On Parity-Preserving Constrained Coding

Ron M. Roth

Computer Science Department
Technion, Haifa 320003, Israel
ronny@cs.technion.ac.il

Paul H. Siegel

ECE Department and CMRR
UC San Diego, La Jolla, CA 92023, USA
psiegel@ucsd.edu

***Abstract*—Necessary and sufficient conditions are presented for the existence of fixed-rate parity-preserving encoders for a given constraint. It is also shown that under somewhat stronger conditions, the stethering method guarantees an encoder that has finite anticipation.**

I. Introduction

Runlength limited (RLL) coding is widely employed in magnetic and optical storage in order to mitigate the effects of inter-symbol interference and clock drifting [3]. The encoder typically takes the form of a finite-state machine, which maps a sequence of input p-bit blocks into a sequence of output q-bit codewords, so that the concatenation of the generated codewords satisfies the RLL constraint. In most applications, the coding scheme also provides DC control (or, more generally, suppression of the low frequencies). This is achieved by allowing some (or all) input p-blocks to be mapped by the encoder to more than one codeword and then, during the encoding process, selecting the codeword that yields the best DC suppression [7, p. 29]. One implementation of this strategy uses *parity-preserving* encoders, whereby the parity (i.e., the modulo-2 sum) of the input sequence within non-overlapping windows (each consisting of one or more p-blocks) is preserved at the output. DC control can be achieved by reserving one input bit in that window and selecting its value so as to minimize the DC contents [3, §11.4.3], [4], [8], [9], [10], [11]. Parity-preserving RLL codes are used in the Blu-ray standard.

Most constructions of parity-preserving codes so far were obtained by ad-hoc methods. The purpose of this work is to initiate a study of parity-preserving encoders, starting with the special case where the window length over which the parity is preserved is a fixed multiple of p. We provide a formal definition of our setting in Section I-B below, preceded by some background and definitions which are taken from [7].

A. Background

A (finite labeled directed) graph is a graph $G = (V, E, L)$ with a nonempty finite state (vertex) set $V = V(G)$, finite edge set $E = E(G)$, and edge labeling $L : E \to \Sigma$. The constraint presented by G, denoted $S(G)$, is the set of words over Σ that are generated by finite paths in G. A graph G is deterministic if all outgoing edges from a state are distinctly labeled. Every constraint has a deterministic presentation. A graph G is lossless if no two paths with the same initial state and the same terminal state generate the same word. The anticipation $\mathcal{A}(G)$ of G is the smallest nonnegative integer a (if any) such that all paths that generate any given word of length $a+1$ from any given state in G share the same first edge. A graph is said to have finite memory μ if μ is the smallest nonnegative integer (if any) such that all paths of length μ that generate the same word terminate in the same state.

A graph G is irreducible if it is strongly-connected. A constraint S is irreducible if it can be presented by a deterministic irreducible graph.

The power G^t of a graph G is the graph with the same set of states $V(G)$ and edges that are the paths of length t in G; the label of an edge in G^t is the length-t word generated by the path. For $S = S(G)$ the power S^t is defined as $S(G^t)$.

Given a constraint S over an alphabet Σ and a lossless presentation G of S, the capacity of S is defined by $\mathsf{cap}(S) = \lim_{\ell\to\infty}(1/\ell)\log_2|S \cap \Sigma^\ell|$. It is known that $\mathsf{cap}(S) = \log_2\lambda(A_G)$ where $\lambda(A_G)$ denotes the spectral radius (Perron eigenvalue) of the adjacency matrix A_G.

Given a constraint S and a nonnegative integer n, an (S, n)-encoder is a lossless graph $\mathcal{E}$ such that $S(\mathcal{E}) \subseteq S$ and each state has out-degree n. An (S, n)-encoder exists if and only if $\log_2 n \le \mathsf{cap}(S)$. In a *tagged* (S, n)-encoder, each edge is assigned an input tag from a finite alphabet of size n, such that edges outgoing from the same state have distinct tags. The anticipation (if finite) of an encoder determines its decoding delay.

A (tagged) rate $p : q$ encoder for a constraint S is a (tagged) $(S^q, 2^p)$-encoder (the tags are then assumed to be from $\{0, 1\}^p$). A rate $p : q$ parity-preserving encoder for a constraint S over $\Sigma = \{0, 1\}$ is a tagged encoder for S in which the parity of the (length-q) label of each edge matches the parity of the (length-p) tag that is assigned to the edge (see also Section I-B below).

Given a square nonnegative integer matrix A and a positive integer n, an (A, n)-approximate eigenvector is a nonnegative nonzero integer vector $\boldsymbol{x}$ that satisfies the inequality $A\boldsymbol{x} \ge n\boldsymbol{x}$ componentwise. The set of all (A, n)-approximate eigenvectors will be denoted by $\mathcal{X}(A, n)$, and it is known that $\mathcal{X}(A, n) \ne \emptyset$ if and only if $n \le \lambda(A)$. Given a constraint S presented by a deterministic graph G and a

This work was done in part while R.M. Roth was visiting the Center for Memory and Recording Research (CMRR), UC San Diego. This work was supported by Grant 2015816 from the United-States–Israel Binational Science Foundation (BSF), by NSF Grant CCF-BSF-1619053, and by Grant 1396/16 from the Israel Science Foundation.

978-1-5386-4780-6/18/$31.00©2018 IEEE

positive integer n, the state-splitting algorithm provides a method for transforming G, through an (A_G, n)-approximate eigenvector, into an (S, n)-encoder with finite anticipation.

For a positive integer b, the set $\{0, 1, 2, \ldots, b-1\}$ will be denoted by $[b\rangle$.

B. Parity-preserving encoders

Let S be a constraint over an alphabet Σ, and fix a partition $\{\Sigma_0, \Sigma_1\}$ of Σ. The symbols in Σ_0 (resp., Σ_1) will be referred to as the *even* (resp., *odd*) symbols of Σ. The partition of Σ to two elements (only) follows from the primary motivation of this work, namely, constructing and analyzing parity-preserving encoders. However, without much further effort, the definitions and results can be extended to a partition of Σ into any number of partition elements.

Given a graph H with labeling in Σ, for $\mathsf{b} \in [2\rangle$, we denote by H_{b} the subgraph of H containing only the edges with labels in Σ_{b}.

Let $S = S(G)$ be a constraint and n_0 and n_1 be nonnegative integers. An (S, n_0, n_1)-*encoder* $\mathcal{E}$ is an $(S, n_0{+}n_1)$-encoder such that for each $\mathsf{b} \in [2\rangle$, the subgraph $\mathcal{E}_{\mathsf{b}}$ is an (S, n_{b})-encoder. In other words, from each state in $\mathcal{E}$, there are n_0 outgoing edges with even labels and n_1 outgoing edges with odd labels. For the applications in mind where we are interested in rate $p : q$ parity-preserving encoders for constraints S over the binary alphabet, the set Σ_0 (resp., Σ_1) will contain length-q words in S having even (resp., odd) parity (see Section I-D and Example 2 in Section III-B). Then any rate $p : q$ parity-preserving encoder for S is a (tagged) $(S^q, 2^{p-1}, 2^{p-1})$-encoder and, conversely, any $(S^q, 2^{p-1}, 2^{p-1})$-encoder can be tagged so that it is parity preserving.

When studying $(S(G), n_0, n_1)$-encoders, there is no loss of generality in assuming that both G and the encoder are irreducible. Indeed, if $\mathcal{E}$ is an $(S(G), n_0, n_1)$-encoder, then an irreducible sink of $\mathcal{E}$ is an (S', n_0, n_1)-encoder, where S' is an irreducible constraint presented by some irreducible component of G (see the proof of [6, Proposition 3]). Note that G_0, G_1, $\mathcal{E}_0$, and $\mathcal{E}_1$ may still be reducible even when G and $\mathcal{E}$ are irreducible.

C. Statement of main result

Theorem 1. *Let S be an irreducible constraint, presented by an irreducible deterministic graph G, and let n_0 and n_1 be positive integers. Then there exists an (S, n_0, n_1)-encoder if and only if $\mathcal{X}(A_{G_0}, n_0) \cap \mathcal{X}(A_{G_1}, n_1) \neq \emptyset$.*

The necessary and sufficient conditions in the theorem are discussed in Sections II and III, respectively. Hereafter, we use the notation $\mathcal{X}(A_{G_0}, A_{G_1}, n_0, n_1)$ for the intersection $\mathcal{X}(A_{G_0}, n_0) \cap \mathcal{X}(A_{G_1}, n_1)$.

In view of Theorem 1, finding the possible pairs (n_0, n_1) for which an $(S(G), n_0, n_1)$-encoder exists for a given G and partition $\{\Sigma_0, \Sigma_1\}$ requires a method for deciding whether $\mathcal{X}(A_{G_0}, n_0)$ and $\mathcal{X}(A_{G_1}, n_1)$ share common vectors. This decision problem can be recast as a linear programming problem, namely, deciding whether there is a *real* vector $\boldsymbol{x}$ that satisfies the following constraints:

$$\begin{array}{rcl} (A_{G_0} - n_0 I)\,\boldsymbol{x} & \geq & \mathbf{0} \\ (A_{G_1} - n_1 I)\,\boldsymbol{x} & \geq & \mathbf{0} \\ \boldsymbol{x} & \geq & \mathbf{0} \\ \mathbf{1}^{\top} \cdot \boldsymbol{x} & = & 1\,, \end{array} \tag{1}$$

where $\mathbf{0}$ and $\mathbf{1}$ stand for the all-zero and all-1 vectors. Since all the coefficients in (1) are integers, if there is a real feasible solution $\boldsymbol{x}$ then there is also a rational solution, and, therefore, there is a nonzero integer solution that satisfies the (first) three inequalities in (1). There are known polynomial-time algorithms for solving linear programming problems, such as Karmarkar's algorithm [5], but it would be interesting to find a more direct method, tailored specifically to the constraints (1), for determining whether $\mathcal{X}(A_{G_0}, A_{G_1}, n_0, n_1) \neq \emptyset$. In comparison, recall that in the context of ordinary (S, n)-encoders, the question of interest is whether $\mathcal{X}(A_G, n)$ is nonempty, which, in turn, is equivalent to asking whether $n \leq \lambda(A_G)$.

D. Going to powers of the constraint

Next, we discuss the effect of going to powers of a constraint, namely, attempting to construct (S^t, n_0, n_1)-encoders, for increasing values of t. To this end, we first need to define the even and odd symbols in Σ^t, which is the alphabet of S^t, given a partition $\{\Sigma_0, \Sigma_1\}$ of Σ. Motivated again by the parity-preserving application, we say that $\boldsymbol{w} \in \Sigma^t$ is even (resp., odd), if it contains an even (resp., odd) number of symbols from Σ_1 (i.e., the parity of $\boldsymbol{w}$ is the modulo-2 sum of the parities of the symbols in $\boldsymbol{w}$). The set of even (resp., odd) words in Σ^t will be denoted by $(\Sigma^t)_0$ (resp., $(\Sigma^t)_1$).

It turns out that in most cases, we can approach the capacity of S with parity-preserving encoders if we let t increase. Note, however, that such an increase may sometimes be *necessary*, even when the capacity of S is $\log_2(n_0{+}n_1)$ (see Example 1 below). This presents a distinction between parity-preserving encoders and ordinary ones: when $\mathsf{cap}(S) = \log_2 n$, capacity is always attained by ordinary encoders already for $t = 1$.

Specifically, we have the following result (we omit the proof due to space limitations).

Theorem 2. *Let G be a deterministic primitive[1] graph, having at least one edge with an even label and one edge with an odd label. Then there exists an infinite sequence of nonnegative integers $n^{(1)}, n^{(2)}, \cdots$ such that $(S(G^t), n^{(t)}, n^{(t)})$-encoders exist and*

$$\lim_{t\to\infty} (\log_2 n^{(t)})/t = \mathsf{cap}(S(G))\,.$$

For a deterministic graph G and a positive integer t, we denote by $n_{\max}(G, t)$ the largest integer n for which there exist $(S(G^t), n, n)$-encoders, and define the largest possible coding ratio attainable by such encoders by

$$\rho(G, t) = (1/t) \log_2 (2\, n_{\max}(G, t))\,. \tag{2}$$

Example 1. Let G be the graph in Figure 1, where $\Sigma_0 = \{a, b\}$ and $\Sigma_1 = \{c, d\}$. We have $\lambda(A_G) = 2$, and the matrices

[1] An irreducible graph is primitive if the gcd of its cycle lengths is 1.

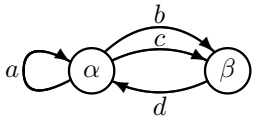

Fig. 1. Graph G for Example 1.

A_{G_0} and A_{G_1} are given by

$$A_{G_0} = \begin{pmatrix} 1 & 1 \\ 0 & 0 \end{pmatrix} \quad \text{and} \quad A_{G_1} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} .$$

It can be shown by induction on t that $\lambda(A_{(G^t)_0}) = \lambda(A_{(G^t)_1}) = 2^{t-1}$. Clearly, $\mathcal{X}(A_{(G^t)_0}, n)$ and $\mathcal{X}(A_{(G^t)_1}, n)$ are empty if and only if $n > 2^{t-1}$. Their intersection, however, turns out to be empty also when $n = 2^{t-1}$. On the other hand, for $n^{(t)} = 2^{t-1}-1$, we do have $(2\ 1)^\top \in \mathcal{X}(A_{(G^t)_0}, A_{(G^t)_1}, n^{(t)}, n^{(t)})$. So, in this case, $\rho(G,t) = (1/t)\log_2(2^t-2)$ (< 1): while we can *approach* the capacity value of 1 as t increases, we cannot actually achieve it by rate $t : t$ parity-preserving encoders.

In contrast, there is a very simple one-state *variable-rate* parity-preserving encoder with coding ratio 1 for $S(G)$: just map the input 0 to a (at rate $1 : 1$) and the inputs 10 and 11 to bd and cd, respectively (at rate $2 : 2$). □

Theorem 2 focused on (S, n_0, n_1)-encoders where $n_0 = n_1$. While this case suits the motivation of parity-preserving encoders (where $n_0 = n_1 = 2^{p-1}$), there seems to be a merit in studying the more general case as well (see Example 2 in Section III-B).

II. Necessary condition

Our proof of the "only if" part of Theorem 1 is a refinement of the proof of Theorem 3 in [6], where it was shown, *inter alia*, that the existence of an (S, n)-encoder implies that $\mathcal{X}(A_G, n) \neq \emptyset$. Since the existence of an (S, n_0, n_1)-encoder implies the existence of $(S(G_\mathsf{b}), n_\mathsf{b})$-encoders for $\mathsf{b} \in [2\rangle$, it also implies that $\mathcal{X}(A_{G_0}, n_0)$ and $\mathcal{X}(A_{G_1}, n_1)$ are both nonempty sets (and, so, $n_0 \leq \lambda(A_{G_0})$ and $n_1 \leq \lambda(A_{G_1})$). Theorem 1 implies that their *intersection* must be nonempty too. We omit the proof due to space limitations.

The next two corollaries (and their proofs) parallel Corollary 1 and Theorem 5 in [6].

Corollary 3. *Let S, G, n_0, and n_1 be as in Theorem 1. Then, for any (S, n_0, n_1)-encoder $\mathcal{E}$,*

$$|V(\mathcal{E})| \geq \min_{\boldsymbol{x} \in \mathcal{X}(A_{G_0}, A_{G_1}, n_0, n_1)} \|\boldsymbol{x}\|_\infty ,$$

where $\|(x_u)_u\|_\infty = \max_u x_u$.

Corollary 4. *With S, G, n_0, n_1, and $\mathcal{E}$ as in Corollary 3,*

$$\mathcal{A}(\mathcal{E}) \geq \log_n \Big(\min_{\boldsymbol{x} \in \mathcal{X}(A_{G_0}, A_{G_1}, n_0, n_1)} \|\boldsymbol{x}\|_\infty \Big) ,$$

where $n = \max\{n_0, n_1\}$.

The Franaszek algorithm is a known method for computing approximate eigenvectors [6, Sec. IX]. Figure 2 presents a modification of it for computing a vector in $\mathcal{X}(A_0, A_1, n_0, n_1)$, where A_0 and A_1 are nonnegative integer $k \times k$ matrices (a nonnegative integer k-vector $\boldsymbol{\xi}$ is provided as an additional parameter to the algorithm). The modified algorithm can be used to compute the lower bounds of Corollaries 3 and 4, and will turn out to be useful also when designing parity-preserving encoders.

By slightly generalizing the proof of validity of the (ordinary) Franaszek algorithm (see [6, Sec. IX]) it follows that the algorithm in Figure 2 returns the largest (componentwise) vector $\boldsymbol{x} \in \mathcal{X}(A_0, A_1, n_0, n_1)$ that satisfies $\boldsymbol{x} \leq \boldsymbol{\xi}$; if no such vector exists, then the algorithm returns $\mathbf{0}$.

```
y ← ξ;
x ← 0;
while (x ≠ y)    {
      x ← y;
      y ← min{⌊(1/n0)A0x⌋, ⌊(1/n1)A1x⌋, x};
            /* apply ⌊·⌋ and min{·,·} componentwise */
}
return x;
```

Fig. 2. Modified Franaszek algorithm.

By analyzing the complexity of Karmarkar's algorithm [5], one can infer an upper bound on the largest entry of the output of the algorithm in Figure 2 (in terms of its input parameters). It is still open whether such a bound can be obtained in a more straightforward manner.

III. Sufficient condition

We start proving the "if" part in Theorem 1 by considering two special cases in Sections III-A and III-B. We then turn to the general case in Section III-C.

A. Deterministic encoders

If $\mathcal{X}(A_{G_0}, A_{G_1}, n_0, n_1)$ contains a 0–1 vector, then a subgraph of G is an $(S(G), n_0, n_1)$-encoder with anticipation 0. By Corollary 4, the existence of such a vector is also a necessary condition for having a deterministic $(S(G), n_0, n_1)$-encoder. If, in addition, G has finite memory μ, then the resulting encoder is $(\mu, 0)$-definite[2] and, therefore, $(\mu, 0)$-sliding-block decodable for any tagging.

B. Encoders with anticipation 1 obtained by state splitting

Suppose now that $\mathcal{X}(A_{G_0}, A_{G_1}, n_0, n_1)$ does not contain a 0–1 vector, yet contains a vector $\boldsymbol{x} = (x_u)_u$ such that for each $\mathsf{b} \in [2\rangle$, an application of one $\boldsymbol{x}$-consistent state splitting round[3] to G_b results in an all-1 induced approximate eigenvector. For $\mathsf{b} \in [2\rangle$, let $\hat{\mathcal{E}}_\mathsf{b}$ be the resulting $(S(G_\mathsf{b}), n_0, n_1)$-encoder. Note that each state $u \in V(G)$ is transformed into x_u descendant states in each encoder $\hat{\mathcal{E}}_\mathsf{b}$; denote those states by $(u, i)_\mathsf{b}$, where $i \in [x_u\rangle$ (the order implied by the index i on the descendant states of a given u can be arbitrary).

[2] A graph is (m, a)-definite if all paths that generate a given word of length $m+a+1$ share the same $(m+1)$st edge. An encoder is (m, a)-sliding-block decodable if these paths share the same tag on their $(m+1)$st edges.

[3] Refer to [7, Ch. 5] for the description of the state-splitting algorithm and for the related terms used here.

Next, construct the following graph $\mathcal{E}$:

$$V(\mathcal{E}) = \{(u,i) \ : \ u \in V(G) \quad \text{and} \quad i \in [x_u\rangle\} \ ,$$

and endow $\mathcal{E}$ with an edge $(u,i) \xrightarrow{a} (v,j)$ if and only if for some $\mathsf{b} \in [2\rangle$, the encoder $\hat{\mathcal{E}}_{\mathsf{b}}$ contains an edge $(u,i)_{\mathsf{b}} \xrightarrow{a} (v,j)_{\mathsf{b}}$. It follows from the construction that $\mathcal{E}_{\mathsf{b}} = \hat{\mathcal{E}}_{\mathsf{b}}$ for $\mathsf{b} \in [2\rangle$. In particular, from each $\mathcal{E}$-state there are n_{b} outgoing edges with labels from Σ_{b}, for each $\mathsf{b} \in [2\rangle$. Moreover, it can be readily seen that every word that can be generated from state (u,i) in $\mathcal{E}$ can also be generated from the parent G-state u in G. Finally, $\mathcal{E}$ has anticipation 1: if a word $w_1 w_2$ is generated in $\mathcal{E}$ by a path π from (u,i) in $\mathcal{E}$ then u and the symbol w_1 uniquely identify the parent G-state, v, of the terminal state of the first edge in π, and the symbol w_2 then uniquely identifies the particular descendant state (v,j) of v in which that edge terminates. Furthermore, if G has finite memory μ, then $\mathcal{E}$ is $(\mu,1)$-definite and, therefore, $(\mu,1)$-sliding-block decodable for any tagging.

Example 2. We consider the 16th power of the $(2,10)$-RLL constraint, as found in the DVD standard [7, §1.7.3 and Example 5.7]. Let G be the 11-state graph presenting that power and let Σ_0 (resp., Σ_1) be the set of all 16-bit words of even (resp., odd) parity that satisfy the $(2,10)$-RLL constraint. Running the algorithm in Figure 2 with $A_0 = A_{G_0}$, $A_1 = A_{G_1}$, and $\boldsymbol{\xi} = 2 \cdot \mathbf{1}$ yields the result

$$\boldsymbol{x} = (1\,1\,2\,2\,2\,2\,1\,1\,1\,1\,0)^\top \ ,$$

for any $n_0 \leq 173$ and $n_1 \leq 178$ (running the algorithm with larger values of n_0 or n_1 yields the all-zero vector). This is also the vector obtained when running the (ordinary) Franaszek algorithm with $A_G = A_{G_0} + A_{G_1}$, $n = 351$, and $\boldsymbol{\xi} = 2 \cdot \mathbf{1}$. In both G_0 and G_1 we can merge states to yield G_0' and G_1' with four states each, and the respective $(A_{G_0'}, A_{G_1'}, n_0{=}173, n_1{=}178)$-approximate eigenvector is

$$\boldsymbol{x}' = (1\,1\,2\,1)^\top \ .$$

Both G_0' and G_1' can be split in one round consistently with $\boldsymbol{x}'$, resulting in the all-1 induced vector and, therefore, in an $(S(G), 173, 178)$-encoder. The out-degree of the encoder used in the DVD is $2^p = 256$, where the set of input tags consists of all 8-bit tuples. Some of the input tags can be mapped to two possible 16-bit words with different parities[4], while the rest are mapped to unique 16-bit words. □

C. Construction using the stethering method

The technique used in Section III-B does not seem to generalize easily if the conditions therein—namely, being able to split G_0 and G_1 in one round and ending up with an all-1 induced approximate eigenvector—do not hold. In fact, due to the fact that the matrices G_0 and G_1 may be reducible, there are examples that show that we may get stuck while attempting to split them. Moreover, we have found an example (which we omit) where multiple rounds of state splitting are required, which do end up with an all-1 approximate eigenvector, yet there is no way one can match the descendant states in $\mathcal{E}_0$ of a given G-state with the respective descendant states in $\mathcal{E}_1$ while maintaining finite anticipation.

Recognizing that the finite anticipation property is not guaranteed even when the state-splitting algorithm is used (at least in the manner we employed this algorithm in Section III-B), we resort to a more general framework of designing encoders, which includes the state-splitting algorithm and the stethering design method of [2] as special cases (see also [1] and [7, §6.2]). As we see, it will be rather easy to adapt the stethering method to design parity-preserving encoders, even though finite anticipation can be guaranteed only under certain conditions.

Next we recall the stethering method, while tailoring it to our setting. Let G be a deterministic graph and $\{\Sigma_0, \Sigma_1\}$ be a partition of its label alphabet Σ, and let $\boldsymbol{x} = (x_u)_{u \in V(G)}$ be in $\mathcal{X}(A_{G_0}, A_{G_1}, n_0, n_1)$. We assume that $\boldsymbol{x} > \mathbf{0}$, or else remove the zero-weight states from G. For $u \in V(G)$, denote by $\Sigma_{\mathsf{b}}(u)$ the set of symbols from Σ_{b} that label edges outgoing from u. For $u \in V(G)$ and $a \in \Sigma_{\mathsf{b}}(u)$, denote by $\tau(u;a)$ the terminal G-state of the unique edge outgoing from u with label a.

For $\mathsf{b} \in [2\rangle$ and $u \in V(G)$, let

$$\Delta_{\mathsf{b}}(u) = \big\{(a,j) \ : \ a \in \Sigma_{\mathsf{b}}(u) \quad \text{and} \quad j \in \big[x_{\tau(u;a)}\big\rangle\big\} \ .$$

Since $\boldsymbol{x} \in \mathcal{X}(A_{G_{\mathsf{b}}}, n_{\mathsf{b}})$ we have $|\Delta_{\mathsf{b}}(u)| = (A_{G_{\mathsf{b}}}\boldsymbol{x})_u \geq n_{\mathsf{b}} x_u$. Thus, we can partition (a subset of) $\Delta_{\mathsf{b}}(u)$ into x_u subsets

$$\Delta_{\mathsf{b}}^{(0)}(u), \ \Delta_{\mathsf{b}}^{(1)}(u), \ \ldots, \ \Delta_{\mathsf{b}}^{(x_u-1)}(u) \ , \tag{3}$$

such that $|\Delta_{\mathsf{b}}^{(i)}(u)| = n_{\mathsf{b}}$ for each i. We fix such a partition and construct the following graph $\mathcal{E}$:

$$V(\mathcal{E}) = \{(u,i) \ : \ u \in V(G) \quad \text{and} \quad i \in [x_u\rangle\} \ ,$$

and for each $\mathsf{b} \in [2\rangle$, $u \in V(G)$, $i \in [x_u\rangle$, and $(a,j) \in \Delta_{\mathsf{b}}^{(i)}(u)$, we endow $\mathcal{E}$ with an edge $(u,i) \xrightarrow{a} (\tau(u;a), j)$.

Proposition 5. *The graph $\mathcal{E}$ is an $(S(G), n_0, n_1)$-encoder.*

We omit the proof (the respective proof for ordinary encoders is essentially contained in [1]).

The number of states of the constructed encoder $\mathcal{E}$ (before any possible merging of states) equals the sum, $\|\boldsymbol{x}\|_1$, of the entries of $\boldsymbol{x}$. Thus, with this construction, we can obtain an encoder $\mathcal{E}$ such that

$$|V(\mathcal{E})| \leq \min_{\boldsymbol{x} \in \mathcal{X}(A_{G_0}, A_{G_1}, n_0, n_1)} \|\boldsymbol{x}\|_1$$

(compare with the lower bound of Corollary 3).

In stethering encoders, the subsets in (3) have a particular structure which we describe next. For $\mathsf{b} \in [2\rangle$ and $u \in V(G)$, assume some ordering on the elements of $\Sigma_{\mathsf{b}}(u)$. For $a \in \Sigma_{\mathsf{b}}(u)$, define $\phi_{\mathsf{b}}(u;a)$ by

$$\phi_{\mathsf{b}}(u;a) = \textstyle\sum_{b \in \Sigma_{\mathsf{b}}(u) \,:\, b < a} x_{u_b} \ ,$$

and for $i \in [x_u\rangle$, let

$$\Delta_{\mathsf{b}}^{(i)}(u) = \big\{(a,j) \ : \ a \in \Sigma_{\mathsf{b}}(u) \ , \ \ j \in \big[x_{\tau(u;a)}\big\rangle \ , \\ \text{and} \ \ i\, n_{\mathsf{b}} \leq \phi_{\mathsf{b}}(u;a) + j < (i{+}1) n_{\mathsf{b}}\big\} \ . \tag{4}$$

[4] There can be at most $173 + 178 - 256 = 95$ input bytes of this type, but in practice their number is slightly smaller.

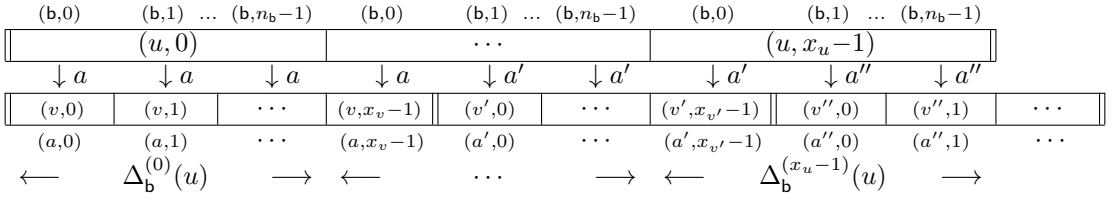

Fig. 3. Descendants of a G-state u in a subgraph $\mathcal{E}_\mathsf{b}$ of a stethering encoder.

This construction is illustrated in Figure 3, for a given G-state u and parity $\mathsf{b} \in [2\rangle$. The boxes in the top row in the figure represent the "descendants" of state u, namely, the states (u,i), for $i \in [x_u\rangle$, and the width of each box in the top row is one unit. The outgoing edges from each state (u,i) are shown as downward arrows, along with their labels, and an assignment of input tags, $(\mathsf{b},0),(\mathsf{b},1),\ldots,(\mathsf{b},n_\mathsf{b}-1)$, is shown above the top row. The boxes at the bottom row are $1/n_\mathsf{b}$ units wide and represent the terminal states of the edges. The respective elements of $\Delta_\mathsf{b}(u)$ are written just below the bottom row, where we have also shown their grouping into the subsets (3) defined by (4). The double vertical lines group the edges according to their labels. So, for example, according to the figure, there is an outgoing edge labeled a' and tagged by $(\mathsf{b},0)$ from (u,x_u-1) to $(v',x_{v'}-1)$, and that edge corresponds to the element $(a',x_{v'}-1) \in \Delta_\mathsf{b}(u)$.

Stethering encoders can have finite anticipation (and be sliding-block decodable if G has finite memory), provided that there is sufficient margin between the target encoder rate $p:q$ and the maximal coding ratio $\rho(G,q)$ (as defined in (2)). Specifically, suppose that $\boldsymbol{x} \in \mathcal{X}(A_{G_0},A_{G_1},n_0+1,n_1+1)$ (namely, we assume even and odd out-degrees larger by 1 than targeted). Using $\boldsymbol{x}$, we first construct a stethering $(S(G),n_0+1,n_1+1)$-encoder $\mathcal{E}^*$ and assign the input tags $(\mathsf{b},0),(\mathsf{b},1),\ldots,(\mathsf{b},n_\mathsf{b})$ to the outgoing edges from each state, as in Figure 3. Then, from $\mathcal{E}^*$ we form a punctured $(S(G),n_0,n_1)$-encoder $\mathcal{E}$ by deleting all edges in $\mathcal{E}$ tagged by either $(0,n_0)$ or $(1,n_1)$.

We have the following result (compare the guaranteed upper bound on $\mathcal{A}(\mathcal{E})$ to the lower bound in Corollary 4).

Theorem 6. *Let G be a deterministic graph and let n_0 and n_1 be positive integers such that $\mathcal{X}(A_{G_0},A_{G_1},n_0+1,n_1+1) \neq \emptyset$. Then, there is an $(S(G),n_0,n_1)$-encoder $\mathcal{E}$, obtained by the (punctured) stethering method, such that $\mathcal{A}(\mathcal{E}) \leq a$, where*

$$a = 1 + \min_{\boldsymbol{x}\in\mathcal{X}(A_{G_0},A_{G_1},n_0+1,n_1+1)} \left\{ \lceil \log_{n+1} \|\boldsymbol{x}\|_\infty \rceil \right\}$$

and $n = \min\{n_0,n_1\}$. Furthermore, if G has finite memory μ, then $\mathcal{E}$ is (μ,a)-definite, and hence any tagged $(S(G),n_0,n_1)$-encoder based on $\mathcal{E}$ is (μ,a)-sliding-block decodable.

The proof is essentially the same as that of Proposition 3 in [2], with modifications to handle the parity-preserving setting. We omit the details.

Recall that $n_{\max}(G,q)$ is the largest integer n for which $(S(G^q),n,n)$-encoders exist. If we use the punctured stethering method to construct rate $p:q$ parity-preserving encoders, then we need to have $2^p+1 \leq n_{\max}(G,q)$. This inequality is satisfied whenever

$$\frac{p}{q} \leq \frac{\log_2 n_{\max}(G,q)}{q} - \frac{\log_2(1+2^{-p})}{q} ,$$

which, in turn, is satisfied whenever

$$\frac{p}{q} \leq \rho(G,q) - \frac{\log_2 e}{2^p q}$$

(see (2)). We conclude that finite anticipation (and sliding-block decodability when G has finite memory) can be guaranteed with a rate penalty of (no more than) $(\log_2 e)/(2^p q)$.

It is still an open problem whether finite anticipation can be guaranteed whenever $\mathcal{X}(A_{G_0},A_{G_1},n_0,n_1) \neq \emptyset$.

References

[1] R.L. Adler, L.W. Goodwyn, B. Weiss, "Equivalence of topological Markov shifts," *Israel J. Math.,* 27 (1977), 49–63.

[2] J.J. Ashley, B.H. Marcus, R.M. Roth, "Construction of encoders with small decoding look-ahead for input-constrained channels," *IEEE Trans. Inf. Theory,* 41 (1995), 55–76.

[3] K.A.S. Immink, *Codes for Mass Data Storage Systems,* Second Edition, Shannon Foundation Publishers, Eindhoven, The Netherlands, 2004.

[4] J.A.H.M. Kalhman, K.A.S. Immink, "Device for encoding/decoding N-bit source words into corresponding M-bit channel words, and vice versa," US Patent 5,477,222, 1995.

[5] D.G. Luenberger, Y. Ye, *Linear and Nonlinear Programming,* Springer, New York, 2008.

[6] B.H. Marcus, R.M. Roth, "Bounds on the number of states in encoder graphs for input-constrained channels," *IEEE Trans. Inf. Theory,* 37 (1991), 742–758.

[7] B.H. Marcus, R.M. Roth, P.H. Siegel, *An Introduction to Coding for Constrained Systems,* Lecture Notes (available online), 2001.

[8] T. Miyauchi, Y. Shinohara, Y. Iida, T. Watanabe, Y. Urakawa, H. Yamagishi, M. Noda, "Application of turbo codes to high-density optical disc storage using 17PP Code," *Jpn. J. Appl. Phys.,* 44 No. 5B (2005), 3471-3473.

[9] T. Narahara, S. Kobayashi, M. Hattori, Y. Shimpuku, G.J. van den Enden, J.A.H.M. Kahlman, M. van Dijk, R. van Woudenberg, "Optical disc system for digital video recording," *Jpn. J. Appl. Phys.,* 39 No. 2B (2000), 912-919.

[10] M. Noda, H. Yamagishi, "An 8-state DC-controllable run-length-limited code for the optical-storage channel," *Jpn. J. Appl. Phys.,* 44 No. 5B (2005), 3462-3466.

[11] W.Y.H. Wilson, K.A.S. Immink, X.B. Xi, C.T. Chong, "A Comparison of two coding schemes for generating DC-free runlength-limited sequences," *Jpn. J. Appl. Phys.,* 39 No. 2B (2000), 815-818.