2013

Algebraic Design and Implementation of Protograph Codes using Non-Commuting Permutation Matrices

Christine A. Kelley
University of Nebraska-Lincoln, ckelley2@unl.edu

Pages: 910 - 918, DOI: 10.1109/TCOMM.2013.012313.110513

Algebraic Design and Implementation of Protograph Codes using Non-Commuting Permutation Matrices

Christine A. Kelley, *Member, IEEE*

***Abstract*—Random lifts of graphs, or equivalently, random permutation matrices, have been used to construct good families of codes known as protograph codes. An algebraic analog of this approach was recently presented using voltage graphs, and it was shown that many existing algebraic constructions of graph-based codes that use commuting permutation matrices may be seen as special cases of voltage graph codes. Voltage graphs are graphs that have an element of a finite group assigned to each edge, and the assignment determines a specific lift of the graph. In this paper we discuss how assignments of permutation group elements to the edges of a base graph affect the properties of the lifted graph and corresponding codes, and present a construction method of LDPC code ensembles based on non-commuting permutation matrices. We also show encoder and decoder implementations for these codes.**

***Index Terms*—Low density parity check codes, voltage graphs, iterative decoding, graph lift.**

I. Introduction

CODES on graphs and iterative decoders have been shown to achieve near-capacity performance on several communication channels and have replaced classical codes in many practical applications. Much work has focused on understanding the asymptotic performance of ensembles of these codes for block lengths tending to infinity. For practical implementation, the design of short to moderate length codes with algebraic structure is desired. One approach is to design the graphs for these codes by taking random lifts of a suitably chosen base graph, or *protograph* [1], [2], [3]. The properties of the base graph influence the properties of the graph lift and resulting codes. Indeed, random lifts of graphs have been heavily studied (e.g., [13], [14], [15]). Among other advantages, these codes can be represented efficiently and perform well compared to randomly designed codes with comparable parameters.

In this paper, we design codes from specific lifts of graphs that are obtained algebraically using voltage graphs, in which group elements are assigned to the edges of the base graph that determine the edge set in the lift. Voltage graphs, originally coined in topological graph theory [4] in the study of embedding graphs on surfaces, may be observed in many well-known families of codes whose underlying graph representations may be interpreted as voltage graphs [5]. For example, quasi-cyclic LDPC codes based on blocks of shifted-identity matrices, array codes, quasi-cyclic repeat accumulate codes, and others fall into this category [6], [7], [8], [10], [11]. The voltage graph approach is a powerful tool for analyzing graph properties of the resulting graph lifts using the properties of the base graph. Voltage graphs have been used to obtain many instances of graphs with extremal properties; see e.g. [16], [17], [18], [19], and thus provide a promising approach for code design.

We outline an algebraic technique of specifying the voltage assignments by restricting the voltage assignments to a permutation group designed in a special way. From the matrix perspective, our construction of protograph codes uses non-commuting permutation matrices. This is in contrast to the large body of work on constructions based on shifted identity permutation matrices. Our method, which may be applied to any base graph or protograph, yields codes having good properties including efficient hardware implementation. Paper [5] initiated the study of voltage graphs for codes, and contains a classification of subgraphs that always cause cycles in the lifted graph for any assignment of commuting group elements to a base graph. In [12], we developed some initial constructions using voltage graphs. This paper provides a general construction method of code ensembles and theoretical results.

Section 2 contains a brief background on voltage graphs. In Section 3 we describe the connection between the voltages in the base graph and the structure of the derived graph. In Section 4, we present a general construction method for assigning voltages from a nonabelian group to the edges of a base graph, and give two explicit examples of algebraic protograph code ensembles obtained by this method[1]. The girth and minimum distance of the codes is discussed in Section 5, with simulation results in Section 6. In Section 7, we use the algebraic structure of the codes to obtain efficient encoder and decoder implementations. Section 8 concludes the paper.

II. Voltage Graphs and LDPC Codes

In this paper we use the theory of voltage graphs to obtain lifts of graphs algebraically. The resulting graphs are designed to be suitable for coding while also retaining some of the desirable random-like characteristics of random lifts. It is worth noting that the voltage graph framework also provides a tool to analyze successful random protograph codes by examining their specific permutations (i.e. voltage assignments)

Manuscript received August 8, 2011; revised March 11, 2012. The associate editor coordinating the review of this paper and approving it for publication was T.-K. Truong.

C. A. Kelley is with the Department of Mathematics, University of Nebraska-Lincoln, Lincoln, NE, 68588, USA (e-mail: ckelley2@math.unl.edu).

This work was supported in part by NSF Grant EPS-0701892 and by the National Security Agency under Grant Number H98230-11-1-0156. The United States Government is authorized to reproduce and distribute reprints not-withstanding any copyright notation herein.

Digital Object Identifier 10.1109/TCOMM.2013.012313.110513

[1]Some of this work was partially published in [12].

in hindsight. This section provides a background on voltage graphs.

An algebraic construction of specific covering spaces for graphs was introduced by Gross and Tucker in the 1970s [4]. Since we will use permutation groups in our construction, we will focus on *permutation voltage graphs*, which will be defined shortly. We refer the reader to [4], [12] for a discussion on *ordinary* voltage graphs, which generate lifts of graphs using arbitrary finite groups. Before we can define permutation voltage graphs, we first introduce some notation. Let $\{[n]\}$ denote the set of integers from 1 to n. Any permutation has a unique decomposition into a product of cyclic permutations, each called a *cycle*. For example, $\sigma = (1572)(34)(6)$ is a permutation of $\{[7]\}$ in cycle representation, in which the numbers within a set of parentheses form a *cycle*, and each number in a cycle is mapped to the number to its right, except for the last number which is mapped to the first number in the cycle. We will use this notation to illustrate permutation voltage graphs and our code constructions.

A *permutation voltage graph* $\mathcal{G} = (V_\mathcal{G}, E_\mathcal{G})$ is a base graph where each edge is assigned an element from a chosen permutation group G on n elements, called the *voltage group*. Specifically, each edge in $\mathcal{G}$ is arbitrarily assigned an orientation, and a function α, called a *permutation voltage assignment*, maps the positive orientation of each edge to an element from G. The values of α on the edges are called *voltages*, and the negative orientation of each edge is assigned the inverse element of its voltage. For example, if $e = (u, v) \in E_\mathcal{G}$, then the (positive) orientation of e, denoted e^+, is either "from u to v" or "from v to u". The assignment $\alpha(e) = \sigma \in \mathsf{G}$ means that the permutation σ is assigned to e^+, whereas the negative (or, reverse) orientation of e, denoted e^-, is assigned σ^{-1} under α. [2]

The base graph $\mathcal{G}$ together with α form the permutation voltage graph. We now explain how the permutation voltage graph determines a specific lift of the graph, $\mathcal{G}^\alpha$, called the *(permutation) derived graph*. If G is a subgroup of S_n, the group of all permutations of n elements (i.e. the symmetric group), then $\mathcal{G}^\alpha$ is a degree n lift of $\mathcal{G}$ with vertex set $V_\mathcal{G} \times \{1, \ldots, n\}$ and edge set $E_\mathcal{G} \times \{1, \ldots, n\}$. If $\pi \in \mathsf{G}$ is a permutation voltage on the edge e oriented from u to v in $\mathcal{G}$, then there is an edge from (u, i) to $(v, \pi(i))$ in $\mathcal{G}^\alpha$ for $i \in \{[n]\}$. We will represent each vertex (v, i) and edge (e, i) in the derived graph by v_i and e_i, respectively. The set of vertices $\{v_i | i = 1, 2 \ldots, n\}$ in the derived graph is called the *cloud* of v, and similarly for edges. The clouds contain precisely those elements in the pre-image of a vertex (or edge) under the *natural projection mapping* $p : \mathcal{G}^\alpha \to \mathcal{G}$.

Figure 1 shows a permutation voltage graph $\mathcal{G}$ with voltages from S_3, and the corresponding derived graph $\mathcal{G}^\alpha$. Figure 2 shows a bipartite permutation voltage graph with voltages from S_3, and its derived graph that may be regarded as a Tanner graph for a code of length 9. The shaded circles and squares represent variable nodes and check nodes, respectively.

A *walk* W in the permutation voltage graph $\mathcal{G}$ with voltage assignment α may be represented by the sequence of oriented edges in the order they are traversed, e.g. $W = e_1^{\sigma_1} e_2^{\sigma_2} \ldots e_m^{\sigma_m}$ where $e_1, \ldots, e_m$ are edges in $\mathcal{G}$ and each σ_i is $+$ or $-$ denoting the direction edge e_i is traversed. The *net voltage* of the walk W is defined as the voltage group product $\alpha(e_1^{\sigma_1})\alpha(e_2^{\sigma_2}) \ldots \alpha(e_m^{\sigma_m})$ of the voltages on the edges of W in the order and direction of the walk. For example, the walk $W = z^+ y^- x^+$ in Figure 1 has net voltage $(12)(3) \times (1)(23) \times (123) = (1)(2)(3)$.

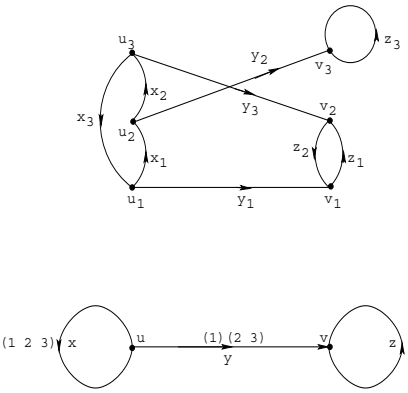

Fig. 1. A permutation voltage graph with voltage group S_3 on the bottom, and its derived graph on the top.

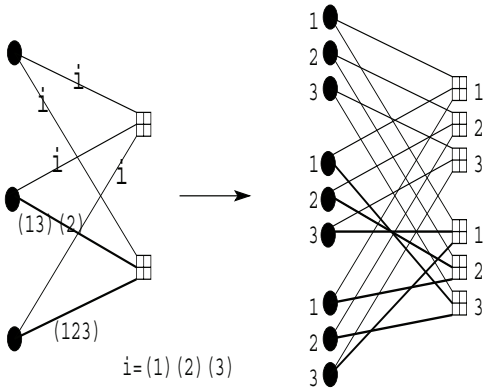

Fig. 2. A permutation voltage graph with voltage group S_3 on the left, and its derived graph on the right, where i denotes the identity permutation.

Theorem 2.1: [4] Let W be a walk with initial vertex v in a voltage graph $\mathcal{G}$ with voltages from a permutation group G on n elements. Then for each vertex v_i in $\mathcal{G}^\alpha$, where $i \in \{[n]\}$, there is a unique walk W_i in $\mathcal{G}^\alpha$ that starts at v_i and projects down[3] to W.

A walk of length m is *closed* if it starts and ends at the same vertex, and *backtrackless* if $e_i \neq e_{i+1}$ for $1 \leq i \leq m-1$. A backtrackless closed walk is said to be *tailless* if $e_m \neq e_1$. For example, in Figure 1, the walk $y^+ z^+ y^-$ is closed, backtrackless, but not tailless, whereas the walk $y^+ z^+ y^- x^+$ is closed, backtrackless, and tailless. The walk $x^+ y^+ y^-$ is not backtrackless. A useful consequence of Theorem 2.1 is as follows.

Corollary 2.2: Assume $W = e_1^{\sigma_1} e_2^{\sigma_2} \cdots e_m^{\sigma_m}$ is closed,

[2]While each edge has a notion of positive and negative orientation, the graph remains undirected. For example, walks and cycles may traverse edges in either direction but it will be important to keep track of the group elements that are assigned to the given orientations of each edge used.

[3]A walk W^α in the derived graph $\mathcal{G}^\alpha$ projects down to W if the edges in W^α are mapped onto the edges of W by the natural projection mapping in the exact order and orientation of W.

backtrackless, and tailless. Then W_i, for any $i \in \{[n]\}$, is a cycle on $\mathcal{G}^\alpha$ if and only if the net voltage of W is the identity of G.

III. Influence of Voltages on Tanner Graph Properties

Permutation voltage graphs provide a natural algebraic analog to random protograph codes. In this section we explain how to choose a permutation group and assign voltages to ensure that the derived graph is connected and has a good cycle structure. These guidelines were initially proposed in [12] and will be used in the new constructions in Section 4.

1)*Choose a non-abelian voltage group.* In [5], we provide a classification of subgraphs that, if present in a base graph, give rise to certain cycles in the derived graph for any assignment of voltages from an abelian group G[4]. The result in [5] gives an alternative short proof that the girth is always at most 12 for codes constructed using commuting permutation matrices in arrays that contain a sub-array of 2×3 non-zero permutation matrices. This girth limitation was originally shown in [7] and later by others (see e.g., [20]). Many researchers have also noted that the use of commuting permutation matrices, such as shifted identity matrices, leads to restrictions on the minimum distance of the associated codes [8], [21], [22], [23], [24], [25], [26]. This indicates that non-commuting voltage assignments have a greater potential in yielding derived graphs with larger girth and improved minimum distance.

2) *Choose permutations that do not have fixed points or cycles of length ≤ 3 in their cycle representation, and use pairwise non-commuting permutations as voltages.* Since non-abelian groups may contain abelian subgroups, the voltages assigned should be pairwise non-commuting. This alone is not enough to guarantee large girth in the lift. Rather, the cycle structure of the lifted graph relates directly to the structure of the net voltages of cycles in the base graph. A *j-cycle of a permutation π* will refer to a cyclic permutation of j elements in the cycle decomposition of π, and should not be confused with a j-cycle in a graph which is a closed path containing j edges. The *cycle structure* of a permutation in S_n is a vector $(c_1, \ldots, c_n)$ where c_j denotes the number of j-cycles in the cycle decomposition of the permutation. For example, $\sigma = (1572)(34)(6)$ contains one 4-cycle, one 2-cycle, and one 1-cycle (i.e. *fixed point*) and has cycle structure $(1, 1, 0, 1, 0, 0, 0)$. The pre-image of each cycle in a permutation voltage graph under the natural projection mapping consists of a union of disjoint cycles in the derived graph. The following result from [4] explains how the length and number of these cycles in the lift are determined.

Theorem 3.1: [4] Let C be a k-cycle with net voltage π in a permutation voltage graph, and let $(c_1, c_2, \ldots, c_n)$ be the cycle structure of π. Then the pre-image of C in the derived graph consists of $c_1 + c_2 + \cdots + c_n$ disjoint cycles, including for each $j \in \{[n]\}$, exactly c_j kj-cycles.

To continue the example, if $\sigma = (1572)(34)(6)$ is the net voltage of a k-cycle C in $\mathcal{G}$ that starts at u, then the pre-image of C in $\mathcal{G}^\alpha$ consists of one cycle of length $4k$ that contains vertices u_1, u_2, u_5, and u_7, one cycle of length $2k$ that contains vertices u_3 and u_4, and one cycle of length k that contains vertex u_6. Thus, since 6 is a fixed point of σ, a cycle of length equal to k occurs in the derived graph. A result similar to Theorem 3.1 was observed in [8].

Due to Theorem 3.1 and Corollary 2.2, the permutation voltages should not have fixed points or cycles of length ≤ 3 in their cycle representation. This will allow our code constructions to surpass the girth 12 restriction that exists in the abelian case, provided that there are no short *products* of these voltages that yield permutations with small cycles in their decomposition. Moreover, we will choose a voltage group where the only group element with fixed points is the identity permutation. This will eliminate fixed points in the net voltages of all graph cycles that do not have the identity permutation as a net voltage.

3) *Use a permutation voltage group whose action on $\{[n]\}$ has just one orbit, and assign voltages that generate the group.* First, to simplify the voltage assignment process, it is enough to assign voltages so that the edges of a spanning tree receive the identity element [4], [19][5]. A spanning tree is a connected spanning subgraph that is a tree. The edges outside of a given spanning tree form the co-tree. For a voltage assignment that assigns the identity element to the edges of spanning tree, the *local voltage group* G' is the group generated by the voltages assigned to the co-tree. In [4], [19] the authors show that the number of connected components in the corresponding derived graph is equal to number of orbits in the action[6] of G' on $\{[n]\}$. Since we want one component in the lift (i.e. a connected graph), our constructions will use a permutation group G whose action yields a single orbit, and we will assign voltages to the co-tree that generate the group G. Thus, $\mathsf{G} \cong \mathsf{G}'$ and the derived graph will be connected. In contrast, randomly chosen permutation voltages may yield disconnected derived graphs.

A. Summary of guidelines for voltage assignments

To surpass the girth limitations of earlier code constructions, voltages should be chosen from a nonabelian group so that they are pairwise non-commuting. (This is stricter than necessary since some pairwise commuting elements are fine if placed appropriately in the base graph.) The net voltages on short cycles in the base graph should not have short cycles in their permutation cycle decompositions. Each edge of an arbitrarily chosen spanning tree should be assigned the identity group element as a voltage, and the edges in the co-tree should be assigned voltages that generate the entire voltage group to ensure connectivity.

[4]The elements commute. That is, for each $a, b \in \mathsf{G}$, $ab = ba$.

[5]Specifically, in [4], [19] it is shown that for any assignment α of edges in $\mathcal{G}$ to voltages in a group G and for any spanning tree $\mathcal{T}$ of $\mathcal{G}$, it is possible to find a voltage assignment α' of edges in $\mathcal{G}$ to G, where the edges of $\mathcal{T}$ are assigned the identity element of G under α' and the resulting graphs $\mathcal{G}^\alpha$ and $\mathcal{G}^{\alpha'}$ are isomorphic.

[6]see e.g. Section 2.4 of [34] for the definitions of group action and orbit.

IV. LDPC Codes from the Nonabelian Group of Order pq

In this section, we present a general method for constructing voltage assignments for code design, and give an explicit example of a code ensemble obtained by this method.

A. Method for assigning voltages

We give a general method for assigning elements from a nonabelian group G to the edges of a base graph. Our method may be applied to any non-abelian voltage group to obtain derived graphs that will be interpreted as Tanner graphs for LDPC codes. Let N_{pq} denote the nonabelian group of order pq for primes p and q with $q|p-1$. N_{pq} is the only nonabelian group (up to isomorphism) of order $m = pq$ where $q|(p-1)$, and if $q \nmid (p-1)$, then there is no nonabelian group of order pq. We will illustrate our method using the group N_{pq}, and the complete bipartite graph $K_{j,k}$ as the base graph, but the method and constructions may be easily applied to any base graph simply by removing some of the edges (and corresponding voltages) or by applying the same guidelines to multiple (or, parallel) edges.

1) *Start with a base graph $\mathcal{G}$ on j check nodes and k variable nodes and orient the edges from the variable nodes to the check nodes.* Permutations will be assigned to these positively oriented edges, and the negative orientation of each edge will be assigned the inverse permutation. The parity-check matrix of the resulting LDPC code is a $j \times k$ array of $m \times m$ permutation matrices determined by the following steps. Note that if an edge is not present in the base graph, then an $m \times m$ all-zero matrix will be used to represent its voltage, and if multiple edges are present, then a superposition of the corresponding permutation matrices is used. For our example, let $\mathcal{G} = K_{j,k}$.

2) *Choose a non-abelian group G of order m, and label the elements of G from 1 to m. Let G act on itself by left multiplication to obtain an isomorphic group P, where P is a permutation group of order m. Let P be the permutation voltage group.* In our examples, P will be the permutation group isomorphic to N_{pq}. Note that P is a subgroup of S_m and has the desirable property that the only element in P with a fixed point is the identity permutation. We construct the group N_{pq} generated by elements c and d of orders p and q, respectively, such that $dc = c^s d$, where $s \not\equiv 1 (\mathrm{mod}\ p)$ and $s^q \equiv 1 (\mathrm{mod}\ p)$ (See Chapter 2 of [34] for more detail on this group, and Example 4.1 in this section for an illustration of this step.)

3) *Choose a spanning tree of the base graph and assign each of its edges the identity permutation i. For the remaining edges, choose nontrivial pairwise non-commuting voltages such that they generate the group P.* Without loss of generality, let the edges in $K_{j,k}$ corresponding to the first row and column of the $j \times k$ array be the spanning tree and assign each of them the identity permutation in P. This leaves $(j-1)(k-1)$ non-identity permutations to assign to the edges of the co-tree. The group P contains one cyclic subgroup of order p, namely the one generated by c, and p cyclic subgroups of order q, namely the ones generated by $c^i d$ for $i = 0, 1, \ldots, p-1$. Permutations chosen from the same cyclic subgroup will commute, therefore choose at most one element from each for the remaining edges. For this to be possible, the number of edges in the co-tree should be at most $p+1$. Moreover, since the identity permutation is the only element in P with a fixed point, the Orbit-Counting Lemma (see e.g. [35]) ensures that P has just one orbit when acting on $\{1, 2, \ldots, m\}$, where $m = |\mathsf{P}|$. Thus, the chosen nontrivial permutations should generate P to ensure that the condition for connectivity is met.

When using N_{pq}, q must be larger than 3 to achieve girth larger than 12 when $\mathcal{G}$ has a 4-cycle. If $q = 3$, P will have permutations of order 3 with 3-cycles in their cycle decompositions. The smallest pq that satisfies these constraints is $m = 55$, where $q = 5$.

Example 4.1: We illustrate Step 2 using the nonabelian group N_6 with $p = 3$ and $q = 2$. N_6 has two generators, c of order 3 and d of order 2, with the relation that $c^2 d = dc$. The elements of N_6 are $\{1, c, c^2, d, cd, c^2 d = dc\}$. Order the elements, e.g., $1 \mapsto 1, c \mapsto 2, c^2 \mapsto 3, d \mapsto 4, cd \mapsto 5, c^2 d \mapsto 6$. The action of c on N_6 by left multiplication yields the set $\{c \cdot g | g \in N_6\} = \{c, c^2, 1, cd, c^2 d, d\} = \{2, 3, 1, 5, 6, 4\}$. This means that $1 \mapsto 2, 2 \mapsto 3, 3 \mapsto 1, 4 \mapsto 5, 5 \mapsto 6, 6 \mapsto 4$, which corresponds to the permutation $(123)(456)$. Similarly, the permutation corresponding to the action of d is $(14)(26)(35)$, of c^2 is $(132)(465)$, of cd is $(15)(24)(36)$, and of $c^2 d$ is $(16)(25)(34)$. The action of the identity permutation $i = (1)(2)(3)(4)(5)(6)$ on N_6 gives i. Thus, P is the subgroup of S_6 containing these six permutations.

The following groups will be used in the forthcoming constructions. The explicit cycle representations for the permutations c and d in both groups may be found in [12].

Example 4.2: Let $m = 55$, so $m = pq$ where $p = 11$ and $q = 5$. Then N_{55} is obtained by two generators, c of order 11 and d of order 5, with the relation that $c^3 d = dc$. The action of c and d on N_{55} yields the isomorphic nonabelian permutation group P of order 55 consisting of the elements $\{c^i d^j | i = 0, \ldots, 10 \text{ and } j = 0, \ldots, 4\}$ where, with an abuse of notation, c and d are the corresponding generating permutations of P. Similarly, when $p = 29$ and $q = 7$, the group N_{203} is obtained by two generators, c of order 29 and d of order 7, with the relation that $c^7 d = dc$. The action of c and d on N_{203} yields the isomorphic nonabelian permutation group P of order 203 consisting of the elements $\{c^i d^j | i = 0, \ldots, 28 \text{ and } j = 0, \ldots, 6\}$.

B. Construction Example

Using the permutation group $\mathsf{P} \cong N_{pq}$, we form the following $j \times k$ matrix M with $j \leq k$ and entries in P that has as its (a,b)th element $M_{a,b}$, the entry $(d^{a+b}c)^{t \cdot a}$ for some fixed integer $0 < t \leq q-1$. (Here, $1 \leq a \leq j-1$, $1 \leq b \leq k-1$.) Further, M has the identity element $1 \in \mathsf{G}$ as entries in the first row and first column.

$$M = \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & (d^2c)^t & (d^3c)^t & \cdots & (d^kc)^t \\ 1 & (d^3c)^{2t} & (d^4c)^{2t} & \cdots & (d^{(k+1)}c)^{2t} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ 1 & (d^jc)^{(j-1)t} & (d^{j+1}c)^{(j-1)t} & \cdots & (d^{(j+k-2)}c)^{(j-1)t} \end{bmatrix}.$$

In M, the exponents of the c's are modulo p and the exponents of the d's are modulo q. The parity-check matrix of the bipartite derived graph corresponding to M is given by

$$H = \begin{bmatrix} I & I & I & \cdots & I \\ I & \Pi_{(d^2c)^t} & \Pi_{(d^3c)^t} & \cdots & \Pi_{(d^kc)^t} \\ I & \Pi_{(d^3c)^{2t}} & \Pi_{(d^4c)^{2t}} & \cdots & \Pi_{(d^{(k+1)}c)^{2t}} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ 1 & \Pi_{(d^jc)^{(j-1)t}} & \Pi_{(d^{j+1}c)^{(j-1)t}} & \cdots & \Pi_{(d^{(j+k-2)}c)^{(j-1)t}} \end{bmatrix},$$

where I represents the $m \times m$ identity matrix and Π_σ represents an $m \times m$ permutation matrix for the permutation $\sigma \in \mathsf{P}$. In particular, the matrix Π_σ has a one in the $\sigma(i)$th row and ith column, for $i = 1, 2, \ldots, m$ and 0's elsewhere.

Using the relation that $dc = c^s d$ from the structure of P, we can show that the permutations in M generate P. Any permutation in P can be expressed as $c^a d^b$ for some integers a, b. With some manipulation, it can be seen that an element from the subgroup generated by c appears as a product of some permutations in M. Thus, the group generated by the elements appearing in M is isomorphic to P and from the discussion in Section 3, the derived graph is connected.

C. Optimized General Construction

An optimal code construction would search over all permutations in P to find the voltage assignments for the edges of the co-tree that satisfy the guidelines and result in a derived graph with the best performance. This is only feasible when the group size is not too large (i.e., for designing short to moderate length codes) and in general, this search over the permutations in P is more efficient than searching over the permutations in S_m to optimize randomly designed protograph codes, especially when the guidelines are incorporated. Due to the careful structure of P, even a randomly chosen set of permutations from P that satisfy the above guidelines can result in a derived group with as good a performance as that of a randomly designed protograph code. Hence, a small enough search to choose the permutations more judiciously from P can yield a derived graph with improved performance.

V. Girth and Minimum Distance

Let N_{pq} be the nonabelian group descibed in Section 4, and let $p > q \geq 5$.

Theorem 5.1: Let H_b be a 2×3 base array with the identity entry in the first row and the first column, and having non-identity permutations c and d from the group N_{pq} in the remaining entries. Then the Tanner graph $\hat{\mathcal{G}}$ of the LDPC code $\mathcal{C}$ (having blocklength $N = 3q$ and dimension $K \geq q+1$) that corresponds to the permutation derived graph has girth $g = 4q$, and the LDPC code $\mathcal{C}$ has a minimum distance $d_{\min} = 2q$.

Proof: The voltage graph whose incidence matrix is given by H_b is $K_{2,3}$. The smallest closed walk in this base graph has length four and its net-voltage is a non-identity element in N_{pq}. Since each non-identity element in N_{pq} is a permutation whose cycle representation has cycle lengths equal to or larger than q, we have by Theorem 3.1 that the closed walk of length four gives rise to cycles of length $4q$ or larger in the derived graph $\hat{\mathcal{G}}$.

Let W be a closed walk of length $2k$ in the base graph, for some integer $2 \leq k \leq q-1$. The net voltage on W is a product of $2k$ permutations $\pi_1, \pi_2, \ldots, \pi_{2k}$, where the number of non-identity permutations is between 1 and $q-1$. The non-identity permutations that can appear on this closed walk are c, c^{-1}, d, and d^{-1} with orders p, p, q, q, respectively (i.e., each is at least q). Hence, the net voltage on W has a cycle decomposition consisting of either p cycles or q cycles, and by Theorem 3.1, W lifts to cycles of length at least $2kq$ in $\hat{\mathcal{G}}$.

Now consider a walk W' of length $2(q+t)$ in the base graph, for $0 \leq t \leq q-1$. The net voltage on W' is a product of $2(q+t)$ permutations. The maximum number of non-identity permutations on such a walk is $q+t$. The non-identity permutations in H_b are c (and c^{-1}) and d (and d^{-1}), and each contributes at most $(q+t)/2$ (i.e. less than q) times to the net voltage. Thus, the net voltage on W' is a non-identity permutation with order at least q. By Theorem 3.1 a cycle of length at least $2(q+t)q$ is obtained in the lifted graph $\hat{\mathcal{G}}$.

Finally, consider a walk of length $4q$ in the base graph that traverses the edge with voltage d a total of q times and edges assigned the identity voltage on the remaining $3q$ steps. Such a walk has net voltage equal to the identity since d has order q. Thus, by Theorem 3.1 a cycle of length $4q$ results in the lifted graph $\hat{\mathcal{G}}$. Hence, the girth of $\hat{\mathcal{G}}$ is $4q$.

Further, since all variable nodes in the graph have degree two, the lift $\hat{\mathcal{G}}$ corresponds to a cycle code [21], yielding a minimum distance equal to $2q$ (i.e, girth/2) for the code $\mathcal{C}$. □

Theorem 5.1 shows that a voltage assignment on $K_{2,3}$ can yield a lift and corresponding LDPC code whose girth and minimum distance grow with q. On the other hand, if the voltage group is abelian, the girth of the resulting permutation derived graph cannot exceed 12 and the minimum distance cannot exceed 6 [6]. Thus, the use of appropriately assigned non-commuting voltages can help surpass the girth and minimum distance limitations of abelian voltages in the design of algebraic protograph codes.

Theorem 5.2: Let $\mathsf{G} \cong N_{pq}$. Let H_b be a $j \times k$ base array with $k > j \geq 2$, the identity entry in the first row and column, and having non-identity pairwise non-commuting permutations from N_{pq} in the remaining entries. Then the Tanner graph $\hat{\mathcal{G}}$ of the LDPC code $\mathcal{C}$ (having block length $N = kq$ and dimension $K \geq (k-j)q + (j-1)$) that corresponds to the permutation derived graph has girth $g \geq 6$, and the LDPC code $\mathcal{C}$ has a minimum distance $d_{\min}(\mathcal{C}) \geq j+1$.

Proof: Consider the voltage graph with incidence matrix H_b. This base graph is bipartite with j check nodes and k variable nodes, and its smallest closed walk has length four. Since no two non-identity permutations assigned to the edges of $\mathcal{G}$ are the same, any closed walk of length 4 has a net voltage that is a non-identity permutation in N_{pq} whose order is at least q. By Theorem 3.1, this 4-cycle lifts to a cycle of length at least $4q$ in $\hat{\mathcal{G}}$. Thus the smallest cycle in $\hat{\mathcal{G}}$ must be greater than four. Since $\hat{\mathcal{G}}$ is bipartite, the girth of $\hat{\mathcal{G}}$ is at least 6. This, along with the tree-bound on the minimum distance derived in [28] shows that the minimum distance of the LDPC code $\mathcal{C}$ represented by the Tanner graph $\hat{\mathcal{G}}$ is $d_{\min}(\mathcal{C}) \geq j+1$. □

We note that typically the girth and the minimum distance are significantly larger than the bound in Theorem 5.2. The actual girth and minimum distance can be further improved. For example, one can incorporate a simple optimization criterion that ensures that the girth in the lifted graph corresponding to any $K_{2,3}$ subgraph of the base graph is as large as possible. This optimization can be extended to larger subgraphs in the base graph such as $K_{2,t}$, for $t > 3$, or $K_{x,y}$, for $x > 2, y > 3$ to further optimize the girth and distance. As nonabelian groups with larger p, q are chosen, the minimum distance of the proposed codes can exceed the minimum distances of protograph LDPC codes designed using abelian groups. Moreover, using the results in [29], the lower bounds in Theorems 5.1 and 5.2 also lower bound the minimum stopping set size and pseudocodeword weight of the codes.

VI. Simulation Results

In this section, we show the performance of our codes proposed in the construction example of Section IV.B (labeled in the figures as Consturction 1) using the groups in Example 4.2, and the performance of voltage codes where voltages were chosen semi-randomly from these groups (labeled in the figures as Construction 2). We compare the performance of our proposed codes in each figure to A) random regular LDPC codes, B) random protograph codes (i.e the permutations are chosen randomly from S_m where $m = |P|$), C) a code of comparable parameters deisigned using the algebraic techniques proposed in [6], [7] (TSF-construction) and [8], [9] (array construction), and D) progressive edge growth (PEG)[7] based LDPC codes [30]. The codes in [6], [7] and [8], [9] use permutations corresponding to shifted identity permutation matrices (i.e. generate an abelian group). Performances are compared on the binary input additive white Gaussian noise channel (BIAWGNC) under sum-product decoding. The maximum number of iterations is limited to 50 in all of our simulations.

Figure 3 shows the performance of $(2, 3)$-regular LDPC codes over the BIAWGNC under sum-product decoding. The figure shows the bit-error-rate (BER) performance as a function of the channel signal to noise ratio (SNR) of our voltage graph-based LDPC code, where the voltage graph is $K_{2,3}$, and the corresponding derived graph is obtained by assigning voltages to $K_{2,3}$ to $\mathsf{P} \cong N_{55}$ as described in Section 4. The block length of the resulting codes is 165 and the code rate is approximately 0.33. Also shown are the performances of $(2, 3)$ LDPC codes having similar block lengths and code rates from the other constructions mentioned above. Our optimized version substantially outperforms these other constructions. The proposed codes have a gain of approximately 1 dB at a bit error rate (BER) of 10^{-4} compared to a randomly designed graph, and a gain of more than 1 dB compared to the array/TSF-type construction and random protograph code.

Figure 4 shows the analogous performance of Figure 3 for the group N_{203} (as described in Example 4.3). The resulting codes have a block length of 609 and a code rate of approximately 0.33. Again, our codes perform significantly better, showing a gain of more than 1.5 dB at a BER of 10^{-4} in comparison to the array/TSF-type and random constructions. The proposed codes perform as well as the PEG LDPC codes at this block length and code rate.

[7]The PEG based LDPC codes were designed using the online software available at David MacKay's website: $http://www.inference.phy.cam.ac.uk/mackay/PEG\_ECC.html$

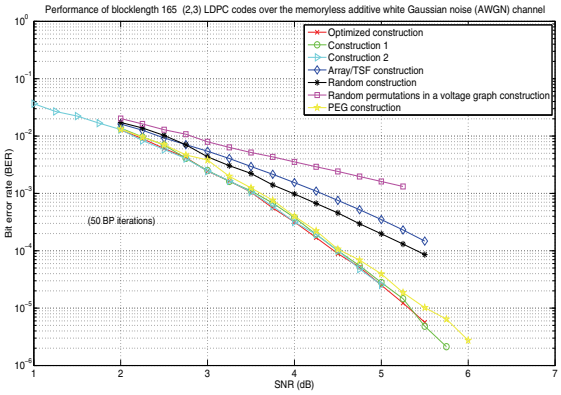

Fig. 3. Performance of block length 165, code rate 0.33, (2,3) LDPC codes on the binary-input additive white Gaussian noise channel under sum-product decoding.

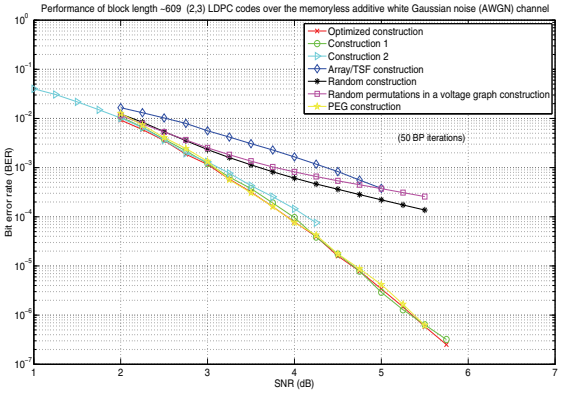

Fig. 4. Performance of block length 609, code rate 0.33, (2,3) LDPC codes on the binary-input additive white Gaussian noise channel under sum-product decoding.

Figure 5 shows the performance of $(3, 5)$-regular LDPC codes over the BIAWGNC under sum-product decoding. The figure shows the performance of our proposed codes using voltages from N_{55} applied to the edges of the base graph $K_{3,5}$. All of the codes in Figure 5 have block length 275 and code rate approximately 0.4. Construction 1 and the optimized version perform comparably to the array/TSF-type construction and the PEG construction for these chosen parameters. However, Construction 2 performs slightly worse. Overall, the algebraic constructions using commutative or non-commutative voltages perform significantly better than the random constructions by at least 0.5 dB at a BER of 10^{-5}.

Figure 6 shows the analogous performance of Figure 5 for the group N_{203}. All of the codes shown have a block length of 1015 and a code rate of approximately 0.4 except the TSF

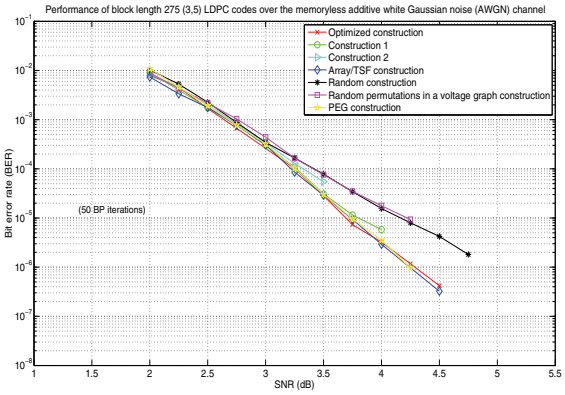

Fig. 5. Performance of block length 275, code rate 0.4, (3,5) LDPC codes on the binary-input additive white Gaussian noise channel under sum-product decoding.

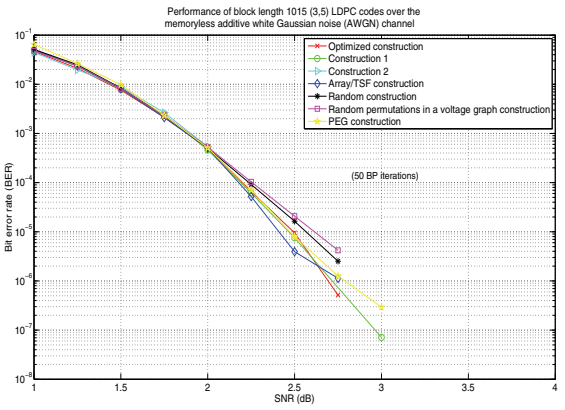

Fig. 6. Performance of block length 1015, code rate 0.4, (3,5) LDPC codes on the binary-input additive white Gaussian noise channel under sum-product decoding.

code, which has a block length of 1055. The results show that our Construction 1 and the optimized version perform comparably to an array/TSF-type construction in the waterfall region. However, the TSF-type construction has a distance upper bounded by $(3+1)! = 24$, whereas the proposed codes do not have such a limitation. The proposed codes (optimized construction, Construction 1) also perform better than the PEG construction at high SNRs due to a better minimum distance. Construction 2 does not perform as well as the rest for the chosen set of parameters. Overall, all of the algebraic constructions perform significantly better than the random constructions by at least 0.25 dB.

We have shown that our codes perform comparably to other algebraically designed codes and can outperform random constructions. Thus, using non-commuting permutations in the design has the potential to provide significant performance improvement over commuting permutations. We believe that when the general relationship between minimum distance and voltage assignments is better understood, it will be possible to describe more effective voltages in later constructions.

VII. Implementation

Due to their inherent algebraic structure, the codes proposed in Section 4 have a succinct description that make them attractive candidates for implementation in practical applications. In this section we address the complexity and hardware implementation aspects of these codes. One important observation is that the proposed codes belong to the class of *matched-lifted LDPC* codes presented in [31]. While the discussion in [31] is mostly general, all of the examples given in [31] assume the (abelian) group of shifted identity matrices in designing the protograph LDPC codes. We can apply similar techniques from [31] to encode and decode the codes proposed in this paper.

A. Alternate representation

Let $\mathsf{G} = \{\pi_1, \pi_2, \ldots, \pi_m\}$ be a permutation voltage group of order m. The elements π_i for $i = 1, 2, \ldots, m$ correspond to permutation matrices P^i, where $P^i_{\pi_i(\ell),\ell} = 1$, $\ell = 1, 2, \ldots, m$. The group operation $\pi_i \cdot \pi_j = \pi_k$ is equivalent to the product of permutation matrices $P^i P^j = P^k$. As in [31], a group-ring $\mathbb{F}_2[\mathsf{G}]$ can be defined having elements that can be represented as binary vectors of length m. In particular, a binary vector $u = (u_1, u_2, \ldots, u_m)$ is identified with the sum $\sum_{i=1}^m u_i \pi_i$. Adding two length m binary vectors u and v in this group-ring is defined by the componentwise addition of u and v over $\mathbb{F}_2$. Multiplying two vectors $u \cdot v$ in $\mathbb{F}_2[\mathsf{G}]$ is defined as follows, where all summations shown are over $\mathbb{F}_2$.

$$(\sum_{i=1}^m u_i \pi_i)(\sum_{j=1}^m v_j \pi_j) = \sum_{i=1}^m \sum_{j=1}^n u_i v_j \pi_i \pi_j$$

$$= \sum_{i=1}^k (\sum_{(i,j):\pi_i \pi_j = \pi_k} u_i v_j)\pi_k = \sum_{k=1}^m (\sum_{(i,j):\pi_i(j)=k} u_i v_j)\pi_k$$

$$= \sum_{k=1}^m (\sum_{i=1}^m u_i v_{\pi_i^{-1}(k)})\pi_k.$$

As each π_i corresponds to the $m \times m$ permutation matrix P^i, the sum $\sum_{i=1}^m u_i \pi_i$ can be interpreted as the matrix sum $M(u) = \sum_{i=1}^m u_i P^i$. We show that $M(u)$ is a matrix over $\mathbb{F}_2$ not only when G is the group of shifted identity permutations but also for more general permutation groups such as those used in this paper.

Lemma 7.1: The map $M(\cdot)$ defined above for the non-abelian permutation group $\mathsf{G} \cong N_{pq}$, ensures that for any length m binary vector u, $M(u)$ is a matrix with only binary entries.

Proof: $M(u)$ is not binary if and only if there exist two permutations in G, call them π_i and π_j for some $i \neq j$, $i, j \in \{1, \ldots, m\}$, where $\pi_i(k) = \pi_j(k)$ for some $k \in \{1, 2, \ldots, m\}$. This is equivalent to having $\pi_j^{-1} \cdot \pi_i(k) = k$. However, $\pi_j^{-1} \cdot \pi_i \in \mathsf{G}$, and the only permutation in G that contains a fixed point in its cycle representation is the identity permutation. This yields that $i = j$, a contradiction. Thus, $M(u)$ is always a binary matrix. □.

While the above result is not true for all permutation groups, it holds for the permutation groups used in this paper. Furthermore, this sum $\sum_{i=1}^m u_i P^i$ uniquely determines u. The length m basis vector $e_i = (0, 0, \ldots, 0, 1, 0, \ldots, 0)$, where the

ith entry is a 1 and the remaining entries are 0 corresponds to permutation matrix P^i under the map $M(\cdot)$ defined above. Without loss of generality, we can assume the vector e_1 denotes the identity permutation in G and corresponds to the identity matrix P^1 of size m.

The constructions in Section 4 start with a bipartite base graph $\mathcal{G}$ that yields the base parity-check matrix H_b of size $j \times k$. The edges of $\mathcal{G}$ are assigned permutation voltages from the permutation voltage group $\mathsf{G} \cong N_{pq}$. These edge assignments can be represented as entries of H_b. Now if we replace each entry in H_b with a binary vector of length m that corresponds to the permutation on that edge in $\mathcal{G}$, we get a $j \times km$ binary matrix. As in [31], applying the map $M(\cdot)$ to each component vector of length m gives a $jm \times km$ parity-check matrix of the lifted code. Further, the resulting binary LDPC code is the set of solutions $\mathbf{x}$ to the equation $M(H_b)\mathbf{x}^T = \mathbf{0}^T$. If we write $\mathbf{x} = (x^1, \ldots, x^n)$, where each x^i is a binary vector of length m, then the equation $M(H_b)\mathbf{x}^T = \mathbf{0}^T$ is also equivalent to $H_b\mathbf{x}^T = \mathbf{0}^T$, where now each x^i is interpreted as an element of $\mathbb{F}_2[\mathsf{G}]$. Thus, we can relate the binary LDPC code associated with the matrix $M(H_b)$ with the LDPC code over the group ring $\mathbb{F}_2[\mathsf{G}]$ associated with the matrix H_b.

We note here that not all random protograph codes that use random permutations satisfy the property of Lemma 7.1, and therefore, not all protograph LDPC codes can use the $\mathbb{F}_2[G]$ group algebra framework shown above.

B. Encoding

As we have shown a similar group-theoretic framework as in [31] in relating the lifted LDPC binary matrix for our proposed codes to the base LDPC matrix over $\mathbb{F}_2[\mathsf{G}]$, we can follow the procedure from [31] to design an efficient encoder for these codes. We will highlight the main points of this procedure and refer the reader to [31] for more details. The encoding procedure uses the low-complexity encoding technique presented in [32]. However, instead of applying this technique on the parity-check matrix of the larger lifted graph $M(H_b)$, we apply this technique to the parity-check matrix of the smaller base graph H_b, where the entries are in $\mathbb{F}_2[\mathsf{G}]$. By row and column operations in the group-ring $\mathbb{F}_2[\mathsf{G}]$ and row-column swaps, H_b is transformed to a matrix of the form $H_b'' = \left[\begin{array}{c|c|c} A & B & T \\ \hline -ET^{-1}A + C & -ET^{-1}B + D & 0 \end{array}\right]$, where T is a lower triangular matrix and the entries of A, B, T, C, D, E are all matrices with entries in $\mathbb{F}_2[\mathsf{G}]$.

Let $\phi = -ET^{-1}B + D$ and let the columns in H_b'' spanning the matrix B correspond to the parity bits p_1, the columns spanning matrix T correspond to the parity bits p_2, and the columns spanning matrix A correspond to the information bits s. Then, given the information sequence s, we solve for the parity bit sequences p_1 and p_2 from the following relations: $As + Bp_1 + Tp_2 = 0, \quad (-ET^{-1}A + C)s + \phi p_1 = 0.$

All of the above operations are carried out in $\mathbb{F}_2[\mathsf{G}]$ arithmetic since the proposed construction can be viewed as a code over $\mathbb{F}_2[\mathsf{G}]$. While the encoding complexity, in terms of the number of arithmetic operations, may still be $O(N + g^2)$, where g is the size of the ϕ matrix and N is the block length of the code, the hardware for the encoder may be implemented efficiently by taking advantage of the group-theoretic framework described above.

C. Decoding

The decoding techniques for matched lifted LDPC codes may be applied to the codes proposed in this paper and we refer the reader to [31] for more details.

VIII. Conclusions

We presented a construction methodology for codes based on permutation voltage graphs. These constructions use non-commuting permutation matrices and yield families of algebraic protograph codes. The construction specifies a permutation assignment from a base graph to a nonabelian group, and is designed to ensure that the resulting derived graph is connected and has no short cycles. Our method may be applied to any base graph, such as those with optimal degree distributions. The performance of the proposed codes is shown to be better than that of random constructions and comparable to other good algebraic constructions. The inherent algebraic structure in the design makes the codes well suited for practical implementation, as demonstrated by the simple encoding and decoding techniques presented. We have also discussed how the girth and the minimum distance for these codes can be strengthened further by optimizing the voltage assignments in specially chosen subgraphs of the base graph. We conclude that codes based on algebraic lifts have the potential to outperform random codes, as well as those based on random lifts.

Acknowledgment

The author would like to thank the three anonymous reviewers for their detailed comments that helped improve the quality of this paper.

References

[1] J. Thorpe, "LDPC codes constructed from protographs," *IPN Progress Report*, pp. 42–154, JPL, Aug. 2003.
[2] J. Thorpe, K. Andrews, and S. Dolinar, "Methodologies for designing LDPC codes using protographs and circulants," in *Proc. 2004 IEEE Intl. Symp. Inf. Theory*, p. 236.
[3] D. Divsalar, S. Dolinar, and C. Jones, "Construction of protograph LDPC codes with minimum distance linearly growing with block size," in *Proc. 2005 IEEE Globecom*, pp. 1152–1156.
[4] J. L. Gross and T. W. Tucker, *Topological Graph Theory*. Wiley, 1987.
[5] C. A. Kelley and J. L. Walker, "LDPC codes from voltage graphs," in *Proc. 2008 Intl. Symp. Inf. Theory*.
[6] R. M. Tanner, D. Sridhara, and T. E. Fuja, "A class of group-structured LDPC codes," in *Proc. 2001 Intl. Symp. Commun. Theory Appl.*, pp. 365–370.
[7] D. Sridhara, T. E. Fuja, and R. M. Tanner, "Low density parity check codes from permutation matrices," in *2001 Conf. Inf. Sci. Syst.*
[8] J. L. Fan, "Array codes as low-density parity-check codes," in *Proc. 2000 Intl. Symp. Turbo Codes Appl.*, pp. 543–546.
[9] O. Milenkovic, N. Kashyap, and D. Leyba, "Shortened array codes of large girth," *IEEE Trans. Inf. Theory*, vol. 5, no. 8, pp. 3707–3722, Aug. 2006.
[10] S. Song, L. Lan, S. Lin, and K. A-Ghaffar, "Construction of quasi-cyclic LDPC codes based on the primitive elements of finite fields," in *Proc. 2006 Conf. Inf. Syst. Sciences*, pp. 835–838.
[11] R. M. Tanner, "Quasi-cyclic repeat accumulate codes," in *Proc. 1999 Allerton Conf. Commun., Control Comput.*, pp. 249–259.
[12] C. A. Kelley, "On codes designed via algebraic lifts of graphs," in *Proc. 2008 Allerton Conf. Commun., Control, Comput.*

[13] N. Linial and E. Rozenman, "Random lifts of graphs: perfect matchings," *Combinatorica*, vol. 25, pp. 407–424, 2005.

[14] A. Amit and N. Linial, "Random lifts of graphs II: edge expansion," *Combinatorics, Probability Comput.*, vol. 15, pp. 317–332, 2006.

[15] A. Amit, N. Linial, and J. Matousek, "Random lifts of graphs: independence and chromatic number," *Random Structures Alg.*, vol. 20, no. 1 , pp. 1–22, Jan. 2002.

[16] G. Exoo, "Voltage graphs, group presentations, and cages," *Electron. J. Combinatorics*, vol. 11, no. 1, 2004.

[17] L. Brankovic, M. Miller, J. Plesnik, J. Ryan, and J. Siran, "Large graphs with small degree and diameter: a voltage assignment approach," *J. Combinatorial Math. Combinatorial Computing*, vol. 24, pp. 161–176, 1997.

[18] L. Brankovic, M. Miller, J. Plesnik, J. Ryan, and J. Siran, "A note on constructing large Cayley graphs of given degree and diameter by voltage assignments," *Electron. J. Combinatorics*, vol. 5, no. 1, R9, 1998.

[19] D. Archdeacon, J.-H. Kwak, J. Lee, and Y. Sohn, "Bipartite covering graphs," *Discrete Mathematics*, vol. 214, pp. 51–63, 2000.

[20] M. P. C. Fossorier, "Quasi-cyclic low-density parity-check codes from circulant permutation matrices," *IEEE Trans. Inf. Theory*, vol. 50, no. 8, pp. 1788–1793, 2004.

[21] D. J. C. MacKay and M. C. Davey, "Evaluation of Gallager codes for short block length and high rate applications," in *Codes, Systems, and Graphical Models*, B. Marcus and J. Rosenthal, editors., vol. 123 of *IMA Volumes in Mathematics and its Appl.*, pp. 113–130. Springer, 2000.

[22] R. Smarandache and P. O. Vontobel, "On regular quasi-cyclic LDPC codes from binomials," in *Proc. 2004 IEEE Intl. Symp. Inf. Theory*.

[23] O. Y. Takeshita, "A new construction for LDPC codes using permutation polynomials over integer rings," arXiv:cs/0506091v1, 2005.

[24] K. Yang and T. Helleseth, "On the minimum distance of array codes as LDPC codes," *IEEE Trans. Inf. Theory*, vol. 49, no. 12, Dec. 2003.

[25] R. Smarandache and P. O. Vontobel, "Quasi-cyclic LDPC codes: influence of proto- and Tanner-graph structure on minimum Hamming distance upper bounds," submitted to *IEEE Trans. Inf. Theory*, Jan. 2009.

[26] B. K. Butler and P. H. and Siegel, "On distance properties of quasi-cyclic protograph-based LDPC codes," in *Proc. 2010 IEEE Intl. Symp. Inf. Theory*, pp. 809–813.

[27] J. Chen, R. M. Tanner, J. Zhang, and M. P. C. Fossorier, "Construction of irregular LDPC codes by quasi-cyclic extension," *IEEE Trans. Inf. Theory*, vol. 53, no. 4, pp. 1479–1483, Apr. 2007.

[28] R. M. Tanner, "A recursive approach to low complexity codes," *IEEE Trans. Inf. Theory*, vol. IT-27, pp. 533–547, Sept. 1981.

[29] C. A. Kelley, D. Sridhara, and J. Rosenthal, "Tree-based construction of LDPC codes having good pseudocodeword weights," *IEEE Trans. Inf. Theory*, vol. 53, no. 4, pp. 1460–1478, Apr. 2007.

[30] X. Y. Hu, E. Eleftheriou, and D. M. Arnold, "Progressive edge-growth Tanner graphs," in *Proc. 2001 IEEE GLOBECOM*, pp. 995–1001.

[31] T. J. Richardson and R. L. Urbanke, *Modern Coding Theory*. Cambridge University Press, 2008.

[32] T. J. Richardson and R. L. Urbanke, "Efficient encoding of low-density parity-check codes," *IEEE Trans. Inf. Theory*, vol. 47, no. 2, pp. 638–656, Feb. 2001.

[33] C. A. Kelley and J. Kliewer, "Algebraic constructions of graph-based nested codes from protographs," in *Proc. 2010 IEEE Intl. Symp. Inf. Theory*.

[34] T. W. Hungerford, *Algebra*, Grad. Texts in Mathematics, vol. 73. Springer-Verlag, 1974.

[35] P. J. Cameron, *Permutation Groups*, London Mathematical Society Student Texts, vol. 45. Cambridge University Press, 1999.

[36] M. M. Mansour and N. R. Shanbhag, "High-throughput LDPC decoders," *IEEE Trans. Very Large Scale Integration (VLSI) Syst.*, vol. 11, no. 6, pp. 976–996, Dec. 2003.

Christine A. Kelley is an Assistant Professor in the Department of Mathematics at the University of Nebraska-Lincoln. She received her Ph.D. in math from the University of Notre Dame in 2006. Before coming to Nebraska, she was a Postdoctoral Fellow at the Fields Institute in Toronto, and in the Department of Mathematics at The Ohio State University. Dr. Kelley's research is in coding theory and applied discrete mathematics. One emphasis is on graph-based codes and algorithms, and the design and analysis of such codes using algebraic methods. Another research interest is in applying algebraic and combinatorial methods to coding for flash memory storage. Her research is currently being supported by an NSA Young Investigator grant (Spring 2011-Spring 2013) and has been supported by an NSF EPSCoR First Award (2009-2010).

In Spring 2010, Dr. Kelley received the University of Nebraska's Harold and Esther Edgerton Junior Faculty Award for "creative research, extraordinary teaching abilities, and academic promise," and she was the Harold and Esther Edgerton Assistant Professor from 2010-2012.