Convolutional Codes on Trees and Cayley Graphs

S. Ali Miri[†] and Frank R. Kschischang[‡]

***Abstract*—Convolutional codes are usually defined as shift-invariant linear spaces (or, more generally, groups) over the integer discrete-time index set $\mathbb{Z}$. In this work, we generalize the notion of discrete-time index sets to infinite regular trees. By viewing the infinite regular tree as the Cayley graph of certain free products of groups, and taking the group generators as shift operators on the tree vertices, we define general shift-invariant group codes and, in particular, convolutional codes on trees. Relative to their conventional time-axis counterparts, such codes may have larger minimum Hamming distance for the same state-space complexity. We also introduce a generalization of conventional tail-biting to deal with the efficient termination of such codes.**

I. Introduction

The near capacity-achieving performance of turbo codes and low-density parity-check codes has focused much attention on codes defined on graphs and iterative decoding (see, e.g., [1]). While the majority of work in this area focuses on random graph constructions, some structured approaches based on Cayley graphs and Ramanujan graphs [2–5] have been shown to yield codes with desirable properties, particularly at short to moderate block lengths. Graphical code representations such as factor graphs [6] can be used to define the relationships between codeword symbols and various constraints, and the graph itself may often be regarded as defining a generalized "time-axis" for the code. Motivated by the fact that the sum-product algorithm is exact in cycle-free graphs [6], we wish to study codes defined on infinite cycle-free graphs: namely, trees.

In this paper, we regard an infinite regular tree of small finite degree as the Cayley graph of a free product of certain groups. In Section III, we define shift operators and shift-invariant linear codes (or, more generally, shift-invariant group codes) on trees. In Section IV we sketch out how the conventional algebraic descriptions of convolutional codes may be extended to handle our generalization. We observe that—unlike their conventional counterparts—termination of convolutional codes on trees leads to an unacceptable rate loss. To deal with this issue, in Section V we introduce a notion of generalized tail-biting by introducing relations among the generators of the free product. The resulting Cayley graph necessarily has cycles, which will strongly influence iterative decoding performance. However, if the relations are chosen judiciously, the girth of the graph, i.e., the length of the shortest graph cycle, can be made large relative to the group order. We begin with some mathematical preliminaries.

[†] Dept. of Mathematics, University of Toronto, Toronto, Ontario, M5S 3G3, `samiri@math.utoronto.ca`

[‡] Dept. of Electrical & Computer Engineering, University of Toronto, Toronto, Ontario, M5S 3G4, `frank@comm.utoronto.ca`

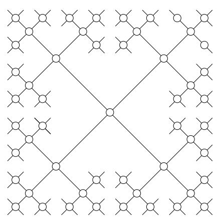

Fig. 1. A portion of a 4-regular tree.

II. Preliminaries

We adopt the usual terminology of graph theory (see, e.g., [7]): a (simple) *graph* is a pair (V, E), where V is a nonempty set of *vertices* and E is a set of *edges*. An edge is an unordered pair $\{u, v\}$ of vertices $u, v \in V$, $u \neq v$. Given a finite positive integer k, a graph is *k-regular* if the degree of each vertex is equal to k. A *cycle* in a graph is a (simple) path without repeated edges from a vertex to itself. The *girth* of a graph is defined as the length of the shortest cycle in the graph. A graph is *connected* if there is at least one path between any pair of vertices, and *cycle-free* if there is at most one path between any pair of vertices. The *graph distance*, $d(u, v)$, between two vertices u and v in a graph is equal to the length of the shortest path from u to v, if such a path exists, where $d(u, v) = 0$ if $u = v$.

A *tree* is a connected, cycle-free graph. A tree is infinite if its vertex set is infinite. In this paper we define convolutional codes over time index sets that are infinite k-regular trees. For example, Fig. 1 shows (a portion of) an infinite 4-regular tree.

We may regard an infinite k-regular tree as the *Cayley graph* (see, e.g., [8]) of certain free products of groups. Cayley graphs have been known and used for over a century, and are defined as follows. Given a group G and a set of generators S for G, the Cayley graph $C(G, S)$ has vertex set G, and edge set that includes an edge $\{v, w\}$ if and only if $v = ws$ or $w = sv$ in G, where $s \in S$ is a generator. To avoid self-loops in the Cayley graph, we consider only generating sets S that do not contain the group identity. When S is understood from context or irrelevant to the discussion, we write $C(G)$ to designate a Cayley graph corresponding to group G.

We consider the vertex corresponding to the group identity as the *root*, $\mathcal{O}$, of the Cayley graph. Paths starting from $\mathcal{O}$ can be identified with products of the elements of S (and their inverses). Since, by construction, S generates G it is easy to see that Cayley graphs are connected.

However, Cayley graphs are not, in general, cycle-free.

Cycles in a Cayley graph starting and ending at $\mathcal{O}$ correspond to nontrivial relations among the generating elements. For example, if $s^n = e$ for some $n > 2$, where $s \in S$ and e is the group identity, then the Cayley graph will have cycles of length n. To avoid cycles, an element $s \in S$ must either satisfy no relations at all, or satisfy a relation of the form $s^2 = e$. In the latter case, no cycle is formed, since the (non-simple) path $(\mathcal{O}, s, \mathcal{O})$ traverses only the single edge $\{\mathcal{O}, s\}$. Thus we are led to consider groups that are the *free product* of a copies of $\mathbb{Z}$ (each generated by a single element) and b copies of $\mathbb{Z}/2\mathbb{Z}$ (each generated by a single element).

Free products are defined, e.g., in [9, Ch. 17]. Loosely, the free product of a collection of groups G_i, $i \in I$, consists of all *reduced words* (i.e., strings) composed of symbols drawn from the groups, with string concatenation as group operation. The empty string serves as the group identity. Reduction of an arbitrary string to the corresponding reduced word is achieved by substring substitution: any occurrence of a group identity (i.e., the identity element from some G_i) is replaced by the empty string, and any occurrence of two adjacent symbols from the same group G_i is replaced by their product in that group.

Let $\{\sigma_1, \ldots, \sigma_a\}$ be a set generators that satisfy no nontrivial relations, so that $\langle \sigma_i \rangle \cong \mathbb{Z}$, $1 \leq i \leq a$. Let $\{\sigma_{a+1}, \ldots, \sigma_{a+b}\}$ be a set of involutory generators, each satisfying the relation $\sigma_i^2 = e$, where e is the group identity, so that $\langle \sigma_i \rangle \cong \mathbb{Z}/2\mathbb{Z}$, $a+1 \leq i \leq a+b$. Finally, let G be the free product

$$G = \langle \sigma_1 \rangle * \langle \sigma_2 \rangle * \cdots * \langle \sigma_{a+b} \rangle. \tag{1}$$

The set

$$\Sigma = \{\sigma_1, \ldots, \sigma_{a+b}\} \tag{2}$$

is clearly a generating set for G.

With this setup, we have the following theorem.

Theorem 1: Let G and Σ be defined as in (1) and (2). The Cayley graph $C(G, \Sigma)$ is an infinite $(2a+b)$-regular tree.

Proof: Straightforward, omitted.

In fact, the converse to this theorem is also true (see Serre [10]).

III. Shift-invariant Group Codes

A group G acts naturally on the vertices of its Cayley graph $C(G)$ via left multiplication, i.e., given a group element g and a vertex v, the map $g \cdot v = gv$ is a (left) action on the vertex set. As always, for each fixed $g \in G$, the map π_g defined as $\pi_g(v) = gv$ is a bijection, i.e., a permutation, of the vertices. We can extend this mapping to the edges of the Cayley graph by defining $\pi_g(\{v, w\}) = \{gv, gw\}$. It is not hard to show the resulting graph mapping is a graph automorphism, i.e., a graph isomorphism from $C(G)$ to itself. Furthermore, the mapping is an *isometry*, since for any pair of vertices u and v, $d(\pi_g(u), \pi_g(v)) = d(u, v)$. In effect, the map π_g translates the root $\mathcal{O}$ to vertex g, leaving all other neighbourhood structure of the graph unchanged.

Let A be a nonempty alphabet, and let $\Gamma = (V, E)$ be a graph. By a *code over A defined on Γ* we mean a nonempty subset C of A^V, i.e., a subset of the set of words with components indexed by V. Given a word $w \in A^V$ and a vertex $v \in V$, we denote the component of w at v by w_v, and we often refer to v as a "time-index." When A is a group, then A^V is a group, and we usually take C to be a subgroup, in which case C is a *group code* over A defined on Γ.

Now, let $\Gamma = C(G)$ be the Cayley graph of a group G. We can define an action of G on the set A^V by setting $g \cdot w = w'$, where $w'_v = w_{g^{-1}v}$ for all $w \in A^V$ and all $g \in G$. When g is not the identity, the word $g \cdot w$ is said to be a *shift* of the word w.

Let H be a subgroup of G, and let C be a code defined on $C(G)$. The code C is said to be *H-shift-invariant* if, for all $c \in C$ and all $h \in H$, we have $h \cdot c \in C$. If, in particular C is H-invariant for $H = G$, then C is *fully shift-invariant.*

Examples of fully shift-invariant codes include conventional convolutional codes ($G = \mathbb{Z}$) and cyclic codes of length n ($G = \mathbb{Z}/n\mathbb{Z}$). Quasi-cyclic codes of length n are H-shift-invariant for a proper subgroup of $\mathbb{Z}/n\mathbb{Z}$. We are interested in defining shift-invariant group codes over infinite k-regular trees, in which case we take G as defined in (1).

IV. Algebraic Structure

We sketch now how convolutional codes on trees can be given an algebraic structure, extending the results on convolutional codes presented originally in [11]. In particular, we show that the words of finite support in a convolutional code defined over a tree can be written (essentially) as polynomials in "delay operators" D_i corresponding to the shifts σ_i defined in (2). Let G be defined as in (1). Let Γ be the corresponding infinite k-regular tree with vertex set V. For simplicity, we will suppose that the convolutional code is defined over a finite field $\mathbb{F}$, though much of this formalism extends more generally to group codes.

We take as (output) symbol alphabet A the vector space $\mathbb{F}^n$. The zero vector is denoted $\mathbf{0}$. A word $w \in A^V$ is said to have *finite support* if it has only finitely many non-$\mathbf{0}$ components.

If $w_v \neq \mathbf{0}$, where $v \in G$, then $v = s_1 s_2 \ldots s_n$ is a reduced word, where each component s_j is σ_i^k for some $\sigma_i \in \Sigma$. Define $D(s_j) = D_i^k$, where D_i is an indeterminate. Each such nonzero component of w contributes a term of the form $w_{s_1 \ldots s_n} D(s_1) D(s_2) \cdots D(s_n)$ to the formal sum that comprises the overall "D-transform" of w. It is important to note that the shift operators do not commute, i.e., if $D_i \neq D_j$, then $D_i D_j \neq D_j D_i$. The resulting algebra is thus fundamentally non-commutative, except in the conventional case.

When C is fully shift-invariant, it is possible to define

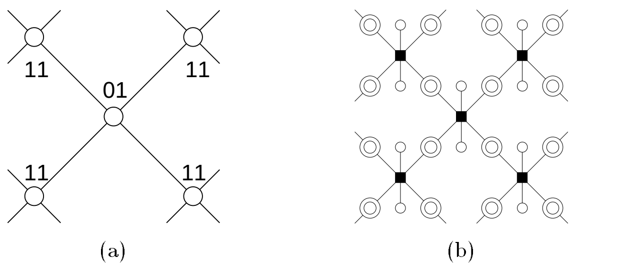

Fig. 2. (a) The "impulse response" for a binary convolutional code defined the infinite 4-regular tree. (b) The corresponding factor graph.

a basis word for C, such that each output word of finite support can be viewed as a linear combination of shifted versions of the basis word. In this case, we can define a mapping from input sequences to output sequences via an appropriate multiplication by a *generator matrix.* (In the linear case, the basis word generates an ideal in a particular ring.) Though space limitations preclude a full exposition of these ideas, an example should help clarify them.

Example 1: Consider a fully shift-invariant code over the 4-regular tree containing the codeword of finite support shown in Fig. 2(a). We assume that $\Sigma = \{\sigma_1, \sigma_2\}$. The "$D$-transform" of this codeword is given by

$$G(D) = (10) + (11)D_1 + (11)D_2 + (11)D_1^{-1} + (11)D_2^{-1}.$$

The convolutional code generated by $G(D)$ is then the set of all codewords of finite support that can be generated by linear combination of a finite number of shifts of this sequence. This set can be described as the image of the set of D-transforms corresponding to input sequences of finite support under right multiplication by $G(D)$. It is easy to show that minimum weight of any nonzero codeword is 9.

The notion of states (or hidden variables) for codes on graphs, as introduced by Wiberg, et al. [12, 13], and generalized by Forney [14] applies also to codes defined on trees. When the underlying graph is cycle-free so that the removal ("cut") of any graph disconnects the graph, the concept of state is well-defined. Roughly speaking, the state variables convey the information needed to make the variables on one side of the cut (the "past") conditionally independent of the the variables on the other side of the cut (the "future"), given the state variables.

Thus, we associate a hidden state variable with each edge of the graph. The size of the state-space (the cardinality of the domain of the state variable) at an edge for a group code C is well known [15, 16] to be the order of the quotient group $C/(C_pC_f)$, where C_p and C_f represent, respectively, the subgroup of codewords whose region of support is confined to one (or other) side of the cut.

Example 1 (cont.): Consider the convolutional code defined on the infinite 4-regular tree of Example 1. Fig. 2(b) shows part of state-space realization (factor graph [6]) induced by cuts at edges. The double circles represent states variables, the empty circles represent inputs and outputs, and the filled squares represent the *check* structure defined for every vertex on the tree. Since only two possible shifts of the generator have a support region that spans each edge, the dimension of the state-space of is two (corresponding to four states), as stated earlier.

V. Generalized Tail-biting

A potential drawback is the question of termination of these codes, which can lead to a large rate loss. In the example above, suppose the support region for input sequences is the set of graph vertices within distance d of $\mathcal{O}$. The output bits are then confined to a support region within distance $d+1$, and it is easy to show that the code rate, as a function of d, is given by

$$R(d) = \frac{1}{2} \cdot \frac{(2 \cdot 3^{d+1} - 1)}{(2 \cdot 3^{d+2} - 1)}.$$

The limiting rate, for large values of d, is given by 1/6, which is a factor 1/3 smaller than the expected rate 1/2.

We will address this problem by introducing a generalized notion of tail-biting by allowing a systematic *folding* of the tree onto itself. This will result in codes with the expected rates.

Conventional (feedforward) convolutional encoders with memory order m are terminated by appending a block of m zero-valued "tail bits" to the message, resulting in a codeword with a state-space representation that both starts and ends in the zero state. This method may introduce a large rate loss if the number of trellis sections is small.

Tail-biting [17, 18] is a well-known method for avoiding this rate loss. The encoder is started in the state equal to the last m bits of the message. In this way, the starting and ending states coincide, resulting in a "circular" trellis representation. (In our terminology, the result is a code defined on a Cayley graph of the cyclic group.) Tail-biting trellis representations may have fewer states than conventional trellis (see, e.g., [19]) representations for the same code.

In the case of convolutional codes over trees, any finite connected subgraph of the tree has a boundary with an exponentially large number of vertices (except in the degree-2 case) which causes large rate loss in termination. We cope with this problem by *folding* the tree onto itself in a generalization of conventional tail-biting. This can be achieved by introducing certain relations between the generators of the free group, thereby generating a finite group. We will give a few examples to illustrate that this construction can be done systematically to lead to graphs where the group order is relatively small and the size of the smallest cycles is relatively large.

Example 2: Consider the k-regular tree of degree two corresponding to the free product G of groups generated by two involutory generators x, and y. The resulting Cayley graph is shown in Fig. 3(a). Next, we introduce the relation that $xyx = yxy$ (or, equivalently, $(xy)^3 = e$)

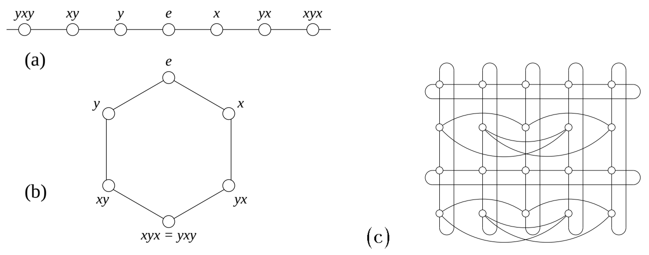

Fig. 3. Three Cayley graphs: (a) of a free product; (b) of the free product modulo a relation; (c) a more complicated example.

to form a new group. Formally, we form the quotient $G/N((xy)^3)$, where $N((xy)^3)$ is the smallest normal subgroup of G containing $(xy)^3$. The group G/N has the (tail-biting) Cayley graph shown in Fig. 3(b).

Theorem 2: Let G be a group with generating set S, and let $\Gamma = C(G, S)$ be its Cayley graph. Then the girth of Γ is the smallest $g > 2$ such that there exists an $x \in S$ and a reduced word $s_1 s_2 \cdots s_{g-1}$, $s_j \in S$, such that $s_1 s_2 \cdots s_{g-1} x = e$.

The girth in the above example is 6, and corresponds to the reduced word $xyxyxy = e$.

Example 3: (*Cyclic extensions of cyclic groups*) Consider the cyclic group of order q generated by an element s of order q. Let r be an integer relatively prime to q, and c be its multiplicative order modulo q, that is

$$r^c \equiv 1 \bmod q.$$

A bigger group G of order qc can be constructed by adjoining a new element t of order c, such that G is defined by the following relations:

$$s^q = t^c = e, \quad t^{-1} s t = s^r.$$

The Cayley graph of the free group on the set of generators $\{s, t\}$ (without the relations above) is the 4-regular tree shown in Fig. 1. Let $q = 5$, and $r = 2$. The Cayley graph of the group generated using the above construction is shown in Fig. 3(c).

Using the Cayley graph of the group, or the relations between generators; it is easy to show that the girth for this graph is 4. In general, the task is to find values of c and r which will result in the largest girth and smallest group order.

Consider a 4-regular graph Γ. Let $n(\Gamma)$ and $c(\Gamma)$ denote the number of vertices (i.e. the group order) and the girth of Γ, respectively. It is easy to show that [20]

$$c(\Gamma) \leq 2 \log_3 \left(\frac{n(\Gamma)}{2} + 1 \right) + 1.$$

Here are a few typical values for girth of graphs using the above construction:

q	c	r	Girth	Ord G	Min. ord(G) needed
5	4	2	4	20	10
11	5	3	5	55	16
13	6	4	6	78	30
19	9	4	7	171	52
31	15	7	10	465	279

VI. Conclusions

We have defined shift-invariant group codes on infinite k-regular trees, viewed as the Cayley graphs of certain free products. Shift operators correspond to the group generators. We have illustrated that these codes in general possess a non-commutative algebraic structure that generalizes the conventional case. Termination of these codes leads to a catastrophic rate loss. To deal with this rate loss, we have proposed introducing relations in the free products, essentially "tail-biting" the tree into the Cayley graph of a finite group.

Much work remains to be done, particularly to determine which generators will lead to convolutional codes with large minimum distance for a given state complexity, and which relations yield Cayley graphs with large girth. The resulting codes should be simulated to determine their performance under iterative decoding.

References

[1] *IEEE Trans. on Inform. Theory,* vol. 47, Feb., 2001. Special Issue on Codes on Graphs and Iterative Algorithms.
[2] M. Sipser and D. A. Spielman, "Expander codes," *IEEE Trans. Inform. Theory,* vol. 42, pp. 1710–1722, Nov. 1996.
[3] A. L. Berger and J. D. Lafferty, "Probabilistic decoding of low-density Cayley codes," in *Proc. 5th Canadian Workshop on Inform. Theory,* (Toronto, Canada), pp. 11–14, June 1997.
[4] J. lafferty and D. Rockmore, "Codes and iterative decoding on algebraic expander graphs," in *Proc. of Int. Symp. on Inform. Theory and Its Applications,* (Honolulu, Hawaii), Oct. 2000.
[5] J. Rosenthal and P. O. Vontobel, "Construction of LDPC codes using Ramanujan graphs and ideas from Margulis," in *Proc. of 38th Allerton Conf. on Commun., Control, and Comp.,* (Monticello, IL), Oct. 2000.
[6] F. R. Kschischang, B. J. Frey, and H.-A. Loeliger, "Factor graphs and the sum-product algorithm," *IEEE Trans. on Inform. Theory,* vol. 47, pp. 498–519, Feb. 2001.
[7] D. B. West, *Introduction to Graph Theory.* Upper Saddle River, NJ: Prentice-Hall, 2nd ed., 2001.
[8] H. S. M. Coxeter and W. O. J. Moser, *Generators and Relations for Discrete Groups.* Springer-Verlag, 1972.
[9] M. Hall, Jr., *The Theory of Groups.* Providence, RI: AMS Chelsea, 2nd ed., 1976.
[10] J.-P. Serre, *Trees.* Berlin Heidelberg: Springer-Verlag, 1980.
[11] G. D. Forney, "Convolutional codes I: Algebraic structure," *IEEE Trans. Inform. Theory,* vol. IT-16, pp. 720–738, Sept. 1970.
[12] N. Wiberg, H. A. Loeliger, and R. Kötter, "Codes and iterative decoding on general graphs," *Euro. Trans. Telecomm.,* vol. 6, pp. 513–525, Sept. 1995.
[13] N. Wiberg, *Codes and Decoding on General Graphs.* PhD thesis, University of Linköping, Sweden, 1996.
[14] G. D. Forney, "Codes on graphs: Normal realizations," *IEEE Trans. Inform. Theory,* vol. 47, pp. 520–548, Feb. 2001.
[15] J. C. Willems, "Models for dynamics," in *Dynamics Reported 2* (U. Kirchgraber and H. O. Walther, eds.), pp. 171–269, Wiley and Teubner, 1989.
[16] G. D. Forney and M. D. Trott, "The dynamic of group codes: State spaces, trellis diagrams and canonical encoders," *IEEE Trans. Inform. Theory,* vol. IT-39, pp. 1491–1513, Sept. 1993.
[17] G. Solomon and H. C. A. V. Tilborg, "A connection between block and convolutional codes," *SIAM J. Appl. Math.,* vol. 37, pp. 358–369, Oct. 1979.
[18] J. H. Ma and J. K. Wolf, "On tail-biting convolutional codes," *IEEE Trans. Commun.,* vol. COM-34, pp. 104–111, Feb. 1986.
[19] A. R. Calderbank, G. D. Forney, and A. Vardy, "Minimal tail-biting trellises: The Golay code and more," *IEEE Trans. Inform. Theory,* vol. 45, pp. 1435–1455, July 1999.
[20] G. A. Margulis, "Explicit constructions of graphs without short cycles and low density codes," *Combinatorica,* vol. 2, no. 1, pp. 71–78, 1982.