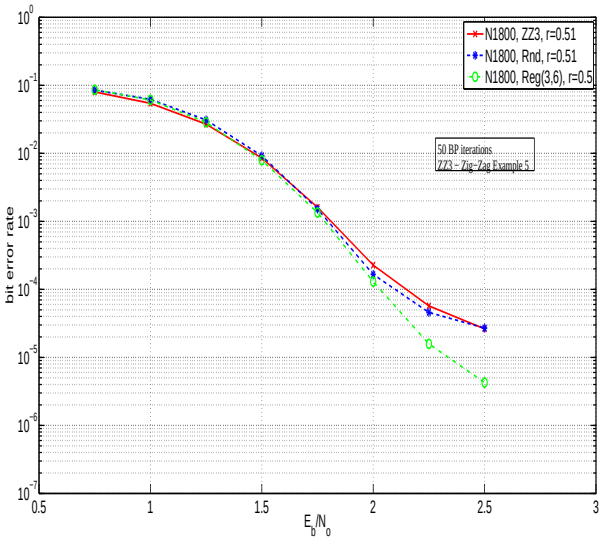

FIGURE 8. LDPC code from the unbalanced bipartite zig-zag product graph based on Example 4.3.

they may be of interest for other applications such as designing communication networks, complexity theory, or for derandomization algorithms. 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 4-step unbalanced bipartite zig-zag product and the replacement product thereafter.

7.1. ITERATIVE CONSTRUCTION OF ORIGINAL ZIG-ZAG PRODUCT GRAPHS [13]. 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

$$G_1 = H^2 = (D^4, D^2, \frac{1}{25}),$$
$$G_{i+1} = G_i^2 ⓩ H.$$

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

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

7.2. ITERATIVE CONSTRUCTION OF UNBALANCED BIPARTITE ZIG-ZAG PRODUCT GRAPHS. The unbalanced bipartite zig-zag product presented in Section 4 cannot be used directly to obtain an iterative construction, due to constraints on the parameters[‖]. Therefore, we slightly modify the zig-zag product by introducing 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 4-step unbalanced bipartite zig-zag product is presented next, followed by an iterative construction that uses this product.

7.2.1. *4-step 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 $\lambda^{(1)}$ and $\lambda^{(2)}$ denote the second largest eigenvalues (in absolute value) 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 in 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 $G = G_1 ⓩ G_2$, 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 3.) 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 $\ell \in \{1, \ldots, c_1\}$.

[‖] The only parameters that are compatible reduce to the special case where the bipartite components are balanced.

- A vertex $(v,k) \in V$ is connected to a vertex in W by making 4-steps in the product graph. The first three steps are the same as in Section 4. The fourth step is:
  - A second small step from right to left in the local copy of G_2. This is a step $(v[k[i]], \ell[j]) \to (v[k[i]], \ell[j][j'])$, 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 3. *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 4-step unbalanced zig-zag product graph $G_1 \textcircled{z} G_2$ 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 $\lambda = \lambda(G_1 \textcircled{z} G_2) \leq \lambda^{(1)} + \lambda^{(2)} + (\lambda^{(2)})^2$. Moreover, if $\lambda^{(1)} < 1$ and $\lambda^{(2)} < 1$, then $\lambda = \lambda(G_1 \textcircled{z} G_2) < 1$.*

The proof is omitted but may be seen intuitively given the expansion of the original unbalanced zig-zag product (Theorem 1) 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 $G = G_1 \textcircled{z} G_2$, 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.

7.2.2. *Iterative construction.* The 4-step unbalanced 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 a graph $G = G_1 \textcircled{z} G_2$ 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, we define the iteration by

$$G_1 = H^3 = (N, M, c_2^2 d_2, c_2 d_2^2, \lambda^3),$$
$$G_{i+1} = G_i^3 \textcircled{z} H.$$

We now show that the above iterative technique yields a family of expanders.

Theorem 4. *Let H be a $(c_2 d_2^2, c_2^2 d_2, c_2, d_2, \lambda)$ 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 $(c_2^{4i} d_2^{5i}, c_2^{4i+1} d_2^{5i-1}, c_2^2 d_2, c_2 d_2^2, \lambda')$ 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 $n_i = n_{i-1}(c_2^4 d_2^5)$ and $m_i = m_{i-1}(c_2^4 d_2^5)$. Since $n_1 = c_2^4 d_2^5$ and $m_1 = c_2^5 d_2^4$, 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 $(c_2^2 d_2, c_2 d_2^2)$-regular.

Let $\lambda^{(i)}$ be the normalized second eigenvalue of G_i in absolute value. Using the result from Theorem 3, 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. □

7.3. 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, λ) graph. In [13], it is shown that the expansion of the replacement product graph is given by

$$\lambda \leq (p + (1-p)f(\lambda^{(1)}, \lambda^{(2)}))^{\frac{1}{3}},$$

where $p = \frac{d_2^2}{(d_2+1)^3}$ 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

$$G_{i+1} = (G_i)^4 Ⓡ H.$$

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

Theorem 5. *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 $G_{i+1} = (G_i)^4 Ⓡ H$. 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 $\lambda^{(i)}$ be the normalized second eigenvalue of G_i in absolute value. Using the result from Equation (7.3), 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 $\lambda^{(i)}$ converges 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. □

In order to obtain an expander family, the above iterative construction required taking one of the component graphs to its fourth power in the iterative step causing the size of the resulted iterated graph to grow fast. Observe, however, that if the above iteration was defined to be $G_{i+1} = (G_i)^2 Ⓡ H$, then for no choice of $\lambda^{(1)}, \lambda^{(2)}$, or d, would the resulting iterative family be expanders. That is, it can be shown that $\lambda(G_i) \to 1$ as $i \to \infty$, in such a case.

8. Conclusions

In this paper we generalized the zig-zag product resulting in a product suitable for unbalanced bipartite graphs. We proved that the resulting graphs are expander graphs as long as the component graphs are expanders. We examined the performance of LDPC codes obtained from zig-zag and replacement product graphs. The resulting product LDPC codes perform comparably to random LDPC codes with the additional advantage of having a compact description and an efficient hardware implementation**. We also introduced iterative constructions for the replacement product and a 4-step unbalanced zig-zag product yielding families of graphs with small, constant or slowly increasing degrees, and good expansion. Although these

** The graph vertices and the edges needed for processing during sum-product decoding can be activated systematically using a simple mathematical function that describes the entire product graph.

iterative schemes do not yield parameters for practical codes as yet, we believe they provide a first step in exploring iterated products for code construction. Designing iterative graph products that result in a family of good practical codes remains an intriguing open problem.

Appendix

Proof of Theorem 1.

Proof. We first set up the normalized adjacency matrix, M_G, of the zig-zag product graph G. Let H_1 denote the $N \times M$ incidence matrix of the vertices in V_1 with the vertices in W_1 in the graph G_1, and let H_2 be the $d_1 \times c_1$ incidence matrix of the the vertices in V_2 with the vertices in W_2 in the graph G_2. Then the normalized adjacency matrix[††] of the graph G_1 may be written as

$$M_1 = \frac{1}{\sqrt{c_1 d_1}} \begin{bmatrix} 0 & H_1 \\ H_1^T & 0 \end{bmatrix}$$

and the normalized adjacency matrix of the graph G_2 may be written as

$$M_2 = \frac{1}{\sqrt{c_2 d_2}} \begin{bmatrix} 0 & H_2 \\ H_2^T & 0 \end{bmatrix}.$$

Thus, the normalized adjacency matrix M_G for the zig-zag product graph G may be written as

$$M_G = \frac{1}{c_2 d_2} \begin{bmatrix} 0 & (H_2 \otimes I_n) A (H_2^T \otimes I_m) \\ (H_2^T \otimes I_m) A (H_2 \otimes I_n) & 0 \end{bmatrix},$$

where $\otimes$ denotes the Kronecker product and A is a permutation matrix of size $Nc_1 \times Nc_1$ that describes the zig-zag product connections.

The proof now follows the following steps:

• **Setting up the second largest eigenvalues of the matrices M_G, M_1, and M_2:**

Since M_G, M_1, and M_2 are all normalized, the largest eigenvalues of each matrix is 1. Let v_0 be the corresponding eigenvector for M_G, and observe it has the form $v_0 = [1, \cdots, 1, r, \cdots, r]^T$ with the value 1 in the first Nd_1 components and r in the remaining Mc_1 components, where $r = \frac{d_2}{c_2} = \frac{d_1}{c_1}$.

Let 1_x denote a column vector of length x with all entries equal to 1. With this notation, $v_0 = \begin{bmatrix} 1_{Nd_1} \\ r 1_{Mc_1} \end{bmatrix}$. In a similar way, we can write the eigenvector corresponding to the eigenvalue 1 of M_1 as $w_0 = \begin{bmatrix} 1_N \\ r_1 1_M \end{bmatrix}$, where $r_1 = \sqrt{\frac{d_1}{c_1}} = \sqrt{r}$. Similarly, $u_0 = \begin{bmatrix} 1_{d_1} \\ r_2 1_{c_1} \end{bmatrix}$ is an eigenvector of M_2 of eigenvalue 1, where $r_2 = \sqrt{\frac{d_2}{c_2}} = r_1 = \sqrt{r}$.

Let $\lambda = \lambda(G)$ be the second largest eigenvalue (in absolute value) of M_G, and 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_b has length Mc_1. Using the variational characterization as in [13],

$$\lambda(G) = \max_{\alpha \perp v_0} \frac{| < \alpha, M_G \alpha > |}{< \alpha, \alpha >} = \max_{\alpha \perp v_0} \frac{\| M_G \alpha \|}{\| \alpha \|},$$

[††]Note that the normalized adjacency matrix is the adjacency matrix divided by a scalar factor equal to its first eigenvalue. The normalized adjacency matrix therefore has the first eigenvalue equal to one.

where $< \alpha, \alpha >$ is the standard inner product in $\mathbb{R}^{Nd_1+Mc_1}$ and $\| \alpha \| = \sqrt{< \alpha, \alpha >}$.

Let e_m be a basis vector with the mth component equal to 1, and 0 in the remaining components. 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 nth vertex cloud, $n \in [N]$, of the graph G, and α_{b_m}, for $m \in [M]$, is defined similarly. Then we can write

$$\lambda = \max_{\alpha \perp v_0} \frac{|2(\alpha_a^T K \alpha_b)|}{< \alpha, \alpha >},$$

where $K = (H_2 \otimes I_N)A(H_2 \otimes I_M)$.

For the proof we need to show that $|2(\alpha_a^T K \alpha_b)|$ is sufficiently smaller than $< \alpha, \alpha >$. Set $s = |2(\alpha_a^T K \alpha_b)|$. First, note that using linear algebra, α_{a_n} may be uniquely decomposed as $\alpha_{a_n} = \alpha_{a_n}^{\parallel} + \alpha_{a_n}^{\perp}$, where $\alpha_{a_n}^{\parallel}$ is a vector parallel to the constant (non-zero) vector (i.e. all of its entries are the same), and $\alpha_{a_n}^{\perp}$ is a vector that is orthogonal to the constant (non-zero) vector (i.e. the sum of its entries is 0). Similarly for α_{b_m}. With this decomposition we can therefore write

$$s = 2 \left(\sum_{n\in[N]} ((\alpha_{a_n}^{\parallel})^T H_2 \otimes e_n) + \sum_{n\in[N]} ((\alpha_{a_n}^{\perp})^T H_2 \otimes e_m) \right) A \left(\sum_{m\in[M]} (H_2 \alpha_{b_m}^{\parallel} \otimes e_m) + \sum_{m\in[M]} (H_2 \alpha_{b_m}^{\perp} \otimes e_m) \right).$$

Our goal is then to show that $s \leq f(\lambda^{(1)}, \lambda^{(2)})(\| \alpha_a \|^2 + \| \alpha_b \|^2)$, where $f(\lambda^{(1)}, \lambda^{(2)})$ is an appropriately chosen positive valued function that satisfies $f(\lambda^{(1)}, \lambda^{(2)}) \leq \lambda^{(1)} + \lambda^{(2)} + (\lambda^{(2)})^2$, and $\lambda^{(1)} = \lambda(G_1)$ and $\lambda^{(2)} = \lambda(G_2)$.

• **Defining the properties of an eigenvector of M_G:**

The eigenvector $\alpha = \begin{bmatrix} \alpha_a \\ \alpha_b \end{bmatrix} \in \mathbb{R}^{Nd_1+Mc_1}$ of M_G is chosen such that $\alpha \perp v_0$ and has the following properties:

1. $\alpha_a \in \mathbb{R}^{Nd_1}$ and can be written as $\alpha_a = [\alpha_{a_1} \ \alpha_A \ \cdots \ \alpha_{a_N}]^T$, where $\alpha_{a_i} \in \mathbb{R}^{d_1}, i \in \{1, 2, \ldots, N\}$.
   Similarly, $\alpha_b \in \mathbb{R}^{Mc_1}$ can be written as $\alpha_b = [\alpha_{b_1} \ \alpha_{b_2} \ \cdots \ \alpha_{b_M}]^T$, where $\alpha_{b_i} \in \mathbb{R}^{c_1}, i \in \{1, 2, \ldots, M\}$.
2. Consider the component $\alpha_{a_n}^{\perp}$ of α_{a_n} that is perpendicular to the constant vector. Since $\begin{bmatrix} \alpha_{a_n}^{\perp} \\ 0 \end{bmatrix}$ is orthogonal to u_0 and $\lambda^{(2)}$ is the second largest eigenvalue of M_2, we have

$$\| M_2 \begin{bmatrix} \alpha_{a_n}^{\perp} \\ 0 \end{bmatrix} \| = \| \frac{1}{\sqrt{c_2 d_2}} \begin{bmatrix} 0 & H_2 \\ H_2^T & 0 \end{bmatrix} \begin{bmatrix} \alpha_{a_n}^{\perp} \\ 0 \end{bmatrix} \| \leq \lambda^{(2)} \| \begin{bmatrix} \alpha_{a_n}^{\perp} \\ 0 \end{bmatrix} \|,$$
$$\Rightarrow \| H_2^T \alpha_{a_n}^{\perp} \| \leq \lambda^{(2)} \| \alpha_{a_n}^{\perp} \|.$$

3. Consider the component $\alpha_{b_m}^{\parallel}$ of α_{b_m} that is parallel to the constant vector. Then using simple algebra it can be shown that $\| H_2 \alpha_{b_m}^{\parallel} \| \leq \| \alpha_{b_m}^{\parallel} \|$ (†)

• **Bounding the value of s defined for M_G above**:
We can rewrite s further as a sum of four parts, $s = s_1 + s_2 + s_3 + s_4$, where

$$s_1 = 2\left(\sum_{n\in[N]} (\alpha_{a_n}^{\|})^T H_2 \otimes e_n\right) A \left(\sum_{m\in[M]} H_2\alpha_{b_m}^{\|} \otimes e_m\right),$$

$$s_2 = 2\left(\sum_{n\in[N]} (\alpha_{a_n}^{\|})^T H_2 \otimes e_n\right) A \left(\sum_{m\in[M]} H_2\alpha_{b_m}^{\perp} \otimes e_m\right),$$

$$s_3 = 2\left(\sum_{n\in[N]} (\alpha_{a_n}^{\perp})^T H_2 \otimes e_n\right) A \left(\sum_{m\in[M]} H_2\alpha_{b_m}^{\|} \otimes e_m\right),$$

and

$$s_4 = 2\left(\sum_{n\in[N]} (\alpha_{a_n}^{\perp})^T H_2 \otimes e_n\right) A \left(\sum_{m\in[M]} H_2\alpha_{b_m}^{\perp} \otimes e_m\right).$$

We will bound each part of s separately.

1. Bounding s_4:
   Since A is a permutation matrix, we have

$$s_4 \leq 2 \parallel \sum_{n\in[N]} (\alpha_{a_n}^{\perp})^T H_2 \otimes e_n \parallel \parallel \sum_{m\in[M]} H_2\alpha_{b_m}^{T} \otimes e_m \parallel .$$

   Since $\parallel (\alpha_{a_n}^{\perp})^T H_2 \parallel \leq \lambda^{(2)} \parallel \alpha_{a_n}^{\perp} \parallel$, by definition of $\lambda^{(2)}$, (and similarly, $\parallel H_2\alpha_{b_m}^{\perp} \parallel \leq \lambda^{(2)} \parallel \alpha_{b_m}^{\perp} \parallel$), we have

$$\begin{aligned} s_4 &\leq 2 \parallel \sum_{n\in[N]} \lambda^{(2)}\alpha_{a_n}^{\perp} \otimes e_n \parallel \parallel \lambda^{(2)} \sum_{m\in[M]} \alpha_{b_m}^{\perp} \otimes e_m \parallel \\ &= 2(\lambda^{(2)})^2 \parallel \alpha_a^{\perp} \parallel \parallel \alpha_b^{\perp} \parallel \\ &\leq (\lambda^{(2)})^2(\parallel \alpha_a^{\perp} \parallel^2 + \parallel \alpha_b^{\perp} \parallel^2) \\ &= (\lambda^{(2)})^2(\parallel \alpha^{\perp} \parallel^2). \end{aligned}$$

2. Bounding s_3:
   Since A is a permutation matrix, we have

$$s_3 \leq 2 \parallel \sum_{n\in[N]} (\alpha_{a_n}^{\perp})^T H_2 \otimes e_n \parallel \parallel \sum_{m\in[M]} H_2\alpha_{b_m}^{\|} \otimes e_m \parallel .$$

   Using the argument from (†), and the fact that $\parallel H_2\alpha_{b_m}^{\|} \parallel \leq \parallel \alpha_{b_m}^{\|} \parallel$, we have

$$s_3 \leq 2\lambda^{(2)} \parallel \alpha_a^{\perp} \parallel \parallel \alpha_b^{\|} \parallel .$$

3. Bounding s_2:
   Similar to the argument for s_3, we can show that

$$s_2 \leq 2\lambda^{(2)} \parallel \alpha_a^{\|} \parallel \parallel \alpha_b^{\perp} \parallel .$$

4. Bounding s_1:
   To upper bound s_1, we first need to define functions that compute the average value of the components in each vertex cloud of the zig-zag product graph, G.

We 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 $\alpha_a'^{\|} = C(\alpha'_a) \otimes \frac{1_{d_1}}{d_1}$.

Similarly, we define a new vector $C'(\alpha'_b)$ for every α'_b such that its nth component is

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

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

The functions $C(\cdot)$ and $C'(\cdot)$ compute the average value of the components in each vertex cloud of the right and left vertex sets, respectively, of the zig-zag product graph G. Therefore, we have

$$C'(A(e_m \otimes \frac{1_{d_1}}{d_1})) = H_1 e_m.$$

Let $\tilde{a}_m$ denote the vertices on the left of the right side of the graph that represents the intermediate step of the construction but do not belong to the final vertex set of the zig-zag product graph, and $\tilde{b}_n$ denote the vertices on the right of the left side of the graph in the intermediate step of the construction but do not belong to the final vertex set of the zig-zag product graph. With this notation, we have

$$\alpha^{\|}_{b_m} H_2 = \alpha^{\|}_{\tilde{a}_m}$$

and

$$(\alpha^{\|}_{a_n})^T H_2 = \alpha^{\|}_{\tilde{b}_n}.$$

That is, since H_2 denotes the connections between the left vertices and right vertices, multiplying by H_2 takes $(\alpha^{\|}_{a_n})^T$ to $\alpha^{\|}_{\tilde{b}_n}$ and $\alpha^{\|}_{b_m}$ to $\alpha^{\|}_{\tilde{a}_m}$. Then we can rewrite s_1 as

$$\begin{aligned} s_1 &= 2(\sum_{n\in[N]} (\alpha^{\|}_{a_n})^T H_2 \otimes e_n)^T A(\sum_{m\in[M]} \alpha^{\|}_{b_m} H_2 \otimes e_m) \\ &= 2(\sum_{n\in[N]} \alpha^{\|}_{\tilde{b}_n} \otimes e_n)^T A(\sum_{m\in[M]} \alpha^{\|}_{\tilde{a}_m} \otimes e_m). \end{aligned}$$

Since

$$\alpha^{\|}_{\tilde{a}} = \sum_{m\in[M]} (\alpha^{\|}_{\tilde{a}_m} \otimes e_m) = C(\alpha_{\tilde{a}}) \otimes \frac{1_{d_1}}{d_1}$$

and

$$\alpha^{\|}_{\tilde{b}} = \sum_{n\in[N]} (\alpha^{\|}_{\tilde{b}_n} \otimes e_n) = (C'(\alpha_{\tilde{b}}) \otimes \frac{1_{c_1}}{c_1},$$

we have

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

This implies that

$$s_1 = \frac{2(C'(\alpha_{\tilde{b}}))^T H_1(C(\alpha_{\tilde{a}}))}{c_1 d_1} = \frac{2(\frac{1}{\sqrt{c_1}}C'(\alpha_{\tilde{b}}))^T H_1(\frac{1}{\sqrt{d_1}}C(\alpha_{\tilde{a}}))}{\sqrt{c_1}\sqrt{d_1}}.$$

Moreover, we can show 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 $\begin{bmatrix} 1_N \\ \sqrt{r}1_M \end{bmatrix}$.

To see this, first note that

$$\sum_{b=1}^{c_1}\sum_{n=1}^{N}\alpha_{\tilde{b}_n} = c_2\sum_{a=1}^{d_1}\sum_{n=1}^{N}\alpha_{a_n}, \quad \text{and} \quad \sum_{a=1}^{d_1}\sum_{m=1}^{M}\alpha_{\tilde{a}_m} = d_2\sum_{b=1}^{c_1}\sum_{m=1}^{M}\alpha_{b_m}. \quad (*)$$

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 can be shown that $\begin{bmatrix} \alpha_a^{\|} \\ \alpha_b^{\|} \end{bmatrix} = \begin{bmatrix} C'(\alpha_a)\otimes\frac{1_{d_1}}{d_1} \\ C(\alpha_b)\otimes\frac{1_{c_1}}{c_1} \end{bmatrix}$ is orthogonal to $\begin{bmatrix} 1_{Nd_1} \\ r1_{Mc_1} \end{bmatrix}$. From this, it is easy to verify that

$$< \begin{bmatrix} \frac{1}{\sqrt{c_1}}C'(\alpha_{\tilde{b}}) \\ \frac{1}{\sqrt{d_1}}C(\alpha_{\tilde{a}}) \end{bmatrix}, \begin{bmatrix} 1_N \\ \sqrt{r}1_M \end{bmatrix} >$$
$$= \frac{1}{\sqrt{c_1}}\sum_{b=1}^{c_1}\sum_{n=1}^{N}\alpha_{\tilde{b}_n} + \frac{1}{\sqrt{d_1}}\frac{\sqrt{d_1}}{\sqrt{c_1}}\sum_{a=1}^{d_1}\sum_{m=1}^{M}\alpha_{\tilde{a}_m} = 0.$$

Hence, from the definition of λ_1, we have

$$\begin{aligned} s_1 &= \frac{2(\frac{1}{\sqrt{c_1}}C'(\alpha_{\tilde{b}}))^T H_1(\frac{1}{\sqrt{d_1}}C(\alpha_{\tilde{a}}))}{c_1 d_1} \\ &\le \frac{\lambda_1}{c_1 d_1}(\frac{1}{c_1}\parallel C'(\alpha_{\tilde{b}})\parallel^2 + \frac{1}{d_1}\parallel C(\alpha_{\tilde{a}})\parallel^2) \\ &\le \lambda_1(\parallel \alpha_a^{\|}\parallel^2 + \parallel \alpha_b^{\|}\parallel^2) \\ &= \lambda_1(\parallel \alpha^{\|}\parallel^2). \end{aligned}$$

The last inequality above is obtained by some simple algebraic manipulation and using the relation in $(*)$.

• **Obtaining an upper bound on λ:**

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

$$\begin{aligned} s &= s_1 + s_2 + s_3 + s_4 \\ &\le \lambda^{(1)}(\parallel \alpha^{\|}\parallel^2) + 2\lambda^{(2)}(\parallel \alpha_a^{\perp}\parallel\parallel \alpha_b^{\|}\parallel + \parallel \alpha_a^{\|}\parallel\parallel \alpha_b^{\perp}\parallel) + (\lambda^{(2)})^2(\parallel \alpha^{\perp}\parallel^2). \end{aligned}$$

However, observe that

$$\begin{aligned} &2\lambda^{(2)}(\parallel \alpha_a^{\perp}\parallel\parallel \alpha_b^{\|}\parallel + \parallel \alpha_a^{\|}\parallel\parallel \alpha_b^{\perp}\parallel) \\ &\quad \le \lambda^{(2)}(\parallel \alpha_a^{\|}\parallel^2 + \parallel \alpha_a^{\perp}\parallel^2 + \parallel \alpha_b^{\|}\parallel^2 + \parallel \alpha_b^{\perp}\parallel^2) = \lambda^{(2)}(\parallel \alpha\parallel^2), \end{aligned}$$

and further

$$\parallel \alpha^{\|}\parallel^2 \le \parallel \alpha\parallel^2 \quad \text{and} \quad \parallel \alpha^{\perp}\parallel^2 \le \parallel \alpha\parallel^2 .$$

Thus, we have

$$s \le (\lambda^{(1)} + \lambda^{(2)} + (\lambda^{(2)})^2)(\parallel \alpha\parallel^2).$$

Recall that the second largest eigenvalue of M_G is defined as $\lambda = \max_{\alpha \perp v_0} \frac{s}{<\alpha,\alpha>}$, where $s = 2\alpha_a^T(H_2 \otimes I_N)A(H_2 \otimes I_M)\alpha_b$. Using the upper bound on s, we obtain

$$\lambda \leq \frac{(\lambda^{(1)} + \lambda^{(2)} + (\lambda^{(2)})^2)(\| \alpha \|^2)}{\| \alpha \|^2} = \lambda^{(1)} + \lambda^{(2)} + (\lambda^{(2)})^2.$$

• **Showing that** $\lambda^{(1)} < 1, \lambda^{(2)} < 1 \Rightarrow \lambda < 1$:
The only remaining step is to show that if $\lambda^{(1)} < 1$ and $\lambda^{(2)} < 1$, then $\lambda < 1$. Let $\lambda^{(1)} < 1, \lambda^{(2)} < 1$ and suppose for our first case that $\| \alpha^{\perp} \| \leq \frac{1-\lambda^{(1)}}{3\lambda^{(2)}} \| \alpha \|$. Then, we can upper bound s as follows

$$\begin{aligned} s &\leq \lambda^{(1)} \| \alpha^{\|} \|^2 + 2\lambda^{(2)} \| \alpha^{\|} \| \| \alpha^{\perp} \| + (\lambda^{(2)})^2 \| \alpha^{\perp} \|^2 \\ &\leq \| \alpha \|^2 + \frac{2(1-\lambda^{(1)})}{3} \| \alpha \|^2 + \frac{(1-\lambda^{(1)})^2}{9} \| \alpha \|^2 \\ &= (1 - \frac{1-\lambda^{(1)}}{3})^2 \| \alpha \|^2 \\ &\leq \| \alpha \|^2 . \end{aligned}$$

Now suppose $\| \alpha^{\perp} \| > \frac{1-\lambda^{(1)}}{3\lambda^{(2)}} \| \alpha \|$.

Since $s = 2(\alpha_a^{\|} + \alpha_a^{\perp})(\sum_{n\in[N]} H_2^T \otimes e_n)A(\sum_{m\in[M]} H_2 \otimes e_m)(\alpha_b^{\|} + \alpha_b^{\perp})$ we can rewrite s in this case as

$$s = 2(\alpha_{\tilde{b}}^{\|} + \sum_{n\in[N]} \alpha_{a_n}^{\perp}(H_2^T) \otimes e_n)A(\alpha_{\tilde{a}}^{\|} + \sum_{m\in[M]} H_2\alpha_{b_m}^{\perp} \otimes e_m.$$

However, $\sum_{n\in[N]} \alpha_{a_n}^{\perp}(H_2^T) \otimes e_n$ is orthogonal to $\alpha_{\tilde{a}}^{\|}$ and $\sum_{m\in[M]} H_2\alpha_{b_m}^{\perp} \otimes e_m$ is orthogonal to $\alpha_{\tilde{b}}^{\|}$. Thus,

$$s = 2(\alpha_{\tilde{b}}^{\|})A(\alpha_{\tilde{a}}^{\|}) + 2(\sum_{n\in[N]} \alpha_{a_n}^{\perp}(H_2^T) \otimes e_n)A(\sum_{m\in[M]} H_2\alpha_{b_n}^{\perp} \otimes e_m).$$

From the previous arguments, we have

$$\begin{aligned} s &\leq \lambda^{(1)}(\| \alpha^{\|} \|^2) + (\lambda^{(2)})^2(\| \alpha^{\perp} \|^2) \\ &= \lambda^{(1)}(\| \alpha \|^2 - \| \alpha^{\perp} \|^2) + (\lambda^{(2)})^2 \| \alpha^{\perp} \|^2 \\ &\leq (\| \alpha \|^2 - \| \alpha^{\perp} \|^2) + (\lambda^{(2)})^2 \| \alpha^{\perp} \|^2 \\ &= (1 - \frac{(1-\lambda^{(1)})^2(1-(\lambda^{(2)})^2)}{9}) \| \alpha \|^2 \\ &\leq \| \alpha \|^2 . \end{aligned}$$

This completes the proof. □

REFERENCES

[1] N. Alon, *Eigenvalues and expanders*, Combinatorica, **6** (1986), 83–96.

[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), 630–637.

[3] S. Hoory, N. Linial and A. Wigderson, *Expander graphs and their applications*, Bull. Amer. Math. Soc. (N.S.) (electronic), **43** (2006), 439–561.

[4] W. Imrich and S. Klavžar, "Product Graphs," Wiley-Interscience, New York, 2000.
[5] H. Janwa and A. K. Lal, *On Tanner codes: minimum distance and decoding*, Appl. Algebra Engrg. Comm. Comput., **13** (2003), 335–347.
[6] 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," (2003), 1280–1289.
[7] C. A. Kelley and D. Sridhara, *Eigenvalue bounds on the pseudocodeword weight of expander codes*, Adv. Math. Commun., **1** (2007), 287–306.
[8] J. Lafferty and D. Rockmore, *Codes and iterative decoding on algebraic expander graphs*, in "Proceedings of ISITA 2000," Honolulu, Hawaii, November 2000.
[9] A. Lubotzky, "Discrete Groups, Expanding Graphs and Invariant Measures" (with an appendix by J.D. Rogawski), Birkhäuser Verlag, Basel, 1994.
[10] A. Lubotzky, R. Phillips and P. Sarnak, *Ramanujan graphs*, Combinatorica, **8** (1988), 261–277.
[11] G. A. Margulis, *Explicit group-theoretic constructions of combinatorial schemes and their applications in the construction of expanders and concentrators* (in Russian), Problemy Peredachi Informatsii, **24** (1988), 51–60. (English transl. in: Problems Inform. Transmission, **24** (1988), 39–46.)
[12] R. Meshulam and A. Wigderson, *Expanders in group algebras*, Combinatorica, **24** (2004), 659–680.
[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** (2002), 157–187.
[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," (2000), 248–257.
[15] M. Sipser and D. A. Spielman, *Expander codes*, IEEE Trans. Inform. Theory, **42** (1996), 1710–1722.
[16] R. M. Tanner, *A recursive approach to low complexity codes*, IEEE Trans. Inform. Theory, **27** (1981), 533–547.
[17] R. M. Tanner, *Explicit concentrators from generalized N-gons*, SIAM J. Algebraic Discrete Methods, **5** (1984), 287–293.
[18] J.-P. Tillich and G. Zémor, *Optimal cycle codes constructed from Ramanujan graphs*, SIAM J. Discrete Math., **10** (1997), 447–459.
[19] J. K. Wolf, *Efficient maximum likelihood decoding of linear block codes using a trellis*, IEEE Trans. Inform. Theory, **IT-24** (1978), 76–80.

Received August 2007; revised August 2008.

E-mail address: ckelley2@math.unl.edu
E-mail address: deepak.sridhara@seagate.com
E-mail address: rosenthal@math.uzh.ch