On Codes Designed via Algebraic Lifts of Graphs

Christine A. Kelley
Department of Mathematics
University of Nebraska-Lincoln
Lincoln, NE 68588, USA.
Email: ckelley2@math.unl.edu

***Abstract*— Over the past few years several constructions of protograph codes have been proposed that are based on random lifts of suitably chosen base graphs. More recently, an algebraic analog of this approach was introduced using the theory of voltage graphs. The strength of the voltage graph framework is the ability to analyze the resulting derived graph algebraically, even when the voltages themselves are assigned randomly. Moreover, the theory of voltage graphs provides insight to designing lifts of graphs with particular properties. In this paper we illustrate how the properties of the derived graphs and the corresponding codes relate to the voltage assignments. In particular, we present a construction of LDPC codes by giving an algebraic method of choosing the permutation voltages and illustrate the usefulness of the proposed technique via simulation results.**

I. Introduction

Codes on graphs, along with message passing decoders, have been shown to achieve near-capacity performance on several communication channels and have replaced classical codes, such as Reed-Solomon codes, in many practical applications. While these codes and decoders have been shown to perform remarkably well by means of simulation, theoretical results that explain why or how they are good remain scarce. 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. Several researchers have proposed structure-based constructions of these codes and each of these constructions has aimed to optimize one or more properties in the resulting graph that intuitively improve the resulting code's performance, such as girth, expansion, diameter, stopping sets, or pseudocodewords. One area of recent interest is *protograph LDPC codes*, which are codes based on graphs obtained by taking random lifts of a suitably chosen base graph, or *protograph* [1], [2], [3]. Among other advantages of this modern approach, these codes can be represented efficiently and perform well compared to randomly designed codes with comparable parameters.

In this paper we consider codes designed from a voltage graph viewpoint wherein specific lifts of graphs are determined via "voltage assignments", i.e., assignments of elements of a so-called *voltage group* to the edges of a base graph, thus making the lifting entirely algebraic. This concept, originally coined in topological graph theory [4], 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, shortened array codes, quasi-cyclic repeat accumulate codes, and others fall into this category [6], [7], [8], [9], [10], [11]. This algebraic characterization of lifts is a powerful tool for analyzing several graph properties of the resulting lifts using the properties of the base graph. In this paper, we examine this relationship and provide guidelines for choosing voltage assignments to obtain good codes. We outline an algebraic technique of specifing the voltage assignments by restricting the voltage assignments to a permutation group designed in a special way. Our construction method, which may be applied to any base graph or protograph, results in a family of codes having good properties.

The paper is organized as follows. We introduce some preliminary definitions and notation in Section II. In particular, we review the terminology of ordinary and permutation voltage graphs. In Section III, we explain the relationship between the voltages in the base graph and the cycle structure and connectivity of the derived graph. In Section IV, we present a method for assigning permutation voltages to the edges of a base graph, which gives an explicit construction of a family of codes. The choices of permutations yields codes with improved girth and distance when compared to protograph codes of the same parameters that use random permutations, and codes where the voltages are cyclic shifts. Simulation results are presented in Section V to illustrate the usefulness of the proposed approach. Finally, a discussion of ongoing work is outlined in Section VI.

II. Voltage graphs and LDPC codes

The graph representation of a code plays a fundamental role in determining the code's performance under iterative decoding algorithms. For instance, a binary low-density parity-check (LDPC) code is defined by a sparse parity-check matrix H or, equivalently, by the incidence graph of H, called the Tanner graph, which is bipartite. The left and right vertices are called variable nodes and check nodes, respectively. The set of codewords may be characterized as the set of all binary assignments to the

variable nodes such that at each check node, the modulo two sum of the variable node assignments connected to that check node is zero.

Several researchers have looked at constructing families of LDPC codes by taking random lifts of a specially chosen base graph, or "protograph", yielding the so-called "protograph codes" (See for example, [1], [2], [3]). The idea exploited in these constructions is that the properties of the base graph may reveal and influence the properties of the graph lift, and therefore also of the resulting codes. Indeed, random lifts of graphs have been heavily studied (see, for example, [12], [13], [14]). While these codes have exhibited good performance, we believe that codes based on algebraically-designed lifts have the potential to outperform these random constructions. The properties of an algebraically-designed lift are determined by the properties of the base graph so that, once the relationship of the parameters like minimum distance, stopping sets, and pseudodistance between the two graphs is fully understood, preserving these properties can be incorporated into the design. Furthermore, for a protograph code obtained using random permutations, the voltage graph framework provides a tool to analyze the resulting code properties by looking at its specific voltage assignments. Ultimately, our goal is to obtain an algebraic construction that provides structure suitable for coding, while also retaining some of the desirable random-like characteristics of random lifts.

An algebraic construction of specific covering spaces for graphs was introduced by Gross and Tucker in the 1970s [4]. Given a graph $\mathcal{G} = (V_\mathcal{G}, E_\mathcal{G})$ where each edge in $\mathcal{G}$ has a positive and negative orientation, a function α, called an *ordinary voltage assignment*, maps the positively oriented edges to elements from a chosen finite group G, called the *voltage group*. The negative orientation of each edge is assigned a voltage that is the inverse element of the voltage assigned to its positive orientation. The base graph $\mathcal{G}$, together with the function α, is called an *ordinary voltage graph*. The values of α on the edges are referred to as *voltages*. A new graph $\mathcal{G}^\alpha$, called the *(right) derived graph*, is a degree $|G|$ lift of $\mathcal{G}$ and has vertex set $V_\mathcal{G} \times G$ and edge set $E_\mathcal{G} \times G$, where, if (u, v) is a positively oriented edge in $\mathcal{G}$ with voltage $h \in G$, then there is an edge from (u, g) to (v, gh) in $\mathcal{G}^\alpha$ for each $g \in G$. Figure 1, which is taken from [4], shows an ordinary voltage graph $\mathcal{G}$ (also referred to as a "base graph") with voltages assigned to its edges from the additive group of integers modulo 5 (i.e., $G = \mathbb{Z}/5\mathbb{Z}$), and the corresponding right-derived graph obtained from this assignment.

When the voltage group is a permutation group, (i.e. a subgroup of the symmetric group S_n on n elements), the base graph may be interpreted as a *permutation voltage graph* $\mathcal{G}$, which gives an alternative type of derived graph[1].Specifically, $\mathcal{G}^\alpha$ is a *permutation derived graph* with vertex set $V_\mathcal{G} \times \{1, \ldots, n\}$ and edge set $E_\mathcal{G} \times \{1, \ldots, n\}$. If $\pi \in S_n$ is a permutation voltage on the edge $e = (u, v)$ of $\mathcal{G}$, then there is an edge from (u, i) to $(v, \pi(i))$ in $\mathcal{G}^\alpha$ for $i = 1, 2, \ldots, n$. We will represent each vertex (v, i) and each edge (e, i) in the derived graph by v_i and e_i, respectively. The set of vertices $\{v_i | i = 1, 2 \ldots, n\}$ is called the *fiber* over v, or *cloud* of v, in the derived graph, and similarly for edges. The fibers contain precisely those elements in the pre-image of a vertex (or edge) under the *natural projection mapping* $p : \mathcal{G}^\alpha \to \mathcal{G}$. Note that $\mathcal{G}^\alpha$ is a degree n lift of $\mathcal{G}$ rather than a degree $|G|$ lift, as it would be if the assignment was an ordinary voltage assignment. Figure 2, also taken from [4], shows a permutation voltage graph $\mathcal{G}$ with voltages assigned to the edges from S_3, and the corresponding derived graph $\mathcal{G}^\alpha$.

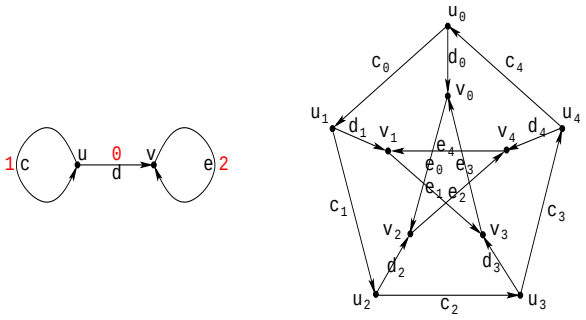


Fig. 1. A voltage graph with voltage group $A = \mathbb{Z}/5\mathbb{Z}$, and its derived graph.

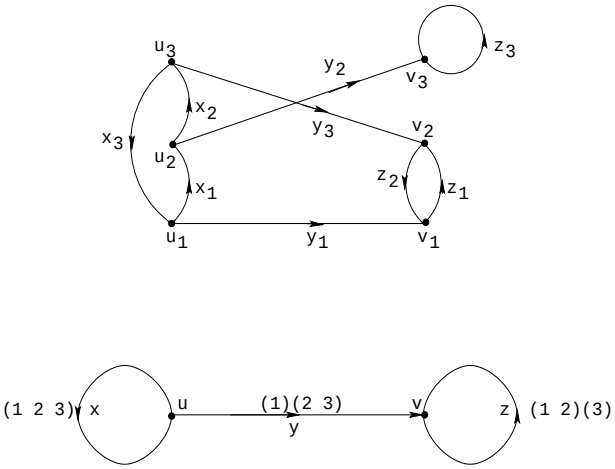


Fig. 2. A permutation voltage graph with voltage group S_3, and its derived graph.

For an edge e, let e^- and e^+ denote the negative and positive orientations, respectively, of e. A *walk* in the ordinary or 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. $W = e_1^{\sigma_1} e_2^{\sigma_2} \ldots e_n^{\sigma_n}$ where each σ_i is $+$ or $-$ and $e_1, \ldots, e_n$ are edges in $\mathcal{G}$. In this setting, the *net voltage* of the walk W is defined as the voltage group product

$$\alpha(e_1^{\sigma_1})\alpha(e_2^{\sigma_2}) \ldots \alpha(e_n^{\sigma_n})$$

[1]In [4], a permutation voltage graph has voltage group S_n, although the voltages assigned need not generate the entire group. In this paper, we design the group that will be generated by the voltages assigned.

of the voltages on the edges of W in the order and direction of the walk.

For example, the walk $W = d^+e^-d^-c^+$ in the voltage graph of Figure 1 has net voltage $0 + (-2) + (-0) + 1 = -1 = 4 \in \mathbb{Z}/5\mathbb{Z}$, and the walk $W = z^+y^-x^+$ in the voltage graph of Figure 2 has net voltage $(12)(3) \times (1)(23) \times (123) = (1)(2)(3)$.

The following theorem is taken from [4].

Theorem 2.1: Let W be a walk in a voltage graph $\mathcal{G}$ with initial vertex v. Then for each vertex v_g in $\mathcal{G}^\alpha$ for $g \in G$ (or $g \in \{1, 2, \ldots, n\}$ for the permutation voltage graph case), there is a unique walk W_g in $\mathcal{G}^\alpha$ that starts at v_g and projects down[2] to W.

A walk is of length n is *closed* if it starts and ends at the same vertex, and *backtrackless* if $e_i \neq e_{i+1}$ for $1 \leq i \leq n-1$. A backtrackless closed walk is said to be *tailless* if $e_n \neq e_1$. An useful consequence of the unique walk theorem is as follows. Assume $W = e_1^{\sigma_1} e_2^{\sigma_2} \cdots e_n^{\sigma_n}$ is closed, backtrackless, and tailless. Then W_g, for any $g \in G$, is a cycle on $\mathcal{G}^\alpha$ if and only if the net voltage of W is the identity of G.

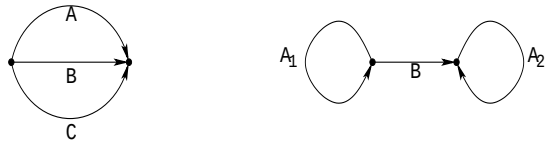
Voltage graphs have been successfully used to obtain many instances of graphs with extremal properties; see e.g. [15], [16].

III. Voltages and code properties

While both ordinary and permutation voltage graphs may be used to represent codes, we will restrict our attention to permutation voltage graphs in this paper. These have the advantage of smaller degree lifts, since the degree of the lift associated with permutations of length n is just n, while the degree of an ordinary voltage graph using voltages from S_n is $n!$. They also provide a natural algebraic analog to random protograph codes that use random permutations. Moreover, we will also focus our investigation on the use of nonabelian groups as voltage groups, since the use of abelian groups has known limitations. After motivating the focus to nonabelian groups, we discuss how to choose voltages to ensure that the derived graph is connected and has a good cycle structure.

A. Abelian voltages

We briefly summarize some known limitations of abelian voltage assignments. First, an (a, b, c)-*theta graph* is a graph consisting of two vertices of degree three that are connected to each other by three disjoint nonempty paths A, B, and C of lengths a, b, and c, respectively. An $(a_1, a_2; b)$-*dumbbell graph* is a connected graph consisting of two disjoint cycles A_1 and A_2 of lengths $a_1 \geq 1$ and $a_2 \geq 1$, that are connected by a path B of length $b \geq 0$. These are shown in Figure 3. In [5], it was shown that if voltages were assigned from an abelian group G to a base graph $\mathcal{G}$, where $\mathcal{G}$ contained either a (a, b, c)-theta graph or a $(a_1, a_2; b)$-dumbbell graph as a subgraph, then a cycle would *inevitably* result in the derived graph (i.e. regardless of how the voltages are assigned to G). The inevitable cycle will have length $2(a + b + c)$ in the first case, and length $2(a_1 + a_2) + 4b$ in the second case. A similar observation for the theta-graph case is shown in [15]. Moreover, in [5] it is also shown that any such "inevitable cycle" in a derived graph specified by an abelian voltage assignment results from a base graph containing one of these two types of subgraphs. This provides a full classification for the cycles that will always occur in the derived graph when abelian voltages are used (regardless of how they are assigned), due to the base graph structure. This suggests that noncommuting voltage assignments have a greater potential in yielding derived graphs with large girth.



Theta graph Dumbbell graph

Fig. 3. A theta graph and a dumbbell graph.

The result on inevitable cycles in [5] provides 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 limitation of girth 12 was originally shown in [7] and later by various groups (see for example, [17]). In addition, many researchers have noted that the use of commuting permutation matrices, such as those arising from shifted identity matrices, leads to restrictions on the minimum distance of the associated code. For example, MacKay and Davey [18] first proved that a code based on a $j \times k$ array of commuting permutation matrices has distance at most $(j+1)!$. Similar results for more generalized arrays are given in [8], [19], [20], [21] and others. While the limitation on distance may be due in part to the limitation on girth, having a large girth is not sufficient for having a large minimum distance. Interestingly, in [22] it is demonstrated that certain subgraphs similar[3] to the ones in Figure 3 have large girth and yet may give rise to small minimum distance. Still, the tree bounds [23] on minimum distance indicate a correlation between girth and distance, and in the case of cycle codes where every variable node has degree two, the distance is exactly half the girth. Thus, the best distance that a 2×3 array code can attain using commuting permutations is 6.

B. Cycle structure

While the use of nonabelian voltages seems a good choice, it is not enough to ensure that the resulting graph has large girth. We now look at how the choice of voltages affects the cycle structure and connectivity

[2]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.

[3]If the graphs in Figure 3 are bipartite and have check nodes in the positions of the vertices of degree three, then these are the graphs identified in paper [22].

of the derived graph. We first briefly review some facts about permutations. Consider the permutation

$$\sigma = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 5 & 1 & 4 & 3 & 7 & 6 & 2 \end{pmatrix}$$

which maps each element in the first row to the element below it on the second row. Any permutation has a unique decomposition into a product of cyclic permutations, each called a *cycle*. For example, σ above may be written in its cycle decomposition as $(1572)(34)(6)$, which consists of one 4-cycle, one 2-cycle, and one 1-cycle (i.e. *fixed point*). 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 decomposition of the permutation. 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. (It should however be clear from the context as to which cycle we are referring to.)

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 derived graph is determined by the cycle structure of the net voltage permutation. This will aid in our determination of suitable voltage assignments.

Theorem 3.1: [4] Let C be a k-cycle in the base graph of a permutation voltage graph with net voltage π, 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 has $c_1 + c_2 + \cdots + c_n$ components, including, for each $j = 1, \ldots, n$, exactly c_j kj-cycles.

It is worth noting that a similar result is also presented in [4] for the case of ordinary voltage graphs, with the important difference that the pre-image of a cycle in the ordinary base graph will be a union of disjoint cycles all having the same length. To continue the example above, 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 the cycle 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. The cycle structure of σ is therefore $(1, 1, 0, 1, 0, 0, 0)$.

Remark 3.2: A result similar to Theorem 3.1 was observed by Fan in [8].

We can extend the previous theorem to closed walks, instead of just cycles, and use the cycle structure of the net voltages on closed walks to reveal the closed walk structure in the derived graph. Since each closed walk in a graph contains a cycle, the length of the closed walks in the base graph also affects the girth of the derived graph.

Theorem 3.3: Let W be a closed walk of length k in the base graph of a permutation voltage graph with net voltage π, and let $(c_1, c_2, \ldots, c_n)$ be the cycle structure of π. Then the pre-image of W in the derived graph has $c_1 + c_2 + \cdots + c_n$ components, including for each $j = 1, \ldots, n$, exactly c_j closed walks of length kj.

The proof of Theorem 3.3 follows the same argument as in [4]. Note that even when W is a closed walk and not necessarily a single cycle, then the pre-image of W may have a component that is a single cycle, which is also a closed walk by definition.

To determine the voltage assignments in our code construction, we will choose permutation voltages that do not have fixed points, and in fact, do not contain cycles of length ≤ 3. This will allow our construction 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. We will also check that these conditions are met for all non-backtracking cycles and closed walks of relatively short length.

C. Connectivity

Last, we want to ensure that our selection of voltages yields a derived graph that is connected. For any assignment α of edges in $\mathcal{G}$ to voltages in a group G, it is possible to find for any spanning tree $\mathcal{T}$ of $\mathcal{G}$, a voltage assignment α' of edges in $\mathcal{G}$ to G, where the edges of $\mathcal{T}$ are assigned the identity permutation under α' and the resulting graphs $\mathcal{G}^\alpha$ and $\mathcal{G}^{\alpha'}$ are isomorphic [4], [24]. We will therefore focus on how to assign voltages to the edges that lie outside of a chosen spanning tree, also called the *co-tree*. With such a voltage assignment, one can consider the group generated by the voltages assigned to the co-tree. This group, G', called a *local voltage group* is a subgroup of the original voltage group. Assume $\mathcal{G}$ is connected. In [24] it is explained that the ordinary derived graph $\mathcal{G}^{\alpha'}$ is connected if and only if $G' = G$. In [4], a similar result is explained for a permutation derived graph, which involves the structure of the orbits when the local group acts on the set $\{1, 2, \ldots . n\}$. In particular, the number of components in the derived graph is equal to the number of orbits under this action. Therefore, we will choose a permutation group whose action yields a single orbit, and ensure that the voltages assigned to the co-tree generate this group. Note that with random permutations, it is possible that the resulting derived graph is disconnected.

D. Guidelines for voltage assignments

In this section, we look at the case where the permutation voltage graph is the complete bipartite graph $K_{2,3}$ (or equivalently, the $(2, 2, 2)$-theta graph), and explain our voltage assignment guidelines (See Figure 4.) While the edges in the base graph may be arbitrarily oriented, our convention is to orient edges *from* variable nodes *to* check nodes. That is, if σ is a permutation voltage on an edge

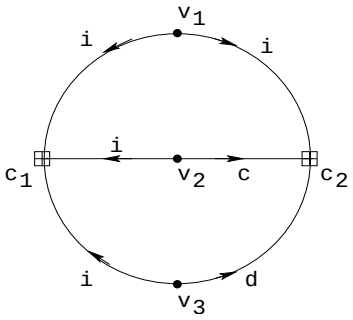


Fig. 4. The incidence graph of a 2×3 array with voltage assignments.

in the base graph, then the corresponding permutation matrix has a 1 in positions (x, y) where $\sigma(y) = x$, and zeros elsewhere. Once the edge directions are established, we can assign permutation voltages from the edges in $K_{2,3}$ to elements in S_m, for m of our choice. We will first choose a spanning tree and assign the identity permutation element to each edge in the tree. Without loss of generality, we choose the spanning tree consisting of the edges in $K_{2,3}$ that correspond to the first row and the first column in the 2 by 3 array of $m \times m$ permutation matrices that give the associated parity-check matrix H. We will therefore only assign nontrivial voltages, c and d, to two of the edges, and these elements will be chosen so that they generate a nonabelian group of order m. The array form of H is given below, where i is used to denote the identity element in the permutation group, $(1)(2)(3)\cdots(m)$,and the corresponding voltage graph is in Figure 4.

$$H = \begin{bmatrix} i & i & i \\ i & c & d \end{bmatrix}$$

For the cycle condition given in Theorem 3.1, we consider the net voltages of all cycles in the base graph. There are six 4-cycles having the following net voltages:

$$\begin{array}{cc} c & c^{-1} \\ cd^{-1} & dc^{-1} \\ d & d^{-1} \end{array}$$

Note that each net voltage in the right column is the inverse of the net voltage in the left column, and therefore corresponds to the same 4-cycle in the base graph. The difference is simply the order in which the edges are traversed. For an assignment of voltages from a nonabelian group to give girth larger than 12 in the derived graph, Theorem 3.1 says that each of these net voltages, when considered as permutations, must have no cycles of length 1, 2 or 3 in its cycle decomposition. For, if π was a net voltage with cycle decomposition containing a 3-cycle, then there would be a cycle of length 12 in the derived graph. Similarly, a 2-cycle or 1-cycle in the cycle decomposition of π would mean there is a cycle of length 8 or 4 in the derived graph, respectively. There are no 6-cycles in the base graph that do not repeat edges, and not enough edges for cycles of size > 6.

To improve the girth we also use Theorem 3.3 to check other closed walks in the base graph when choosing the permutation voltages. For example, we consider the two walks of length six in the graph $K_{2.3}$ with net voltages

$$cd^{-1}c^{-1} \qquad dc^{-1}d^{-1}.$$

Observe that there are no closed walks of length six that start at a check node, and that the other closed walks of length six in the graph have net voltages that already appeared in the list for 4-cycle net voltages. Again, to surpass a girth of 12, Theorem 3.3 indicates that these net voltages must not have 1 or 2 cycles in their cycle decompositions as permutations. Of course, this just gives an upper bound on the girth. Similarly, we can consider the closed walks of length 8 and length 12, and check that none of them contain fixed points. The closed walks of length 8 are given below:

$$\begin{array}{cc} c^2 & (c^{-1})^2 \\ d^2 & (d^{-1})^2 \\ cd & d^{-1}c^{-1} \\ dc & c^{-1}d^{-1} \\ cdc^{-1} & c^{-1}d^{-1}c \\ dcd^{-1} & d^{-1}c^{-1}d \end{array}$$

For cycles and closed walks of longer lengths, and in general, for protographs of any type, similar net voltages and conditions may be enumerated. In summary, to surpass the girth limitations of earlier code constructions, permutation voltages should be chosen from a nonabelian group so that the above lists of net voltages do not have short cycles in their cycle decompositions. Moreover, the edges in the co-tree will be assigned generating voltages to ensure connectivity. In the next section, we give a method for choosing such voltages algebraically.

IV. Construction

In this section we will present a method of assigning permutation voltages to base graphs. We outline the method in the first sub-section and illustrate the method with two examples in the second.

We start with a complete graph on j check nodes and k variable nodes and orient the edges from the variable nodes to the check nodes. The permutation assignments will be to these positively oriented edges (while the negative direction of each edge will be assigned the inverse permutation as a voltage.) The parity check matrix of the resulting LDPC code will be a $j \times k$ array of $m \times m$ permutation matrices that are determined by the method outlined next. The proposed voltage assignment method may be applied to other types of protographs as well in a straightforward manner.

A. Method:

We present a voltage assignment scheme from the edges of the complete bipartite graph $K_{j,k}$ to a nonabelian group G. This scheme may be applied to any chosen base graph, and not just complete bipartite graphs. We list the

general steps of our construction, and then describe each step in greater detail in the following subsections.

1) Choose a nonabelian group G of order m, and label the elements of G from 1 to m.
2) Let G act on itself by left multiplication to obtain an isomorphic group P, where P is a permutation group of order m. That is, P is a subgroup of S_m and is a nonabelian group of order m. P will be the permutation voltage group and has the desirable property that the only element in P with a fixed point is the identity permutation.
3) Choose a spanning tree of the base graph and assign each edge in the tree the identity permutation i. For the remaining edges, choose the nontrivial voltages so that each belongs to a distinct cyclic subgroup of the group P, and, in addition, the generators of the group should be chosen among this set.

We now elaborate on each of these steps. We will use the nonabelian group of order 6 to illustrate these steps, but not for our actual construction since it has elements with 2-cycles and 3-cycles in their cycle decompositions.

1) Choosing a nonabelian group G: We choose $m = pq$ such that p and q are prime, $q < p$, and $q|(p-1)$. We construct the nonabelian group G generated by elements c and d such that the order of c is p, the order of d is q, and $dc = c^s d$, where $s \not\equiv 1 (\text{mod } p)$ and $s^q \equiv 1 (\text{mod } p)$ (See e.g. [25].) We then label the elements of G from 1 to m. Note that nonabelian groups do not exist for every m. In particular, the nonabelian group just described is the only nonabelian group (up to isomorphism) for order $m = pq$ where $q|(p-1)$, and if $q \nmid (p-1)$, then there is no nonabelian group of order pq.

Example: Let $p = 3$ and $q = 2$. Then the nonabelian group G of order 6 is obtained by two generators, c of order 3 and d of order 2, with the relation that $c^2 d = dc$. Then the elements of G are $\{1, c, c^2, d, cd, c^2 d = dc\}$. Order the elements in G as $1 \mapsto 1, c \mapsto 2, c^2 \mapsto 3, d \mapsto 4, cd \mapsto 5, c^2 d \mapsto 6$.

2) Obtaining the permutation group P: A well-known result in algebra is that any finite group is isomorphic to a permutation group, where a permutation group is defined as a subgroup of the symmetric group S_n on n elements. Letting each element of G act on G from the left generates an element of the isomorphic permutation group P of order m which is a subgroup of S_m. In this way, the desired permutation voltage group P may be obtained.

For example, the action of c on G yields the set $\{c \cdot g | g \in G\} = \{c, c^2, 1, cd, c^2 d, d\} = \{2, 3, 1, 5, 6, 4\}$. 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 it can be shown that the permutation corresponding to the action of d is $(14)(26)(35)$, of c^2 is $(132)(465)$, of cd is $(15)(24)(36)$, of $c^2 d$ is $(16)(25)(34)$. Clearly, the action of i on G gives i, and corresponds to the identity permutation $(1)(2)(3)(4)(5)(6)$. Thus, P is the subgroup of S_6 containing these six permutations.

3) Choosing the voltages: Choose the edges in $K_{j,k}$ corresponding to the first row and first column of the $j \times k$ array to be the chosen spanning tree, and assign each of them to the identity permutation in P. This leaves $(j-1)(k-1)$ nontrivial permutations to assign to the remaining edges. Observe that 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 of these subgroups for the remaining edges. Note that for this to be possible, $(j-1)(k-1)$ 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. [26]) ensures that P has just one orbit when acting on $\{1, 2, \ldots, 55\}$. Thus, the nontrivial permutations we choose should generate P to ensure that the condition for connectivity is met. In addition, if one wants to achieve girth larger than 12, then $q > 3$, since if $q = 3$, there will be permutations in P with order 3 which will therefore contain 3-cycles in their cycle decompositions. If a 4-cycle in the base graph contains a 3-cycle in its net voltage decomposition, then the derived graph with have a 12-cycle. Thus, the smallest nonabelian group we can use with these constraints has order $m = 55$.

B. Examples

We construct examples of codes using the nonabelian permutation group of order 55 as the permutation voltage group for a 2×3 array and a 3×5 array. Note that $m = 5 \cdot 11$, thus $p = 11$ and $q = 5$. We start by constructing the group G in step 1, with generators c and d. We enumerate the elements in terms of c and d and label them. Then, the action of c on the group G yields a permutation of the numbered elements in the group. (Since it will be a generator of the isomorphic permutation group, we will also call this permutation c). Similarly we let d act on the group G to give the other generator of the permutation group (a permutation we will also call d). Our labeling gives that the permutations c and d are, in cycle notation,

$$c : (1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11) \cdot$$
$$(12, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25) \cdot$$
$$(13, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35) \cdot$$
$$(14, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45) \cdot$$
$$(15, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55).$$

$$d : (1, 12, 13, 14, 15)(2, 18, 34, 40, 49, 2) \cdot$$
$$(3, 21, 32, 42, 53)(4, 24, 30, 39, 46) \cdot$$
$$(5, 16, 28, 44, 50)(6, 19, 26, 38, 54) \cdot$$
$$(7, 22, 35, 43, 47)(8, 25, 33, 37, 51) \cdot$$
$$(9, 17, 31, 42, 55)(10, 20, 36, 48, 10) \cdot$$
$$(11, 23, 27, 41, 52)$$

Using c and d as generators and the relation $c^3 d = dc$, we obtain the nonabelian permutation group P of order 55.

For the base graph $K_{2,3}$, we just have two nontrivial voltages to assign (see Figure 4.) We assign the edge oriented from v_2 to c_2 the voltage c, and the edge oriented from v_3 to c_2 the voltage d. Thus, letting i denote the identity permutation, the parity-check matrix has array form

$$\begin{bmatrix} i & i & i \\ i & c & d \end{bmatrix}$$

where the $m \times m$ permutation matrices are given by the permutations i, c, and d. For the base graph $K_{3,5}$, we assign the nontrivial voltages $c, c^2d^2, c^5d, c^8d^3, d, c^3d^3, c^4d^4$, and c^6d^2 to obtain a parity-check matrix with the following array form:

$$\begin{bmatrix} i & i & i & i & i \\ i & c & c^2d^2 & c^5d & c^8d^3 \\ i & d & c^3d^3 & c^4d^4 & c^6d^2 \end{bmatrix}$$

The nontrivial voltages were chosen from distinct cyclic subgroups and therefore do not commute. The code defined by the $K_{2,3}$ permutation voltage graph has block length 165 and code rate 0.33. One can check that the net voltages of each 4-cycle in the base graph is a permutation with smallest cycle at least 5 in its permutation cycle decomposition. The cycle with net voltage d has cycles of length 5 in its decomposition. Consequently, the pre-image of each 4-cycle in the derived graph contains cycles of length at least $4 \times 5 = 20$, with some equal to 20. Since the corresponding code is a cycle code, the distance is equal to half the girth. Thus, the distance of the coresponding code is 10. The code defined by the $K_{3,5}$ permutation voltage graph with parity-check matrix above has block length 275 and code rate 0.4. Note that no element in the permutation group P has a cycle of size < 5 in its cycle decomposition, except for the identity. Thus the net voltage along any closed walk will not have fixed points or cycles of length less than 5 in the cycle decomposition, unless the net voltage is the identity. This technique can be extended naturally to other arrays and other base graphs.

V. Simulation Results

In this section, we examine the performance of the proposed codes and compare them with codes that are either designed randomly or designed using the voltage approach with randomly chosen permutation assignments.

Figure 5 shows the performance of $(2, 3)$-regular LDPC codes over the binary input additive white Gaussian noise channel (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 a voltage graph based LDPC code presented in Section IV, where the voltage graph is the complete bipartite graph $K_{2,3}$ on two left and three right vertices, respectively, and the corresponding derived graph is obtained by assigning voltages to $K_{2,3}$ to the permutation group of size 55. The block length of the resulting code is 165 and the code rate is approximately 0.33. Also shown in the figure are the performances of a) a $(2, 3)$-regular LDPC code of same block length designed randomly, b) a $(2, 3)$-regular LDPC code designed using the array or SFT-type construction of [7], and c) a $(2, 3)$-regular LDPC code designed using the voltage graph approach wherein the voltages are chosen randomly from the symmetric group S_{55}. Clearly, the proposed construction that optimizes the cycle structure in the derived graph substantially outperforms these other constructions. The proposed construction has a gain of approximately 1 dB compared to a randomly designed graph, and a gain of more than 1 dB compared to the array construction and a voltage construction that uses randomly chosen permutations. Thus, the results indicate that using a voltage graph approach to construct lifts algebraically is an effective way to optimize the properties of the derived graph for improving code performance.

Figure 6 shows the analogous performance of $(3, 5)$-regular LDPC codes over the BIAWGNC under sum-product decoding. The figure shows that code proposed in Section IV that is designed by assigning the edges of a $K_{3,5}$ to voltages from a permutation group of size 55 performs better, by about 0.5 dB, than a corresponding code that is also designed using the voltage approach wherein the voltages are assigned randomly from S_{55}. The block length of the codes are 275 and the code rate is approximately 0.40. An array or SFT-type construction for a $(3, 5)$-regular LDPC code of comparable parameters has performance very similar to the proposed code in this case. Thus, the above results indicate that the proposed construction yields codes that perform at least as well, if not better than, other structured constructions proposed earlier. In addition, we believe that when the relationship between distance and voltage assignments is determined, it will be clearer how to choose even more effective voltages. In particular, in the above assignment, we chose the voltages so that the exponent of c and d appearing in each row were distinct, and this gave a better code than when all the nontrivial voltages had the form $c^i d$, for $i \in \{0, 1, \ldots 10\}$.

Finally, due to the inherent algebraic structure introduced in the construction, the codes proposed in Section IV have a succinct description that make them attractive candidates for implementation in practical applications.

VI. Conclusions

We presented a construction scheme for codes based on permutation voltage graphs. This gives a family of codes that may be viewed as 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. The examples in this paper use the complete bipartite graph as a base graph, but the method may be applied to any base graph, such as those with optimal degree distributions. This work is currently being extended in many ways. First, other nonabelian groups may be used in this construction with the same general method and are being investigated. This will allow for a

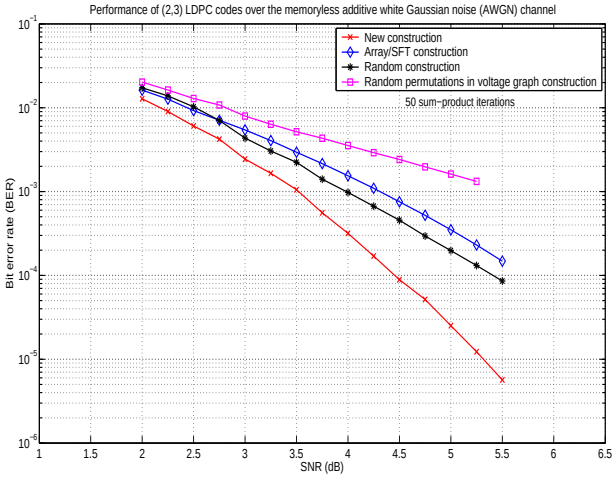


Fig. 5. Performance of (2,3)-LDPC codes on the BIAWGNC under sum-product decoding.

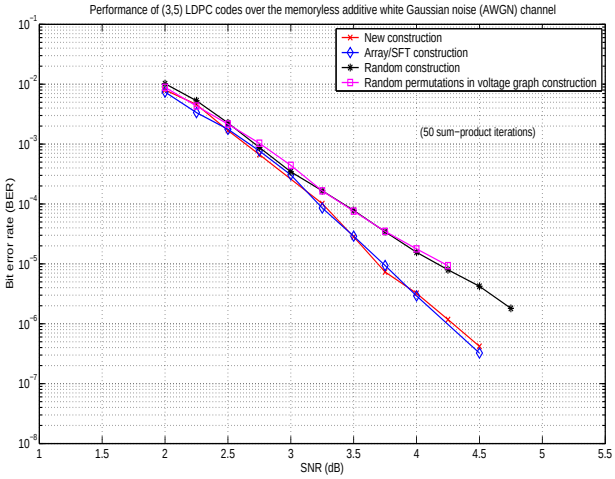


Fig. 6. Performance of (3,5)-LDPC codes on the BIAWGNC under sum-product decoding.

wider range of permutation groups, and therefore, more block lengths will be possible. Nonabelian groups with more generators will also make the choice of voltages easier for larger base graphs. Secondly, we are also interested in constructions using ordinary voltage assignments. Finally, we hope to determine an explicit relationship between the distance properties of the resulting codes and the voltage assignments. Such a result would help optimize the choice of permutation (or ordinary) voltages, and could be directly incorporated into the code constructions. In conclusion, the codes constructed in this paper demonstrate that codes based on algebraic lifts have the potential and the ability to outperform random codes, as well as those based on random lifts.

Acknowledgement: The author thanks D. Sridhara for assistance with the simulations.

REFERENCES

[1] J. Thorpe, "LDPC codes constructed from protographs", *IPN progress report*, pp. 42-154, JPL, August 2003.

[2] J. Thorpe, K. Andrews, and S. Dolinar, "Methodologies for designing LDPC codes using protographs and circulants.", in *Proceedings of the IEEE International Symposium on Information Theory*, p. 236, Chicago, June 2004.

[3] D. Divsalar, S. Dolinar, and C. Jones, "Construction of protograph LDPC codes with minimum distance linearly growing with block size",*IEEE Globecom*, St. Louis, MO, Nov. 2005, pp. 1152?1156 2005.

[4] J.L. Gross and T.W. Tucker, *Topological graph theory*, Wiley, NY, 1987.

[5] C. A. Kelley and J. L. Walker, "LDPC codes from voltage graphs", in *Proceedings of the International Symposium on Information Theory*, Toronto, Canada, June 2008.

[6] R. M. Tanner, D. Sridhara, and T. E. Fuja, "A class of group-structured LDPC codes", in *Proc. of Intl. Symp. on Communication Theory and Applications*, Ambleside, U.K., pp. 365–370, July 2001.

[7] D. Sridhara, T. E. Fuja, and R. M. Tanner, "Low density parity check codes from permutation matrices", 2001 Conf. on Info. Sci. & Systems, Johns Hopkins University, March 2001.

[8] J. L. Fan, "Array codes as low-density parity-check codes", in *Proceedings of the 2nd International Symposium on Turbo Codes and their applications*, Brest, France, Sept. 2000, pp. 543–546.

[9] O. Milenkovic, N. Kashyap, and D. Leyba, "Shortened array codes of large girth," in *IEEE Transactions on Information Theory*, Vol. 5, No. 8, pp. 3707-3722, August 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. of Conf. on Info. Systems and Sciences*, March 22-24, 2006, pp. 835-838.

[11] R. M. Tanner, "Quasi-cyclic repeat accumulate codes", in *Proc. 37th Allerton Conference on Communication, Control and Computing*, Monticello, IL, Oct. 1999, pp. 249?259.

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

[13] A. Amit and N. Linial, "Random lifts of graphs II: Edge expansion", *Combinatorics Probability and Computing*, vol. 15, pp. 317-332, 2006.

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

[15] G. Exoo, "Voltage graphs, group presentations, and cages", *The Electronic Journal of Combinatorics*, vol. 11(1), 2004.

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

[17] M. P. C. Fossorier, "Quasi-cyclic low-density parity-check codes from circulant permutation matrices", *IEEE Transactions on Information Theory*, vol.50, no.8, pp.1788?1793, 2004.

[18] 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, Eds., vol. 123 of *IMA Volumes in Mathematics and its Applications*, pp. 113 -130. Springer, New York, 2000.

[19] R. Smarandache and P.O. Vontobel, "On regular quasi-cyclic LDPC codes from binomials", in *Proceedings of the IEEE International Symposium on Information Theory*, Chicago, IL, June 2004.

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

[21] K. Yang and T. Helleseth, "On the minimum distance of array codes as LDPC codes", in *IEEE Transactions on Information Theory*, vol. 49, No. 12, December 2003.

[22] J. Chen, R.M. Tanner, J. Zhang, M.P.C. Fossorier, "Construction of irregular LDPC codes by quasi-cyclic extension", *IEEE Transactions on Information Theory*, pp. 1479-1483, Vol. 53, No. 4, April 2007.

[23] R. M. Tanner, "A recursive approach to low complexity codes," in *IEEE Transactions on Information Theory*, vol. IT-27, pp. 533-547, Sept. 1981.

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

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

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