Construction of Encoders with Small Decoding Look-Ahead for Input-Constrained Channels

Jonathan J. Ashley* Brian H. Marcus* Ron M. Roth†

Abstract

An input-constrained channel is defined as the set S of finite sequences generated by a finite labeled directed graph which defines the channel. A construction based on a result of Adler, Goodwyn, and Weiss is presented for finite-state encoders for input-constrained channels. Let $G = (V, E)$ denote a smallest deterministic presentation of S. For a given input-constrained channel S and for any rate $p : q$ up to the capacity $c(S)$ of S, the construction provides finite-state encoders of fixed-rate $p : q$ that can be implemented in hardware with a number of gates which is at most polynomially large in $|V|$. When $p/q < c(S)$, the encoders have order $\leq 12|V|$, namely, they can be decoded by looking ahead at up to $12|V|$ symbols, thus improving slightly on the order of previously-known constructions. Furthermore, when $p/q \leq c(S) - ((\log_2 e)/(2^p q))$ and S is of finite memory, the encoders can be decoded by a sliding-block decoder with look-ahead $\leq 2|V| + 1$.

Key words: Input-constrained channel; Sliding-block decoding; State-splitting algorithm.

*IBM Research Division, Almaden Research Center, 650 Harry Road, San Jose, CA 95120.

†Computer Science Department, Technion — Israel Institute of Technology, Haifa 32000, Israel. Part of this work was done while visiting IBM Research Division, Almaden Research Center, San Jose, CA. This research was supported in part by the United-States—Israel Binational Science Foundation.

1 Introduction

Input-constrained channels, also known as constrained systems, are widely-used models for describing the read-write requirements of secondary storage systems, such as magnetic disks or optical memory devices [15][22][25]. A constrained system S is defined by means of a finite labeled directed graph, the paths of which identify the set S of *constrained sequences* allowed to be written into the device. The finite set of symbols appearing on the edges of the presenting labeled graph is called the *constrained-system alphabet* and is denoted hereafter by Σ. An example of a constrained system is the (d, k)-run-length-limited (RLL) system, defined as the set of binary words in which any two consecutive 1's are separated by runs of zeros of lengths ranging between d and k.

One of the major goals in the study of constrained systems is designing encoders that map unconstrained sequences, referred to as *source sequences,* over a finite set Φ of *source tags,* into constrained sequences of a given constrained system S [1][2][10]–[13][14][17][20]. Let S be a constrained system and let Φ be a given set of source tags (e.g., $\Phi = \{0, 1\}$). A *fixed-rate $p : q$ encoder over Φ into S* is a Mealy-type finite-state machine $\mathcal{M}$ [19, p. 323] defined by a finite set of states V, an output function $\Gamma : V \times \Phi^p \to \Sigma^q$, and a transfer (or next-state) function $T : V \times \Phi^p \to V$, such that the following holds for all $r = 0, 1, 2, \ldots$:

- at each time slot r, the machine $\mathcal{M}$, being at some state $v_r \in V$, receives as input a p-tuple $\mathbf{s}_{r+1}$ of source tags over Φ, generates a q-tuple $\mathbf{w}_{r+1} = \Gamma(v_r, \mathbf{s}_r)$ of symbols over the constrained-system alphabet Σ, and transfers into state $v_{r+1} = T(v_r, \mathbf{s}_r)$;
- the sequences $\mathbf{w}_1\mathbf{w}_2 \ldots \mathbf{w}_r$ obtained by concatenating the q-tuples generated by $\mathcal{M}$ are constrained sequences of S;
- $\mathcal{M}$ is lossless: knowing the initial state v_0, the final state v_r, and the channel sequence $\mathbf{w}_1\mathbf{w}_2 \ldots \mathbf{w}_r$ generated in between by $\mathcal{M}$, allows to reconstruct the source sequence $\mathbf{s}_1\mathbf{s}_2 \ldots \mathbf{s}_r$ which was input to $\mathcal{M}$.

Note that losslessness is a necessary condition we must impose on encoders in order to allow unique decodability. Practical considerations dictate additional requirements from encoders. For example, in virtually all applications, we require on-line decodability, in which

case the decoder ought to be a finite-state machine that reconstructs the source sequence, possibly delayed by a prescribed number of time slots. On-line decodability is equivalent to requiring that the encoder $\mathcal{M}$ be lossless of finite order, namely, that there is an integer N such that the encoder state at each time slot r, with the q-tuples over Σ generated at times $r, r+1, \ldots, r+N-1$, determine uniquely the p-tuple over Φ that was input at time slot r. The smallest number N for which such a reconstruction is possible is called the *order* of $\mathcal{M}$. The order of an encoder determines the smallest delay with which any decoder for such an encoder will reconstruct the source sequence.

An additional requirement that the decoder be sliding-block is added in many cases to avoid catastrophic propagation of errors and to simplify the decoder structure. In such a decoding scheme, the source sequence which was input to the encoder is reconstructed by applying a decoding function on a 'window' of symbols in the constrained sequence. At each time slot, the window is slid by q symbols along the constrained sequence, to reconstruct successively the p-tuples of tags of the source sequence. A sliding-block decoder is characterized by two parameters, namely, the number a of *look-ahead* q-tuples of Σ that are included in such a window and the number m of *look-behind* q-tuples in the sliding window. The length of the window thus totals to $m + a + 1$.

The length of a window of a sliding-block decoder [5][16] is one example among many others for efficiency requirements. Other examples of design parameters may be the number of states in a finite-state implementation of encoders; or, for more practical aspects, the number of gates in a hardware implementation; or the time and space complexity of a program on a random-access machine (RAM) [4, Ch. 1] if software implementation is desired.

Several schemes have been suggested in the literature for constructing finite-state encoders, most notable of which are Franaszek's methods [10]–[13], Lempel and Cohn's [20], and the Adler-Coppersmith-Hassner algorithm, also known as the state-splitting algorithm [1]. In addition to generating finite-state encoders with 'almost' the smallest number of states [21], the state-splitting algorithm provides encoders which are lossless of finite-order and, thus, on-line decodable. Furthermore, for the important subclass of constrained systems of finite memory, the resulting encoders are sliding-block decodable. Improvements and generalizations of the state-splitting algorithm can be found in [2][14][17]. See [22] for a tutorial on the subject.

In order to measure the complexity of encoders, we need to set some canonical presentation of the constrained system S. An example of such a presentation is a deterministic labeled graph G with the smallest number of states. So far, there hasn't been found any 'small' (e.g., polynomial in the size of G) upper bound on the number of gates required in hardware implementation of encoders generated by the state-splitting algorithm. Furthermore, the only known general upper bound on the order (and also on the length of a window of a sliding-block decoder) of a fixed-rate $p : q$ encoder over Φ into S generated by the state-splitting algorithm may be *exponentially* large. This bound is related to the sum of components of a so-called approximate eigenvector associated with G, and there are examples where this bound is exponential in the number k of states in G [5][21].

In [5], Ashley obtained a construction of encoders with order which is *linear* in the number k of states in G by applying the state-splitting algorithm on a *power* of the labeled graph G; the encoder thus obtained has rate $p\,t : q\,t$ for some $t = O(k)$. When the construction in [5] is translated into a $p : q$ encoder, we still end up with an encoder with order $O(k)$, but typically this encoder is not sliding-block decodable anymore: instead of having one decoding function, the decoder has t decoding functions, one for each time slot within a period of t time slots. Also, the number of encoder states tends to be much larger than the minimum.

In Section 3 we present a method for constructing fixed-rate encoders which we will refer to as *stethering* encoders. The stethering method is a modification of a construction which was used by Adler, Goodwyn, and Weiss in [3] to show that certain topological Markov shifts with the same capacity are equivalent in a very strong sense. The number of states in the encoders obtained in this way is 'almost' the smallest possible: it does not exceed the known general upper bound on the number of states obtained by the state-splitting algorithm. Based on complexity results, we show in Section 4 that the number of gates and memory-cells required for a hardware implementation of stethering encoders is at most polynomial in q, $\log|\Sigma|$, and the number k of states of the presentation G.

Then, in Section 5 we show that for any constrained system S and positive integers p and q such that $(p/q)\log|\Phi| < c(S)$, the stethering method, applied to a power of the graph G, provides encoders with order $\leq 12k$ (hereafter all logarithms are taken to base 2). We point out that the construction in [5] also gives encoders with order at most linear in k (this was not made explicit in [5]). The order is slightly smaller in the stethering

construction. A considerably smaller bound on the order is obtained when $(p/q)\log|\Phi| \leq c(S) - ((\log e)/(|\Phi|^p q))$, where e stands for the base of natural logarithms: for this range of rates we obtain encoders with order $\leq 2k+2$. If, in addition, S is of finite memory, then the resulting encoders can be decoded by sliding-block decoders with look-ahead $\leq 2k+1$. To the best of our knowledge, the stethering method is the first general construction method that can be applied to a fairly wide range of rates to yield sliding-block decodable encoders with linear look-ahead.

The case $(p/q)\log|\Phi| = c(S)$ is covered in Section 6, where we present a variation of the construction of [5] as a stethering encoder obtained from a power of the presenting graph G. The resulting encoder has a polynomial-size implementation and order which is at most $9k + 6\lceil(\log k)/(p\log|\Phi|)\rceil$.

The properties of all encoder constructions discussed in this work are summarized in Section 7. In Section 8 we present some extensions of the stethering construction, and within this framework we compare losslessness of finite order and sliding-block decodability. Finally, we discuss some limitations of the stethering method in Section 9.

2 Background

We follow the definitions and notations as in [21]. A *labeled graph* $G = (V, E, L)$ is a finite directed graph with set of states $V = V_G$, set of edges $E = E_G$, and a labeling $L : E \to \Sigma$ for some finite alphabet Σ.

A *constrained system* is the set of all words (i.e., finite sequences) obtained from reading the labels of paths in a labeled graph G. We call the set of sequences $S(G)$, and we say that G presents $S = S(G)$. If a path is labeled by a word $\mathbf{w}$, we say that the path generates $\mathbf{w}$.

Let G be a labeled graph, $S = S(G)$ and q a positive integer. Denote by G^q the labeled graph which has the same states as G, has edges from u to v for each path of length q from u to v in G, and labels inherited from the labels of these paths. The constrained system S^q is defined as $S(G^q)$ and it consists of all words of S of lengths divisible by q.

A labeled graph G is *deterministic* if for each state $v \in V$ the outgoing edges from v

are distinctly labeled. Every constrained system has a deterministic presentation. A labeled graph G is *lossless* if any two distinct paths with the same initial state and terminal state have different labelings.

Let S be a constrained system and n be a positive integer. An (S, n)-*encoder* is a labeled graph $\mathcal{E}$ such that —

- each state of $\mathcal{E}$ has out-degree n;
- $S(\mathcal{E}) \subseteq S$;
- $\mathcal{E}$ is lossless.

Note that a finite-state machine $\mathcal{M}$ is a fixed-rate $p : q$ encoder over Φ into S if and only if the state diagram of $\mathcal{M}$ (with the source tags ignored) is an $(S^q, |\Phi|^p)$-encoder.

A labeled graph G is *irreducible (or strongly connected)* if for every pair of states (u, v) in G there is a path from u to v. A constrained system S is irreducible if it can be presented by an irreducible graph. For an irreducible constrained system there is a unique deterministic graph presentation with the smallest number of states, and such a presentation is irreducible [8][9].

Given a labeled graph G, the adjacency matrix $A_G = \left[(A_G)_{u,v}\right]$ is the $|V_G| \times |V_G|$ matrix with $(A_G)_{u,v}$ being the number of edges from state u to state v. Note that $(A_G)^q = A_{G^q}$.

For a square real matrix A, denote by $\lambda(A)$ the largest of the absolute values of the eigenvalues of A. If A is nonnegative then, by the Perron-Frobenius Theorem [23, p. 14]), $\lambda(A)$ is an eigenvalue of A. Furthermore, there are nonnegative left and right eigenvectors associated with $\lambda(A)$.

The *capacity* $c(S)$ of a constrained system S is the asymptotic growth rate of the number of words in S i.e.,

$$c(S) = \lim_{n\to\infty} \frac{1}{n} \log \left| \text{ words of length } n \text{ in } S \right| .$$

When G is a lossless labeled graph, we have $c(S) = \log \lambda(A_G)$. We assume that $c(S) > 0$, equivalently $\lambda(A_G) > 1$.

Given a square nonnegative integer matrix A and an integer n, an (A, n)-*approximate eigenvector* is a nonnegative integer vector $\mathbf{x} \neq \mathbf{0}$ such that $A\mathbf{x} \geq n\mathbf{x}$, where the (weak)

inequality holds component-by-component. The set of all such vectors is denoted by $\mathcal{X}(A, n)$. It is known that $\mathcal{X}(A, n) \neq \emptyset$ if and only if $\lambda(A) \geq n$ [26, Lemma 2.2]. For a vector $\mathbf{x} = [x_u]_u$ we use the notations $\|\mathbf{x}\|_1$ and $\|\mathbf{x}\|_\infty$ for $\sum_u |x_u|$ and $\max_u |x_u|$, respectively. The number of nonzero components in $\mathbf{x}$ will be denoted by weight($\mathbf{x}$).

We say that G is lossless of finite order if there is an integer N such that any two paths of length N with the same initial state and labeling must have the same initial edge. The order $\mathcal{O}(G)$ of G is the smallest N for which this holds. In case G is not lossless of finite order we define $\mathcal{O}(G) = \infty$. The following lower bounds on the number of states and order of any (S, n)-encoder were proved in [21].

Theorem 1. *Let S be a constrained system presented by a deterministic labeled graph G and let n be a positive integer. Assume that $c(S) \geq \log n$. Then, for any (S, n)-encoder $\mathcal{E}$,*

$$\begin{aligned} &(i) \quad |V_{\mathcal{E}}| \geq \min_{\mathbf{x} \in \mathcal{X}(A_G, n)} \|\mathbf{x}\|_\infty , \quad \text{and} — \\ &(ii) \quad \mathcal{O}(\mathcal{E}) \geq 1 + \min_{\mathbf{x} \in \mathcal{X}(A_G, n)} \left\{ \frac{\log \|\mathbf{x}\|_\infty}{\log n} \right\} . \end{aligned}$$

On the other hand, the state-splitting algorithm yields the following upper bounds [1].

Theorem 2. *Let S be a constrained system presented by a deterministic labeled graph G and let n be a positive integer. Assume that $c(S) \geq \log n$. Then, there exists an (S, n)-encoder $\mathcal{E}$ such that,*

$$\begin{aligned} &(i) \quad |V_{\mathcal{E}}| \leq \min_{\mathbf{x} \in \mathcal{X}(A_G, n)} \|\mathbf{x}\|_1 , \quad \text{and} — \\ &(ii) \quad \mathcal{O}(\mathcal{E}) \leq \min_{\mathbf{x} \in \mathcal{X}(A_G, n)} \Big\{ \|\mathbf{x}\|_1 - \text{weight}(\mathbf{x}) \Big\} . \end{aligned}$$

The upper bound of Theorem 2(i) is at most $|V_G|$ times the lower bound of Theorem 1(i) (and it is known that both bounds are not tight in many cases). Such a gap should not be considered as too significant, since, in hardware implementation, the number of memory cells required to code an encoder state is logarithmic in the number of states. An additive term $\log |V_G|$ in the memory-cell count, in turn, becomes negligible especially in those constrained systems for which the number of encoder states is exponential in $|V_G|$ [5][21].

On the other hand, the gap between the order guaranteed by Theorem 2(ii) and the lower bound of Theorem 1(ii) is more substantial. This gap was considerably reduced in [5], where a construction of (S,n)-encoders $\mathcal{E}$ was presented with $\mathcal{O}(\mathcal{E}) \leq 15|V_G| + 3\lceil(\log|V_G|)/\log n\rceil$ (and $\mathcal{O}(\mathcal{E}) \leq 9|V_G| + 6\lceil(\log|V_G|)/\log n\rceil$ if $\log n = c(S)$). The construction to be shown in this work guarantees even smaller order when $\log n < c(S)$.

We now turn to the stronger property of encoders we referred to in Section 1, namely that of sliding-block decodability.

Let $\mathcal{E}$ be an (S,n)-encoder. For each state $v \in V_{\mathcal{E}}$ we assign distinct source tags to the outgoing edges from v; each tag is an integer between 0 and $n-1$. The encoding process is carried out by regarding $\mathcal{E}$ as a state diagram of a finite-state machine i.e., by reading the labeling of a path defined by some initial state v and an input sequence of source tags. Given a tagged encoder $\mathcal{E}$ and nonnegative integers m and a, we say that $\mathcal{E}$ is (m,a)-*sliding-block decodable* if for any $(m+a+1)$-symbol word $\mathbf{w} = w_{-m}w_{-m+1}\ldots w_{-1}w_0w_1\ldots w_a$ over Σ generated by $\mathcal{E}$ (starting from any state), the tag at $r=0$ is uniquely defined. In other words, $\mathcal{E}$ is sliding-block decodable if it can be decoded by a sliding-block decoder with look-ahead a and look-behind m, resulting in a window length $m+a+1$. For any tagged (S,n)-encoder $\mathcal{E}$, each state of which having at least one incoming edge, the required look-ahead for decoding is bounded from below by $\mathcal{O}(\mathcal{E})-1$ [21].

A deterministic graph G is said to have *finite memory* if there is an integer N such that any two paths of length N with the same labeling terminate at the same state of G. The smallest N for which this holds is called the *memory* of G and is denoted $\mu(G)$. If G has finite memory, then $\mu(G) \leq \frac{1}{2}|V_G|(|V_G|-1)$ [19, Ch. 14]. A constrained system is said to have finite memory if it can be presented by a deterministic graph G of finite memory.

When the state-splitting algorithm is applied to a graph G of finite memory $\mu(G)$, then the resulting encoder $\mathcal{E}$ is $(\mu(G),a)$-sliding-block decodable, where $a+1$ is bounded from above by the bound on $\mathcal{O}(\mathcal{E})$ of Theorem 2(ii). The improvement of [5] on this bound does not apply to sliding-block decodable encoders since, as was mentioned in Section 1, the encoders obtained in [5] are typically not sliding-block decodable. On the other hand, when $\log(n+1) \leq c(S)$, the stethering construction (or, rather, a punctured version thereof) does provide $(S(G),n)$-encoders which are $(\mu(G), 2|V_G|+1)$-sliding-block decodable.

3 Stethering encoders

Let $G = (V, E, L)$ be a deterministic labeled graph and let n be an integer not greater than $\lambda(A_G)$. We now describe a construction of an (S, n)-encoder $\mathcal{E}$ for $S = S(G)$, based on a construction used by Adler, Goodwyn, and Weiss in [3, Lemma 1]. We call $\mathcal{E}$ a *stethering encoder* for reasons to be apparent in the sequel.

3.1 Definition of stethering encoders

Let $\mathbf{x} = [x_u]_{u \in V_G}$ be an (A_G, n)-approximate eigenvector. Without loss of generality we can assume that the entries of $\mathbf{x}$ are all strictly positive. Otherwise, delete from G the states u for which $x_u = 0$, with their incoming and outgoing edges. The set of states of $\mathcal{E}$ is given by

$$V_{\mathcal{E}} \;=\; \Big\{(u,i) \;\Big|\; u \in V_G \quad \text{and} \quad i = 0, 1, \ldots, x_u - 1\Big\} \,. \tag{1}$$

Clearly, $|V_{\mathcal{E}}| = \|\mathbf{x}\|_1$.

For a state $u \in V_G$, let $E_G(u) = E(u)$ denote the set of outgoing edges from u in G and let $L(E(u))$ denote the set of labels of the edges of $E(u)$. Since G is deterministic, the mapping $E(u) \to L(E(u))$ defined by $e \mapsto L(e)$ is one-to-one and onto. For each $e \in E(u)$ with $a = L(e)$, let $\tau(u; a)$ be the terminal state of e in G. The definition of the edges in $\mathcal{E}$ assumes some ordering on the edges in $E(u)$ for every $u \in V_G$ — or, equivalently, an ordering on the symbols in $L(E(u))$. Note that symbols of Σ may have different ordering relations in $L(E(u))$ for distinct states u.

For $u \in V_G$ and $a \in L(E(u))$, define $\phi(u; a)$ by

$$\phi(u;a) \;=\; \sum_{b \in L(E(u)) \,:\, b < a} x_{\tau(u;b)} \;,$$

where the sum is zero on an empty set.

For each $u, v \in V_G$, $0 \le i < x_u$, $0 \le j < x_v$, and $a \in \Sigma$, we set one edge labeled a from (u, i) to (v, j) in $\mathcal{E}$ if and only if

$$a \in L(E(u)) \,, \quad v = \tau(u;a) \,, \quad \text{and} \quad i\,n \;\le\; \phi(u;a) + j \;<\; (i+1)\,n \,. \tag{2}$$

The following proposition is essentially contained in [3] (although the proof therein was given only for irreducible deterministic labeled graphs G with no parallel edges and $c(S) = \log n$). We repeat the proof here for the sake of completeness and for future reference in this work.

Proposition 1. *Let S be a constrained system presented by a deterministic graph G and let n be a positive integer $\leq \lambda(A_G)$. Then, the labeled graph $\mathcal{E}$ defined by (1)–(2) is an (S, n)-encoder for $S = S(G)$.*

Proof. For $u \in V_G$ and $a \in L(E(u))$, define the set $Z(u; a)$ of integers

$$Z(u; a) \;=\; \Big\{\phi(u; a) + j \mid 0 \leq j < x_{\tau(u;a)}\Big\} \;.$$

Let $a < a'$ be two successive symbols in $L(E(u))$ i.e., there is no symbol $b \in L(E(u))$ such that $a < b < a'$. By the definition of $\phi(u; \cdot)$,

$$\phi(u; a') - \phi(u; a) = x_{\tau(u;a)} \;.$$

It follows that every element of $Z(u; a')$ is greater than every element of $Z(u; a)$ and, therefore,

$$\begin{aligned} \Big|\Big\{\phi(u; a) + j \mid a \in L(E(u)),\; 0 \leq j < x_{\tau(u;a)}\Big\}\Big| \;&=\; \sum_{a \in L(E(u))} |Z(u; a)| \\ &=\; \sum_{a \in L(E(u))} x_{\tau(u;a)} \;=\; (A_G \mathbf{x})_u \;, \end{aligned}$$

where the last term stands for the uth entry in $A_G\mathbf{x}$. Hence, for each $u \in V_G$, the $(A_G\mathbf{x})_u$ terms $\phi(u; a) + j$, $a \in L(E(u))$, $0 \leq j < x_{\tau(u;a)}$, range over all integers between 0 and $\max_{a \in L(E(u))} \Big\{\phi(u; a) + x_{\tau(u;a)} - 1\Big\} = (A_G\mathbf{x})_u - 1$, with each such integer obtained exactly once. Therefore, by (2), the out-degree of each state in $\mathcal{E}$ does not exceed n. Furthermore, since $\mathbf{x}$ is an (A_G, n)-approximate eigenvector, we have $(A_G\mathbf{x})_u \geq n x_u$, which thus implies that the out-degree of each state in $\mathcal{E}$ is exactly n.

We show next that $S(\mathcal{E}) \subseteq S$. Let $\mathbf{w} = w_1 w_2 \ldots w_\ell$ be a word in $S(\mathcal{E})$ i.e., $\mathbf{w}$ is generated by a path

$$\pi_{\mathcal{E}} \;=\; (u_0, i_0) \xrightarrow{w_1} (u_1, i_1) \xrightarrow{w_2} \cdots \xrightarrow{w_\ell} (u_\ell, i_\ell) \tag{3}$$

in $\mathcal{E}$. By construction, the word $\mathbf{w}$ is generated by a path

$$\pi_G \;=\; u_0 \xrightarrow{w_1} u_1 \xrightarrow{w_2} \cdots \xrightarrow{w_\ell} u_\ell \tag{4}$$

in G and, therefore, $\mathbf{w} \in S$. Hence, $S(\mathcal{E}) \subseteq S$.

It remains to show that $\mathcal{E}$ is lossless. Let $\mathbf{w} = w_1 w_2 \ldots w_\ell$ be a word in $S(\mathcal{E})$ generated by a path $\pi_\mathcal{E}$ in $\mathcal{E}$, starting at state (u_0, i_0) and terminating at state (u_ℓ, i_ℓ). Let (3) denote such a path, with (u_0, i_0) and (u_ℓ, i_ℓ) being pre-determined. As stated above, each path $\pi_\mathcal{E}$ that generates $\mathbf{w}$ defines a path π_G, as in (4), that generates the word $\mathbf{w}$ in G. However, since G is deterministic, there is a unique path π_G in G that starts at the pre-determined state u_0 and which generates $\mathbf{w}$. Hence, the components u_r in the $\mathcal{E}$-state names (u_r, i_r) along the path $\pi_\mathcal{E}$ are uniquely determined. Furthermore, by (2) we have

$$i_{r-1} \;=\; \left\lfloor \frac{\phi(u_{r-1}; w_r) + i_r}{n} \right\rfloor \;, \quad r = \ell, \ell-1, \ldots, 1 \;,$$

which thus specifies the path $\pi_\mathcal{E}$ completely. Hence, $\mathcal{E}$ is lossless. Note that in order to reconstruct the path $\pi_\mathcal{E}$, there is no need to pre-determine the component i_0 of the first state (u_0, i_0) in the generating path. □

3.2 Array presentation of stethering encoders

The stethering construction given by (1)–(2) can be illustrated through the array presentation shown in Table 1. Given a deterministic labeled graph G, a positive integer $n \leq \lambda(A_G)$, and an (A_G, n)-approximate eigenvector $\mathbf{x} = [x_u]_u$, we construct for every $u \in V_G$ an array $\mathcal{C}(u) = \mathcal{C}(u; G, \mathbf{x}, n)$ according to some ordering on the set of edges $E(u)$ (which is an ordering also on $L(E(u))$). The array $\mathcal{C}(u)$ consists of two rows referred to as row 0 and row 1. The length of row 0 is x_u units whereas row 1 is of length $(1/n)(A_G\mathbf{x})_u$ units and therefore row 1 is never shorter than row 0.

Row 0 is sub-divided into x_u *unit-intervals* $\mathcal{C}_0(u, i)$, $i = 0, 1, \ldots, x_u - 1$, each of length 1, and with the ith unit-interval corresponding to state (u, i) as in (1). Each unit-interval $\mathcal{C}_0(u, i)$ appears in row 0 of Table 1 as a box of length 1 labeled by the name of its respective state (u, i).

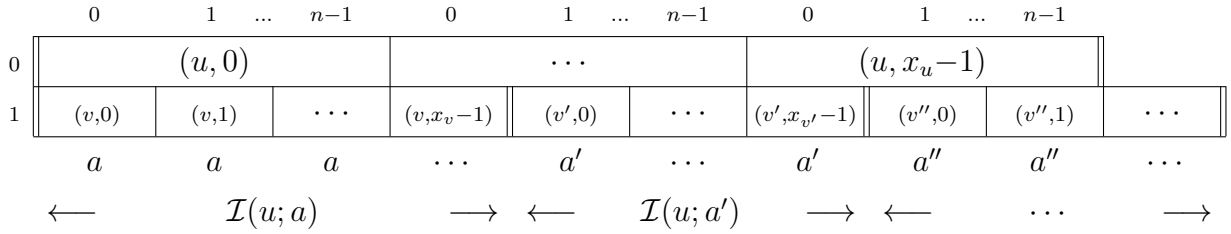

Table 1: Array $\mathcal{C}(u) = \mathcal{C}(u; G, \mathbf{x}, n)$.

Row 1 is divided into *sub-intervals* $\mathcal{C}_1(u; a; j)$, $0 \le j \le x_{\tau(u;a)}$, each of length $1/n$. These sub-intervals are laid in row 1 according to the ordering on $a \in L(E(u))$ and according to increasing values of j. Each sub-interval $\mathcal{C}_1(u; a; j)$ appears in row 1 of Table 1 as a box of length $1/n$. To avoid complex notations in the table, we chose to label the box by (v, j), where $v = \tau(u; a)$, with the symbol a written right underneath the box.

Each one of the first nx_u sub-intervals $\mathcal{C}_1(u; a; j)$ corresponds to an edge $(u, i) \xrightarrow{a} (v, j)$ in $\mathcal{E}$, where, by (2), v and i are uniquely defined by $v = \tau(u; a)$ and $i = \lfloor (\phi(u; a) + j)/n \rfloor$. So, the labels in the boxes in Table 1 that correspond to such sub-intervals $\mathcal{C}_1(u; a; j)$ are the terminal states of the respective edges in $\mathcal{E}$.

The outgoing edges from (u, i) are tagged by $\{0, 1, \ldots, n-1\}$ according to the order of the respective sub-intervals $\mathcal{C}_1(u; a; j)$. These tags are marked in Table 1 on top of row 0.

Hence, the encoding process using a stethering encoder $\mathcal{E}$ can be viewed using Table 1 as follows: given an initial state (u, i) and source tag $s \in \{0, 1, \ldots, n-1\}$, the next state is the label (v, j) in the $(i\,n + s)$th box in row 1 of $\mathcal{C}(u)$ (the leftmost box being the zeroth box) and the output symbol is the one written underneath that box.

For each state $u \in V_G$ and label $a \in L(E(u))$, we group all the sub-intervals $\mathcal{C}_1(u; a; j)$ together into a *super-interval* $\mathcal{I}(u; a)$. The boundaries of the super-intervals $\mathcal{I}(u; a)$ are delineated in Table 1 by *vertical double-lines*, whereas any other sub-interval boundaries are marked by *vertical single-lines* (so is the marking of the boundaries between unit-intervals in row 0). Note that, indeed, by construction, the sub-intervals $\mathcal{C}_1(u; a; j)$ for the same state u and label a form a contiguous interval in row 1. In terms of edges of $\mathcal{E}$, the super-interval $\mathcal{I}(u; a)$ marks the set of all edges $(u, i) \xrightarrow{a} (\tau(u; a), j)$ (if any) in $\mathcal{E}$ that originate from the edge

$u \xrightarrow{a} \tau(u;a)$ in G. The term 'stether', which is a short form for 'stick-together', comes from the fact that all edges $(u,i) \xrightarrow{a} (\tau(u;a),j)$, through their associated sub-intervals $\mathcal{C}_1(u;a;j)$, stick together next to each other in $\mathcal{C}(u)$ for any given u and a. By definition, the length $|\,\mathcal{I}(u;a)\,|$ of $\mathcal{I}(u;a)$ equals $x_{\tau(u;a)}/n$ and

$$\phi(u;a) \;=\; n \;\cdot \sum_{b \in L(E(u))\,:\,b<a} |\,\mathcal{I}(u;b)\,|\;.$$

Using the arrays $\mathcal{C}(v)$, $v \in V_G$, we now define an infinite array $\mathcal{B}(u)$ as shown in Table 2. We number the rows of $\mathcal{B}(u)$ starting with row 0. Row 0 is identical to its counterpart in $\mathcal{C}(u)$. Row 1 in $\mathcal{B}(u)$ is obtained by truncating row 1 of $\mathcal{C}(u)$ to equal in length to row 0 and putting a vertical double-line at the right-hand end point of row 1. The two rows of $\mathcal{B}(u)$ have length x_u.

Row 2 in $\mathcal{B}(u)$ is obtained from row 1 in the following manner. Let $\mathcal{C}_1(u;a;j)$ be a sub-interval in row 1 of $\mathcal{B}(u)$; this sub-interval is labeled (v,j) in Table 2, where $v = \tau(u;a)$. We now lay down n *sub-sub-intervals* in row 2 right underneath the sub-interval $\mathcal{C}_1(u;a;j)$. These sub-sub-intervals are down-scaled copies of the n sub-intervals in row 1 of $\mathcal{B}(v)$ that appear right underneath the unit-interval associated with (v,j) in row 0 of $\mathcal{B}(v)$ (namely, the unit-interval $\mathcal{C}_0(v,j)$). The down-scaling is by a factor of n and therefore each sub-sub-interval is of length $1/n^2$. The sub-sub-intervals maintain the order of the sub-intervals from which they originate. We also maintain the boundaries of the super-intervals in the down-scaling process. Hence, the length of a down-scaled super-interval that originates from a super-interval $\mathcal{I}(v;b)$ is $x_{\tau(v;a)}/n^2$ (unless $\mathcal{I}(v;b)$ is a super-interval that was shortened while forming row 1 of $\mathcal{B}(v)$ out of row 1 of $\mathcal{C}(v)$). The (down-scaled) super-interval boundaries are marked in row 2 of Table 2 by vertical double-lines.

Row 3 in $\mathcal{B}(u)$ is now obtained in a similar way by identifying sub-sub-intervals of row 2 as unit-intervals in row 0 and down-scaling the n respective sub-intervals of row 1 by a factor of n^2. The down-scaling process now continues to form subsequent rows in $\mathcal{B}(u)$. The length of each row equals x_u and row r is divided into $n^r x_u$ (down-scaled) unit-intervals, each of length $1/n^r$. The vertical double-lines separate edges in $\mathcal{E}$ that correspond to different edges in G. Note that, in the arrays, beneath any vertical double-lines, we see only vertical double-lines.

Having constructed arrays $\mathcal{B}(u)$ for an (S,n)-encoder $\mathcal{E}$, let $\mathbf{w} = w_1 w_2 \ldots w_\ell$ be a word

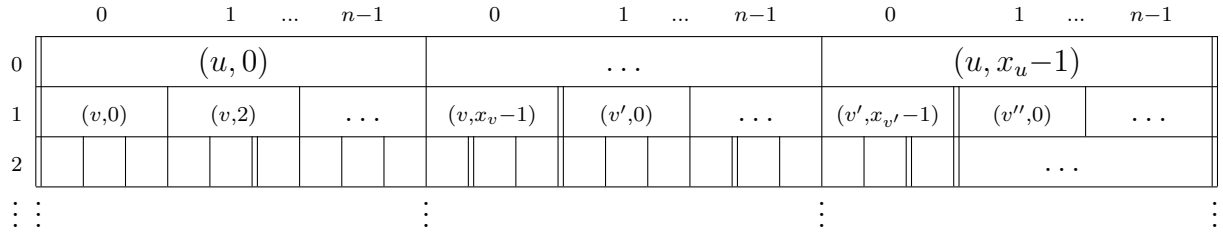

Table 2: Array $\mathcal{B}(u)$.

of length ℓ generated by a path

$$\pi \;=\; (u_0, i_0) \xrightarrow{w_1} (u_1, i_1) \xrightarrow{w_2} \cdots \xrightarrow{w_\ell} (u_\ell, i_\ell)$$

in $\mathcal{E}$ and let $\mathbf{s} = s_1 s_2 \ldots s_\ell$ be the source tag sequence associated with the edges of π. Regarding each row of $\mathcal{B}(u_0)$ as the real interval $[0, x_{u_0})$, we associate with π a point Q_π in that interval whose abscissa is given by $i_0 + q_\pi$, where q_π is the rational whose representation to base n equals $0\,.\,s_1 s_2 \ldots s_\ell$. Let $\mathcal{L}_\pi$ be a vertical line that extends down from Q_π in $\mathcal{B}(u_0)$. For $r = 1, 2, \ldots, \ell$, the line $\mathcal{L}_\pi$ intersects row r of $\mathcal{B}(u_0)$ within a copy of the super-interval $\mathcal{I}(u_{r-1}; w_r)$, down-scaled by a factor of n^{r-1}. Since the original presentation G is deterministic, the sequence $u_1 u_2 \ldots u_\ell$ is determined uniquely by u_0 and $\mathbf{w}$. In particular, the super-intervals intersected by $\mathcal{L}_\pi$ in rows 0 through ℓ of $\mathcal{B}(u_0)$ are completely determined by u_0 and $\mathbf{w}$.

3.3 Examples

Example 1. Let S be the constrained system over $\Sigma = \{a, b, c\}$ presented by the deterministic labeled graph G shown in Figure 1, with $V_G = \{\alpha, \beta, \gamma\}$. It is easy to verify that $[\,3\;2\;1\,]^\top$ is an $(A_G, 2)$-approximate (in fact exact) eigenvector.

Assume first the ordering $a < b < c$ on $L(E(u))$ for all u. The stethering encoder $\mathcal{E}_1$ obtained by this ordering has six states, namely $(\alpha, 0)$, $(\alpha, 1)$, $(\alpha, 2)$, $(\beta, 0)$, $(\beta, 1)$, and $(\gamma, 0)$. The outgoing edges from the first three states can be figured out from the array $\mathcal{C}(\alpha)$ shown in Table 3. Row 0 in $\mathcal{C}(\alpha)$ consists of $x_\alpha = 3$ unit-intervals $\mathcal{C}_0(\alpha, i)$, $i = 0, 1, 2$, one unit-interval for each state (α, i). Below each unit-interval $\mathcal{C}_0(\alpha, i)$ there are $n = 2$ sub-intervals $\mathcal{C}_1(\alpha; w; 0)$

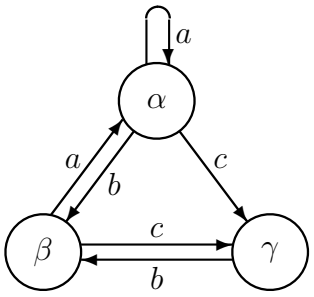

Figure 1: Graph G of Example 1.

	0		1		0		1		0		1
0	$(\alpha,0)$			$(\alpha,1)$				$(\alpha,2)$			
1	$(\alpha,0)$		$(\alpha,1)$		$(\alpha,2)$		$(\beta,0)$		$(\beta,1)$		$(\gamma,0)$
	a		a		a		b		b		c
	$\longleftarrow$		$\mathcal{I}(\alpha;a)$		$\longrightarrow$	$\longleftarrow$		$\mathcal{I}(\alpha;b)$		$\longrightarrow$	$\leftarrow \mathcal{I}(\alpha;c) \rightarrow$

Table 3: Array $\mathcal{C}(\alpha)$ for Example 1.

and $\mathcal{C}_1(\alpha;w;1)$ in row 1, each of length $1/n = 1/2$, where w ranges over $L(E(\alpha)) = \{a,b,c\}$. Each sub-interval $\mathcal{C}_1(\alpha;w;j)$ corresponds to an edge $(u,i) \stackrel{w}{\rightarrow} (\tau(\alpha;w),j)$ in $\mathcal{E}_1$, where i is uniquely defined by (2). The lengths of the resulting super-intervals $\mathcal{I}(\alpha;w)$ are given by $|\mathcal{I}(\alpha;a)| = x_\alpha/2 = 3/2$, $|\mathcal{I}(\alpha;b)| = x_\beta/2 = 1$, and $|\mathcal{I}(\alpha;c)| = x_\gamma/2 = 1/2$. The super-interval boundaries are marked in row 1 by vertical double-lines.

The outgoing picture from states $(\beta,0)$ and $(\beta,1)$, with $L(E(\beta)) = \{a,c\}$, is shown in the array $\mathcal{C}(\beta)$ of Table 4. Similarly, the array $\mathcal{C}(\gamma)$ that corresponds to state $(\gamma,0)$, with $L(E(\gamma)) = \{b\}$, is shown in Table 5.

The resulting fixed-rate $1:1$ encoder $\mathcal{E}_1$ over $\{0,1\}$ into S is presented in Table 6 which specifies the values of the transfer (next-state) function $T : V_{\mathcal{E}_1} \times \{0,1\} \rightarrow V_{\mathcal{E}_1}$ and of the output function $\Gamma : V_{\mathcal{E}_1} \times \{0,1\} \rightarrow \{a,b,c\}$. Two states in $\mathcal{E}_1$, namely $(\alpha,0)$ and $(\beta,0)$, are equivalent and thus can be merged into one state. The encoder thus obtained is lossless, but not lossless of finite order: we can generate an arbitrarily long sequence $bababab\cdots$ from state $(\gamma,0)$ without being able to identify the first edge in the path. An equivalent encoder

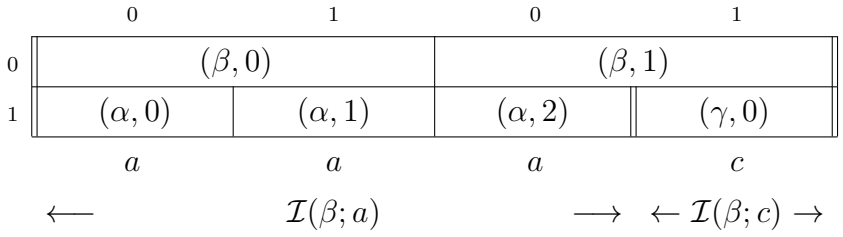

Table 4: Array $\mathcal{C}(\beta)$ for Example 1.

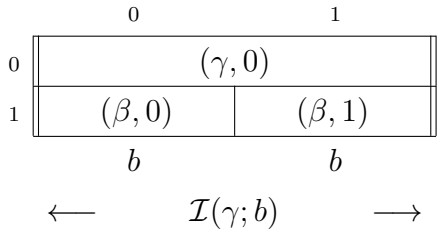

Table 5: Array $\mathcal{C}(\gamma)$ for Example 1.

is obtained if we choose the ordering $c < b < a$ for all $L(E(u))$. •

Example 2. Consider the same constrained system as in Example 1, except that now we assume the ordering $b < a < c$ in $L(E(u))$ for every $u \in V_G$. The six states of the resulting stethering encoder can be merged into three states, $A = \{(\alpha,0),(\gamma,0)\}$, $B = \{(\alpha,1),(\beta,0)\}$, and $C = \{(\alpha,2),(\beta,1)\}$, to obtain the encoder $\mathcal{E}_2$ shown in Table 7. The encoder $\mathcal{E}_2$ is, again, not lossless of finite order. The same encoder is obtained if we choose the ordering

	$T(s,v),\ \Gamma(s,v)$	
v	$s=0$	$s=1$
$(\alpha,0)$	$(\alpha,0),a$	$(\alpha,1),a$
$(\alpha,1)$	$(\alpha,2),a$	$(\beta,0),b$
$(\alpha,2)$	$(\beta,1),b$	$(\gamma,0),c$
$(\beta,0)$	$(\alpha,0),a$	$(\alpha,1),a$
$(\beta,1)$	$(\alpha,2),a$	$(\gamma,0),c$
$(\gamma,0)$	$(\beta,0),b$	$(\beta,1),b$

Table 6: Stethering encoder $\mathcal{E}_1$ for Example 1.

	$T(s,v),\ \Gamma(s,v)$	
v	$s=0$	$s=1$
A	B,b	C,b
B	A,a	B,a
C	A,c	C,a

Table 7: Stethering encoder $\mathcal{E}_2$ for Example 2.

	0	1	0	1	0	1
0	$(\alpha,0)$		$(\alpha,1)$		$(\alpha,2)$	
1	$(\beta,0)$	$(\beta,1)$	$(\gamma,0)$	$(\alpha,0)$	$(\alpha,1)$	$(\alpha,2)$
	b	b	c	a	a	a

Table 8: Array $\mathcal{C}(\alpha)$ for Example 3.

$c < a < b$ for all $L(E(u))$. •

Example 3. Referring again to the constrained system of Example 1, but with the ordering $b < c < a$, we obtain the arrays $\mathcal{C}(\alpha)$, $\mathcal{C}(\beta)$, and $\mathcal{C}(\gamma)$ which are shown in Tables 8–10. The six states of the resulting stethering encoder can be merged into three states to yield the encoder $\mathcal{E}_3$ of Table 11. This encoder has order 3. The graph representation of $\mathcal{E}_3$ is shown in Figure 2, where each edge is labled by the source tag and the output symbol (the double notation on the outgoing edge from state B stands for two parallel edges to state A with the indicated source tags and output symbols). The same encoder is obtained also when we assume the ordering $a < c < b$. In fact, we will get the very same encoder also by the state-splitting algorithm if we start with the vector $\mathbf{x} = [\ 3\ 2\ 1\ ]^\top$ (see also Section 9). By Theorem 1, the number of states in $\mathcal{E}_2$ and $\mathcal{E}_3$ and the order of $\mathcal{E}_3$ are the smallest possible.•

3.4 Losslessness of finite order of stethering encoders

As was exhibited in the previous examples, the stethering construction does not necessarily yield encoders that are lossless of finite order. The following result provides a criterion for a

	0	1	0	1
0	$(\beta, 0)$		$(\beta, 1)$	
1	$(\gamma, 0)$	$(\alpha, 0)$	$(\alpha, 1)$	$(\alpha, 2)$
	c	a	a	a

Table 9: Array $\mathcal{C}(\beta)$ for Example 3.

	0	1
0	$(\gamma, 0)$	
1	$(\beta, 0)$	$(\beta, 1)$
	b	b

Table 10: Array $\mathcal{C}(\gamma)$ for Example 3.

	$T(s,v),\ \Gamma(s,v)$	
v	$s=0$	$s=1$
A	B, b	C, b
B	A, a	A, c
C	B, a	C, a

Table 11: Stethering encoder $\mathcal{E}_3$ for Example 3.

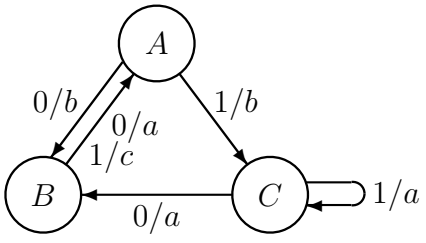

Figure 2: Graph representation of $\mathcal{E}_3$.

stethering encoder to be lossless of finite order in terms of the properties of the arrays $\mathcal{B}(u)$ of Table 2.

Proposition 2. *let S be a constrained system presented by a deterministic labeled graph G and let $\mathcal{E}$ be a stethering (S,n)-encoder. Also, let $\{\mathcal{B}(u)\}_u$ be the arrays associated with $\mathcal{E}$ as in Table 2. Then $\mathcal{E}$ is lossless of finite order if and only if there is a positive integer N such that: for each array $\mathcal{B}(u)$ and for each vertical single-line $\mathcal{L}$ that appears in row 1 of $\mathcal{B}(u)$, but not immediately below a vertical line in row 0, if we extend $\mathcal{L}$ downward through the array, then we must hit a vertical double-line in row N.*

The order of $\mathcal{E}$ is the smallest among all such integers N.

Proof. Assume that such a integer N exists and let

$$\pi \;=\; (u_0,i_0) \xrightarrow{w_1} (u_1,i_1) \xrightarrow{w_2} \cdots \xrightarrow{w_N} (u_N,i_N)$$

and

$$\pi' \;=\; (u'_0,i'_0) \xrightarrow{w_1} (u'_1,i'_1) \xrightarrow{w_2} \cdots \xrightarrow{w_N} (u'_N,i'_N)$$

be two paths of length N in $\mathcal{E}$ with the same initial state $(u_0,i_0)=(u'_0,i'_0)$ that generate the same word $\mathbf{w}=w_1w_2\ldots w_N$. Since G is deterministic, we have $u_0u_1\ldots u_N=u'_0u'_1\ldots u'_N$.

We show that $(u_1,i_1)=(u'_1,i'_1)$, thus proving that $\mathcal{O}(\mathcal{E})\le N$. Suppose, to the contrary, that $i_1<i'_1$. By construction, the (down-scaled) sub-intervals $\mathcal{C}_1(u_{r-1};w_r;i_r)$ and $\mathcal{C}_1(u_{r-1};w_r;i'_r)$ are contained in the same (down-scaled) super-interval $\mathcal{I}(u_{r-1};w_r)$ in row r of $\mathcal{B}(u_0)$ for every $r=1,2,\ldots,N$. On the other hand, by losslessness we must have $i_r\neq i'_r$. Therefore, $i_r<i'_r$ for every $r=1,2,\ldots,N$.

Let $\mathcal{L}$ denote the vertical single-line in row 1 of $\mathcal{B}(u_0)$ that separates the sub-interval $\mathcal{C}_1(u_0;w_1;i_1)$ from $\mathcal{C}_1(u_0;w_1;i_1{+}1)$. The upward extension of $\mathcal{L}$ meets the interior of the unit-interval $\mathcal{C}_0(u_0,i_0)$ in row 0 of $\mathcal{B}(u_0)$. It then follows by the definition of N that the (down-scaled) sub-intervals $\mathcal{C}_1(u_{N-1};w_N;i_N)$ and $\mathcal{C}_1(u_{N-1};w_N;i'_N)$ lie in row N at opposite sides (left and right, respectively) of a vertical double-line that coincides with the downward extension of $\mathcal{L}$. But this contradicts the fact that these two sub-intervals belong to the same (down-scaled) super-interval $\mathcal{I}(u_{N-1};w_N)$. Thus, $i_1=i'_1$ as desired.

Now, suppose there is a vertical single-line $\mathcal{L}$ in row 1 of some array $\mathcal{B}(u_0)$ whose upward extension hits row 0 of $\mathcal{B}(u_0)$ in the interior of some unit-interval $\mathcal{C}_0(u_0, i_0)$ and whose downward extension hits each row $r = 1, 2, \ldots, \ell$ of $\mathcal{B}(u_0)$ in the interior of a (down-scaled) super-interval $\mathcal{I}(u_{r-1}; w_r)$. We show that the order of $\mathcal{E}$ must be greater than ℓ. For every $r = 1, 2, \ldots, \ell$, denote by $\mathcal{C}(u_{r-1}; w_r; i_r)$ and $\mathcal{C}(u_{r-1}; w_r; i_r{+}1)$ the two (down-scaled) subintervals in $\mathcal{I}(u_{r-1}; w_r)$ in row r of $\mathcal{B}(u_0)$ which are separated by the downward extension of $\mathcal{L}$. By construction of stethering encoders it follows that there exist two paths

$$\pi \;=\; (u_0, i_0) \xrightarrow{w_1} (u_1, i_1) \xrightarrow{w_2} (u_2, i_2) \xrightarrow{w_3} \cdots \xrightarrow{w_\ell} (u_\ell, i_\ell)$$

and

$$\pi' \;=\; (u_0, i_0) \xrightarrow{w_1} (u_1, i_1{+}1) \xrightarrow{w_2} (u_2, i_2{+}1) \xrightarrow{w_3} \cdots \xrightarrow{w_\ell} (u_\ell, i_\ell{+}1)$$

in $\mathcal{E}$ with the same initial state but with distinct initial edges and both paths generate the same word $w_1 w_2 \ldots w_\ell$. Hence, the order of $\mathcal{E}$ is greater than ℓ. □

Returning to Example 1, the downward extension of the vertical single-line in row 1 of $\mathcal{B}(\gamma)$ never hits a vertical double-line. Therefore, the encoder $\mathcal{E}_1$ is not lossless of finite order. On the other hand, in Example 3, every vertical single-line in row 1 of any array $\mathcal{B}(u)$ hits a vertical double-line in row 3, but the downward extension of the vertical single-line in row 1 of $\mathcal{B}(\gamma)$ does not hit any vertical double-lines in row 2. Therefore, $\mathcal{O}(\mathcal{E}_3) = 3$.

Although stethering encoders might not be lossless of finite order, there are variations of the stethering construction that do lead to encoders which are lossless of finite order. Such variations will be presented in Sections 5, 6, and 8. In the latter section we also present some extensions of Proposition 2.

4 Encoding complexity

We now turn to determining the complexity of stethering encoders. We assume first that these encoders are implemented by a universal program on a random-access machine (RAM) [4, Ch. 1] and show that encoding can be carried out in polynomial time. Then we use known results from complexity theory to show that finite-state machine realization of stethering encoders requires at most a polynomial number of gates.

We formalize the encoding problem as follows. We are to write a universal encoding program P on a RAM; an input instance to P consists of the following entries:

- a deterministic labeled graph G over an alphabet Σ,
- an integer q,
- an integer n,
- an (A_G^q, n)-approximate eigenvector $\mathbf{x} = [x_u]_{u \in V_G}$,
- a pair (u, i) such that $u \in V_G$ and $0 \leq i < x_u$, and —
- a source tag $s \in \{0, 1, \ldots, n-1\}$.

For any input instance, the program P first verifies that $\mathbf{x}$ is, indeed, an (A_G^q, n)-approximate eigenvector; and, if it is, the program P computes an output q-tuple of symbols over Σ and the next state of the tagged stethering encoder $\mathcal{E}$ defined by (1)–(2), given we are at state (u, i) in $\mathcal{E}$ and the current source tag is s.

The resulting stethering encoder $\mathcal{E}$ is an $(S(G^q), n)$-encoder where, typically, the integer n will be of the form $|\Phi|^p$, in which case $\mathcal{E}$ is a fixed rate $p : q$ encoder over Φ into $S(G)$. Note that we could combine the first two entries in the input instance to P into one entry, namely into a graph G^q. However, by separating G and q we emphasize that the time complexity of P is measured with respect to q and the size of representation of G (whereas G^q might have a number of edges which is exponentially large in q).

In the definition of the encoding problem, we have assumed some fixed ordering on the set $E^q(u)$ of edges outgoing from each state u in G^q, or, equivalently, on the set $L(E^q(u))$ of labels of these edges. To obtain encoders with polynomial complexity, we need the label $\mathbf{w}$ of an edge in $E^q(u)$ and its terminal state to be computable from the index θ of that edge in $E^q(u)$ in time complexity which is $\mathrm{Poly}(\log \theta) = \mathrm{Poly}(|V_G|, q, \log |\Sigma|)$, where $\mathrm{Poly}(\cdot)$ denotes a fixed arbitrary multinomial whose coefficients are absolute constants, independent of its arguments. On the other hand, polynomial decodability requires θ to be efficiently computed from the initial state u and the edge label $\mathbf{w}$.

A straightforward choice of ordering on $E^q(u)$ is the one induced by the lexicographic ordering on Σ^q, according to some fixed ordering on the alphabet Σ. It is easy to verify that

the computation of $\mathbf{w}$ out of θ, and vice versa, can be carried out easily once we know the values of $\sum_{v \in V_G} (A_G^\ell)_{u,v}$ for every $\ell < q$ (see also the following proof of Lemma 1). Clearly, such values can be computed in polynomial time. We adopt here the lexicographic ordering as the ordering on $E^q(u)$, although it is clear that this is not the only ordering that allows for a polynomial-time encoding.

Having defined an ordering on $E^q(u)$, the outgoing edges in $\mathcal{E}$ from state (u, i) to the states $\{\, (\tau(u; \mathbf{w}), j) \,\}_{\mathbf{w}, j}$ are now tagged by $0, 1, \ldots, n-1$, according to the ordering on $\mathbf{w} \in L(E^q(u))$ and the integers j.

Lemma 1. *There exists a universal encoding program P on a RAM that solves the encoding problem in time complexity which is at most* $\mathrm{Poly}\big(|V_G|, \log \|\mathbf{x}\|_1, q, \log |\Sigma|\big)$.

Proof. By (2), all the program P needs to do is to find the largest word $\mathbf{w} \in L(E^q(u))$ such that $\phi(u; \mathbf{w}) \leq i\,n + s$. Then $\mathbf{w}$ is the output symbol on the sth edge outgoing from (u, i) in $\mathcal{E}$ and the next state (v, j) is given by $v = \tau(u; \mathbf{w})$ and $j = (s + i\,n) - \phi(u; \mathbf{w})$.

Hence, it remains to show how to find the word $\mathbf{w}$. For $v \in V_G$, $a \in \Sigma$, and $\ell = 1, 2, \ldots, q$, define the values $\phi_\ell(v; a)$ by

$$\phi_\ell(v; a) \;=\; \sum_{b<a} \sum_{z \in V_G} (A_G^{q-\ell})_{\tau(v;b),z}\, x_z \;.$$

Clearly, the values $\phi_\ell(v; a)$ can be computed in polynomial time. Now, it can be readily verified that for each path

$$u = u_0 \xrightarrow{a_1} u_1 \xrightarrow{a_2} u_2 \xrightarrow{a_3} \cdots \xrightarrow{a_q} u_q$$

in G we have,

$$\phi(u; a_1 a_2 \ldots a_q) \;=\; \sum_{\ell=1}^{q} \phi_\ell(u_{\ell-1}; a_\ell) \;.$$

This suggests an enumerative method for finding the desired word $\mathbf{w} = w_1 w_2 \ldots w_q$ as follows: Assume we have determined the symbols w_ℓ for $1 \leq \ell < r$ and let u_ℓ, $0 \leq \ell < r$, be defined by $u_0 = u$ and $u_\ell = \tau(u_{\ell-1}; w_\ell)$. Then, the symbol w_r is the largest element in Σ for which

$$\phi_r(u_{r-1}; w_r) \;\leq\; i\,n + s \;-\; \sum_{\ell=1}^{r-1} \phi_\ell(u_{\ell-1}; w_\ell) \;.$$

The word $\mathbf{w}$ is now obtained by applying this procedure iteratively for $r = 1, 2, \ldots, q$. We remark that a similar (in fact simpler) procedure can be used to compute the label $\mathbf{w}$ of an edge $u \xrightarrow{\mathbf{w}} v$ in G^q out of the index θ of that edge in $E^q(u)$. $\square$

For a positive integer ℓ denote by I_ℓ the set of all possible inputs to P of size ℓ, according to some standard coding of the input. Verifying that the running time of P on each element of I_ℓ is polynomial in ℓ, it follows that for any input size ℓ there exists a circuit C_ℓ with $\mathrm{Poly}(\ell) = \mathrm{Poly}\Big(|V_G|, \log \|\mathbf{x}\|_1, q, \log |\Sigma|\Big)$ gates that implements P for inputs in I_ℓ. Furthermore, such circuits C_ℓ are 'uniform' in the sense that there is a program on a RAM that generates the layouts of C_ℓ in time complexity which is $\mathrm{Poly}(\ell)$. This is a consequence of a known result on the equivalence between polynomial circuit complexity and polynomial RAM complexity of decision problems [24, Theorem 2.3].

Having such a polynomial circuit at hand, we can now obtain an implementation of any stethering encoder $\mathcal{E}$ by 'hard-wiring' into the circuit a 'small' (A_G^q, n)-approximate eigenvector $\mathbf{x}$, e.g., a vector $\mathbf{x}$ with $\|\mathbf{x}\|_\infty \leq n^{2|V_G|}$. Such a vector always exists whenever $n \leq \lambda(A_G^q)$; furthermore, if we do not insist on minimizing $\|\mathbf{x}\|_\infty$ (yet still having $\|\mathbf{x}\|_\infty \leq n^{2|V_G|}$), then $\mathbf{x}$ can be computed in time complexity which is polynomial in the representation of A_G^q and n [5, Lemmas 2,7]. The resulting stethering encoders will have $O\Big(|V_G| \log n\Big)$ memory bit-cells to code the states (u, i) of $\mathcal{E}$ and $\mathrm{Poly}\Big(|V_G|, q, \log |\Sigma|\Big)$ gates. We summarize this in the following theorem.

Theorem 3. *For every constrained system S over an laphabet Σ presented by a deterministic labeled graph G and for any positive integers q and $n \leq \lambda(A_G^q)$ there exists a stethering (S^q, n)-encoder defined by (1)–(2) that can be implemented by a circuit consisting of $\mathrm{Poly}\Big(|V_G|, q, \log |\Sigma|\Big)$ gates and $O\Big(|V_G| \log n\Big)$ memory bit-cells. Furthermore, there exists a program on a RAM that generates the layout of such an implementation in polynomial-time.*

5 Encoders at rates smaller than capacity

In this section we use the stethering method as a basis of a construction that yields encoders that are lossless of finite order. We first show how to obtain such (S, n)-encoders for $S = S(G)$

whenever $n \leq \lambda(A_G) - 1$, where G is a deterministic presentation of S. Then, we use these encoders to construct (S, n)-encoders whenever $n < \lambda(A_G)$.

5.1 The case $n \leq \lambda(A_G) - 1$

Given a deterministic labeled graph G and an integer $n \leq \lambda(A_G)-1$, we start by constructing a stethering $(S, n+1)$-encoder $\mathcal{E}$ for $S = S(G)$, using an $(A_G, n+1)$-approximate eigenvector $\mathbf{x}$. For each state (u, i) in $\mathcal{E}$, we tag the outgoing edges from (u, i) by distinct elements of $\{0, 1, \ldots, n\}$ according to the ordering on their terminal states $(\tau(u; a), j)$ (recall that this ordering is induced by the ordering on $a \in L(E(u))$ and the ordering on j). Next, we delete all edges in $\mathcal{E}$ tagged by n, resulting in an (S, n)-encoder $\mathcal{E}^*$ which we refer to as a *punctured* version of $\mathcal{E}$.

Proposition 3. *Let S be a constrained system presented by a deterministic graph G and let n be a positive integer $\leq \lambda(A_G)-1$. Let $\mathcal{E}$ be a stethering $(S, n+1)$-encoder obtained by (1)–(2) for some $(A_G, n+1)$-approximate eigenvector $\mathbf{x}$ and let $\mathcal{E}^*$ be a punctured version of $\mathcal{E}$. Then, $\mathcal{E}^*$ is an (S, n)-encoder with*

$$\mathcal{O}(\mathcal{E}^*) \;\leq\; 2 \;+\; \left\lceil \frac{\log \|\mathbf{x}\|_\infty}{\log (n+1)} \right\rceil .$$

Proof. Let $\mathcal{E}$ be a stethering $(S, n+1)$-encoder obtained using an $(A_G, n+1)$-approximate eigenvector $\mathbf{x}$. Let $\mathcal{B}(u)$ be the infinite array that corresponds to $\mathcal{E}$ as shown in Table 2 (note that each unit-interval in row 0 is divided here into $n+1$ sub-intervals in row 1 and the down-scaling factor of subsequent rows is $n+1$).

Let $\mathbf{w} = w_1 w_2 \ldots w_\ell$ be a word of length ℓ generated by a path

$$\pi \;=\; (u_0, i_0) \xrightarrow{w_1} (u_1, i_1) \xrightarrow{w_2} \cdots \xrightarrow{w_\ell} (u_\ell, i_\ell)$$

in $\mathcal{E}$ and let $\mathbf{s} = s_1 s_2 \ldots s_\ell$ be the source tag sequence over $\{0, 1, \ldots, n\}$ associated with the edges of π. Following the notations of Section 3.2, we associate with π a point Q_π in $[0, x_{u_0})$ whose abscissa is given by $i_0 + q_\pi$, where q_π is the rational whose representation to base $n+1$ equals $0 \,.\, s_1 s_2 \ldots s_\ell$.

As we have already shown in the proof of Proposition 1, the sequence $u_1 u_2 \dots u_\ell$ is uniquely determined by u_0 and $\mathbf{w}$. In particular, the (down-scaled) super-intervals in rows 0 through ℓ of $\mathcal{B}(u_0)$ which are intersected by the vertical line extending down from Q_π are completely determined by u_0 and $\mathbf{w}$.

We now assume that the path π exists also in $\mathcal{E}^*$ i.e. that $s_r < n$ for $r = 1, 2, \dots, \ell$. In order to complete the proof, it remains to find a condition on ℓ that allows to determine s_1 uniquely, given u_0 and $\mathbf{w}$. Indeed, since $s_1 < n$, we can guarantee the uniqueness of s_1 by determining the value of q_π up to an additive error which is at most $\pm 1/(n+1)^2$ (actually, an error $\pm 1/(n(n+1))$ would suffice). Now, the longest super-interval in row ℓ of $\mathcal{B}(u_0)$ is of length $\leq \|\mathbf{x}\|_\infty/(n+1)^\ell$. Hence, if

$$\frac{\|\mathbf{x}\|_\infty}{(n+1)^\ell} \;\leq\; \frac{1}{(n+1)^2}\,,$$

we know the value of q_π up to an accuracy which allows to reconstruct the value of s_1. Taking logarithms, we obtain

$$\ell \;\geq\; 2 \;+\; \frac{\log \|\mathbf{x}\|_\infty}{\log (n+1)}$$

as a sufficient condition for $\mathcal{E}^*$ to be of order $\leq \ell$. □

It can be readily verified that the proof of Proposition 3 is constructive i.e., it suggests a polynomial-time decoding algorithm for $\mathcal{E}^*$. Hence, there exists a finite-state decoder for $\mathcal{E}^*$ that can be implemented by $\mathrm{Poly}\Big(|V_G|, q, \log|\Sigma|\Big)$ gates and $O\Big(|V_G|\,\log\,n\Big)$ memory bit-cells.

Applying Proposition 3 on G^q, we obtain the following.

Corollary 1. *Let G be a deterministic presentation of S and let q and n be positive integers such that $n \leq \lambda(A_G^q) - 1$. Let $\mathcal{E}$ be a stethering $(S^q, n+1)$-encoder obtained by (1)–(2) for some $(A_G^q, n+1)$-approximate eigenvector $\mathbf{x}$ and let $\mathcal{E}^*$ be a punctured version of $\mathcal{E}$. Then,*

$$\mathcal{O}(\mathcal{E}^*) \;\leq\; 2 \;+\; \left\lceil \frac{\log \|\mathbf{x}\|_\infty}{\log (n+1)} \right\rceil\,.$$

In particular, for $\|\mathbf{x}\|_\infty \leq (n+1)^{2|V_G|}$,

$$\mathcal{O}(\mathcal{E}^*) \;\leq\; 2|V_G| \;+\; 2\,.$$

Furthermore, such a bound can be achieved by an encoder $\mathcal{E}^*$ and a respective finite-state decoder that have circuit implementations of $\mathrm{Poly}\big(|V_G|, q, \log|\Sigma|\big)$ *gates and* $O\big(|V_G|\,\log\,n\big)$ *memory bit-cells.*

In terms of fixed-rate $p:q$ encoders over Φ into S, Corollary 1 becomes

$$\mathcal{C}(\mathcal{E}^*) \;\le\; 2 \,+\, \left\lceil \frac{\log\|\mathbf{x}\|_\infty}{\log\left(|\Phi|^p+1\right)} \right\rceil ,$$

where $n+1 = |\Phi|^p + 1 \le \lambda(A_G^q)$. This condition on p and q is satisfied whenever

$$\frac{p}{q}\log|\Phi| \;\le\; c(S) \;-\; \frac{\log(1+|\Phi|^{-p})}{q} ,$$

which is, in turn, satisfied whenever

$$\frac{p}{q}\log|\Phi| \;\le\; c(S) \;-\; \frac{\log\,e}{|\Phi|^p q} . \tag{5}$$

Corollary 2. *Let G be a deterministic labeled graph with finite memory $\mu(G)$ and let q and n be a positive integers such that $n \le \lambda(A_G^q) - 1$. Let $\mathcal{E}$ be a stethering $(S, n+1)$-encoder obtained by (1)–(2) for some $(A_G^q, n+1)$-approximate eigenvector $\mathbf{x}$ and let $\mathcal{E}^*$ be a punctured version of $\mathcal{E}$. Then, $\mathcal{E}^*$ is an (m,a)-sliding-block decodable (S^q, n)-encoder with $m \le \lceil \mu(G)/q \rceil$ and*

$$a \;\le\; 1 \,+\, \left\lceil \frac{\log\|\mathbf{x}\|_\infty}{\log\left(n+1\right)} \right\rceil .$$

In particular, for $\|\mathbf{x}\|_\infty \le (n+1)^{2|V_G|}$,

$$a \;\le\; 2|V_G| \,+\, 1 ,$$

and such a look-ahead can be achieved by an encoder $\mathcal{E}^*$ and a respective sliding-block decoder that have polynomial-size implementations.

Proof. This is a consequence of the fact that in the proof of Proposition 3 (as was the case in Proposition 1), the entry i_0 in the name (u_0, i_0) of the first state of a path in $\mathcal{E}^*$ that generates a given word $\mathbf{w}$ is not required for reconstructing the first tag. As for u_0, it is uniquely determined by the $\lceil \mu(G)/q \rceil$ symbols, over Σ^q, that precede the time slot for which the source tag is reconstructed. The polynomiality of the sliding-block decoder follows from $\mu(G) \le \frac{1}{2}|V_G|(|V_G|-1)$ (see [19, Ch. 14]). □

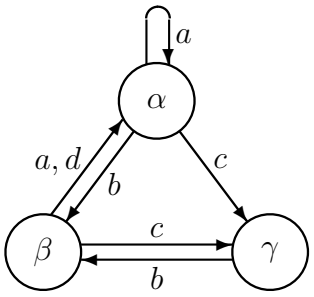

Figure 3: Graph G of Example 4.

The condition (5) on the rate $p : q$ guarantees also sliding-block decodability when G is of finite memory. We remark that Corollary 2 holds regardless of the tagging of the edges of $\mathcal{E}$.

Example 4. Let S be the constrained system over $\Sigma = \{a, b, c, d\}$ presented by the deterministic labeled graph G of Figure 3 (note that there are two parallel edges from state β to state α). Here $c(S) = \log \lambda(A_G) = \log(1 + \sqrt{2}) \approx 1.27$, which readily suggests rate $5 : 4$ for a fixed-rate encoder over $\{0, 1\}$ into S. The matrix A_G^4 is given by

$$A_G^4 = \begin{bmatrix} 17 & 12 & 12 \\ 16 & 13 & 12 \\ 8 & 4 & 5 \end{bmatrix},$$

and $\mathbf{x} = [\, 3\ 3\ 1\,]^\top$ is an $(A_G^4, 2^5)$-approximate eigenvector with a largest component, 3, which is the smallest possible. Since $A_G^4 \mathbf{x} \geq 33\,\mathbf{x}$, the vector $\mathbf{x}$ is also an $(A_G^4, 33)$-approximate eigenvector with a largest component which is the smallest possible. By Corollary 1, it follows that any punctured stethering $(S^4, 32)$-encoder $\mathcal{E}^*$ has order ≤ 3, regardless of the ordering on the sets $E^4(u)$ of edges outgoing from each state u in G^4. Furthermore, since $\mu(G) = 1$, the resulting encoder $\mathcal{E}^*$ is $(1, 2)$-sliding-block decodable. However, for certain orderings of edges we can have smaller order and look-ahead.

Consider for instance the $(S^4, 33)$-encoder $\mathcal{E}$ defined by Tables 12–14. The notation $m\times\|(v, 0)\ (v, 1)\ (v, x_v{-}1)\|$ in an array $\mathcal{C}(u)$ stands for m consecutive super-intervals $\mathcal{I}(u; \mathbf{w})$, each consisting of x_v sub-intervals $\mathcal{C}_1(u; \mathbf{w}; j)$ of length $1/(n + 1) = 1/33$, with $\mathbf{w}$ ranging over all 4-tuples in $L(E^4(u))$ that label an edge $u \to v$ in G^4. The outgoing edge tagged by $n = 32$ from each state of $\mathcal{E}$ is deleted when we puncture $\mathcal{E}$ to obtain $\mathcal{E}^*$.

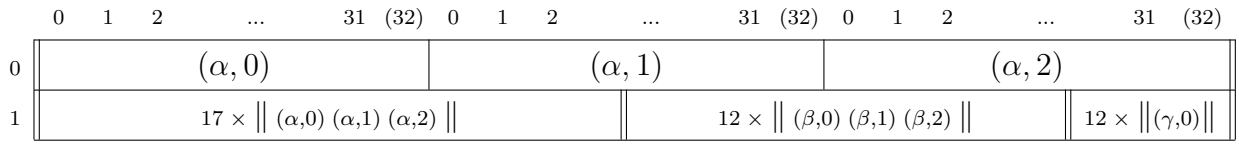

Table 12: Array $\mathcal{C}(\alpha)$ for an $(S^4, 33)$-encoder for Example 4.

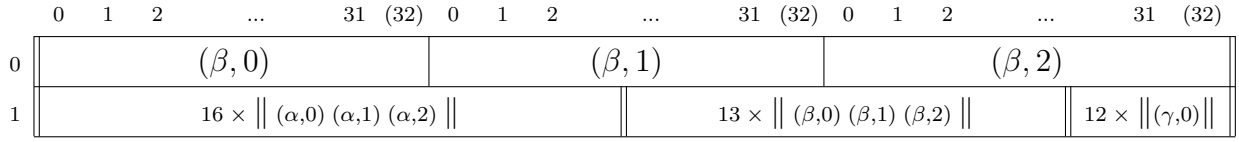

Table 13: Array $\mathcal{C}(\beta)$ for an $(S^4, 33)$-encoder for Example 4.

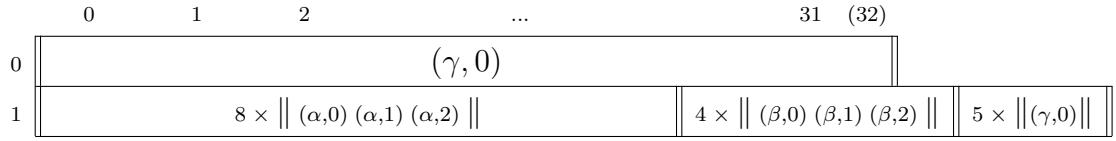

Table 14: Array $\mathcal{C}(\gamma)$ for an $(S^4, 33)$-encoder for Example 4.

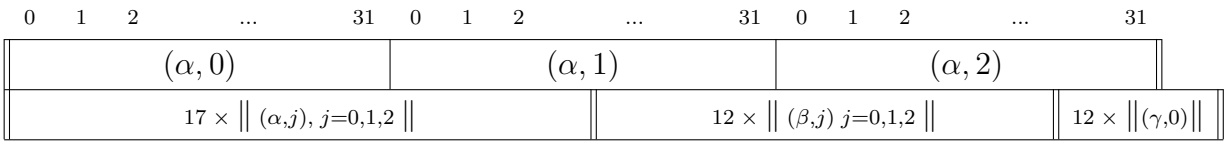

Table 15: Array $\mathcal{C}^*(\alpha)$ for an $(S^4, 32)$-encoder for Example 4.

In all three arrays, there is a vertical double-line in row 1 underneath each vertical single-line in row 0. Hence, row 2 in the arrays $\mathcal{B}(u)$ associated with $\mathcal{E}$ (Table 2) have vertical double-lines at all points with abscissas of the form $m/(n+1) = m/33$, $m = 0, 1, 2, \ldots$, and for all states $u \in V_G$. By Proposition 2 it thus follows that $\mathcal{E}$, and therefore $\mathcal{E}^*$, both have order 2 and are $(1, 1)$-sliding-block decodable. Note that by Theorem 1(ii) this is the smallest order possible for any $(S^4, 32)$-encoder.

In fact, the existence of a vertical double-line underneath each vertical single-line in all three arrays suggests that $\mathcal{E}$ can be obtained by one round of the state-splitting algorithm [1][5]. •

A slightly different description of punctured stethering encoders will turn out to be useful in Section 8, where we define the notion of pseudo-stethering which, in turn, contains both stethering and punctured stethering as special cases. We illustrate the alternative description of punctured stethering through the $(S^4, 32)$-encoder of Example 4. Rather than starting with arrays $\mathcal{C}(u)$ of a stethering $(S^q, n+1)$-encoder, we regard the $(A_G^q, n+1)$-approximate eigenvector $\mathbf{x}$ as an (A_G^q, n)-approximate eigenvector and construct respective arrays $\mathcal{C}^*(u)$ where the length of each sub-interval $\mathcal{C}_1^*(u; a; j)$ in row 1 is $1/n$ (and not $1/(n+1)$). Table 15 depicts the array $\mathcal{C}^*(\alpha)$ for the $(A_G^4, 33)$-approximate eigenvector $\mathbf{x} = [\ 3\ 3\ 1\ ]^\top$ and the edge ordering that were used in Example 4.

We now construct the infinite array $\mathcal{B}^*(u)$ for each $u \in V_G$ as follows. Row 0 of $\mathcal{B}^*(u)$ is equal to row 0 of $\mathcal{C}^*(u)$. Row 1 of $\mathcal{B}^*(u)$ is obtained by first removing every $(n+1)$st sub-interval $\mathcal{C}_1^*(u; a; j)$ from row 1 of $\mathcal{C}^*(u)$ and closing the resulting gaps. The excess part of row 1 beyond length x_u (if any) is then truncated to equal in length to row 0, leaving the vertical double-line at the right-hand end point of row 1. Subsequent rows of $\mathcal{B}^*(u)$ are obtained by the down-scaling process (by a factor of n) as was described in Section 3.2.

0 1 2 ... 31		0 1 2 ... 31			0 1 2 ... 31	
$(\alpha,0)$		$(\alpha,1)$			$(\alpha,2)$	
$10\times\|(\alpha,j),j{=}0,1,2\|$	$(\alpha,0)(\alpha,1)$	$6\times\|(\alpha,j),j{=}0,1,2\|$	$4\times\|(\beta,j),j{=}0,1,2\|$	$(\beta,0)(\beta,1)$	$7\times\|(\beta,j),j{=}0,1,2\|$	$11\times\|(\gamma,0)\|$

Table 16: First two rows of $\mathcal{B}^*(\alpha)$ for the punctured $(S^4, 32)$-encoder for Example 4.

Table 16 depicts the first two rows of $\mathcal{B}^*(\alpha)$ for the $(S^4, 32)$-encoder of Example 4. Iterating the process for all states $u \in V_G$, we can 'read' the punctured (S^q, n)-encoder $\mathcal{E}^*$ out of the first two rows of the arrays $\mathcal{B}^*(u)$ as if it were a (non-punctured) stethering encoder. While punctured stethering means removing every $(n+1)$st sub-interval from row 1 of $\mathcal{C}^*(u)$, pseudo-stethering allows arbitrary sub-intervals to be removed from row 1. This will be discussed in more detail in Section 8.

Example 5. The upper bound on the order of punctured stethering encoders, as stated in Proposition 3, is similar in form to the lower bound stated in Theorem 1(ii). In fact, there are many cases where the difference between these bounds is not greater than 1. However, for every $n \geq 2$ there exists an infinite sequence of constrained systems $\{S_k\}_k$ with capacity $\log(n+1)$ for which the smallest upper bound obtained by Proposition 3 is linearly increasing with the number of states in a deterministic presentation G_k of S_k, whereas the lower bound of Theorem 1(ii) is sharp and equals to 2. Such an infinite sequence can be obtained by modifying the example given in [21, Figure 1]: Given n, the constrained-system alphabet for all S_k is of fixed size n^2+n+1 and is given by $\{a\} \cup \{b_i\}_{i=1}^{n^2+n-1} \cup \{c\}$. The deterministic presentation G_k of each S_k consists of $2k$ states and is given in Figure 4 (note that from each state $u \leq k$ there are n^2+n-1 parallel outgoing edges labeled by the b_i's to state $k+u$). It is easy to verify that $\lambda(A_{G_k}) = \lambda = n+1$ and that every (A_{G_k}, λ)-approximate eigenvector is a multiple of $[\ \lambda\ \lambda^2\ \ldots\ \lambda^k\ 1\ \lambda\ \ldots\ \lambda^{k-1}\ ]^\top$. Hence, the smallest upper bound obtained by Proposition 3 is $2+k = 2+\frac{1}{2}|V_{G_k}|$. On the other hand, the vector $\mathbf{x} = [x_u]_u$ whose nonzero components are $x_k = n$ and $x_{2k} = 1$ is an (A_{G_k}, n)-approximate eigenvector that yields an (S_k, n)-encoder of order 2 by (non-punctured) stethering or by one round of the state-splitting algorithm applied on G_k. This is the smallest order possible, since there is no (A_{G_k}, n)-approximate eigenvector consisting of just 0's and 1's. •

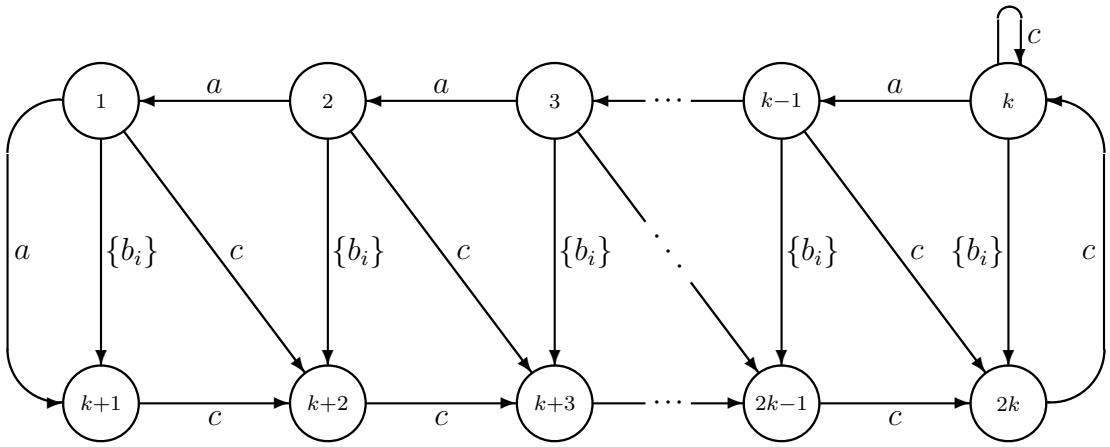

Figure 4: Graph G_k of Example 5.

5.2 The case $n < \lambda(A_G)$

We now consider the case where (5) is not satisfied, but still $(p/q)\log|\Phi| < c(S)$. Our strategy in this case will be first to apply the punctured stethering construction on a power graph G^t for $t = 3|V_G|$, thus obtaining an encoder of rate $p\,t : q\,t$ which is lossless of finite order. Then we translate the resulting encoder into a fixed-rate $p : q$ encoder which has order $\leq 12|V_G|$.

To show the existence of such an integer t we use the following lemma.

Lemma 2. *Let A be a $k \times k$ nonnegative integer matrix and let n be a positive integer smaller than $\lambda(A)$. Then, there exists an $(A^{3k}, n^{3k}+1)$-approximate eigenvector $\mathbf{x}$ with $\|\mathbf{x}\| \leq n^{2k}$ which can be computed in polynomial-time.*

Proof. Let $\mathbf{x}$ be an (A,n)-approximate eigenvector with $\|\mathbf{x}\|_\infty \leq n^{2k}$. Such a vector can be computed in polynomial-time [5, Lemma 7]. By [5, Lemma 8] we have, for every $t \geq k$,

$$A^t\mathbf{x} \;\geq\; n^{t-k}\,\mathbf{1} \;+\; n^t\,\mathbf{x}\,,$$

where $\mathbf{1}$ denotes the all-one vector. Setting $t = 3k$ we thus obtain

$$A^{3k}\mathbf{x} \;\geq\; n^{2k}\,\mathbf{1} \;+\; n^{3k}\,\mathbf{x} \;\geq\; (n^{3k}+1)\,\mathbf{x}\,,$$

as claimed. □

Let G be a deterministic presentation of a constrained system S over an alphabet Σ and let $n < \lambda(A_G)$. An (S, n)-encoder with order $\leq 12|V_G|$ is now constructed as follows. For $t = 3|V_G|$, we first construct a stethering $(S^t, n^t + 1)$-encoder $\mathcal{E}_1$. Puncturing $\mathcal{E}_1$, we obtain an (S^t, n^t)-encoder $\mathcal{E}_2 = \mathcal{E}_1^*$ of order ≤ 3, where the order is measured in symbols of Σ^t. Next, we construct the Moore-type equivalent $\mathcal{E}_3$ of $\mathcal{E}_2$ [19, p. 352]: for every state u in $\mathcal{E}_2$ and symbol $\mathbf{a} \in \Sigma^t$ on an edge incoming to u, we define a state $[u, \mathbf{a}]$ in $\mathcal{E}_3$, and for each edge $u \xrightarrow{\mathbf{b}} v$ in $\mathcal{E}_2$ we set an edge $[u, \mathbf{a}] \xrightarrow{\mathbf{a}} [v, \mathbf{b}]$ in $\mathcal{E}_3$. It is easy to verify that $S(\mathcal{E}_3) = S(\mathcal{E}_2)$ and, therefore, $\mathcal{E}_3$ is an (S^t, n^t)-encoder. By construction, the outgoing edges from each state in $\mathcal{E}_3$ all have the same label and, in addition, $\mathcal{O}(\mathcal{E}_3) \leq 4$. As a final step, we construct an (S, n)-encoder $\mathcal{E}_4$ by replacing the n^t outgoing edges from each state in $\mathcal{E}_3$ by an n-ary tree of depth t. The encoder $\mathcal{E}_4$ has order $\leq 4t = 12|V_G|$ (now measured in symbols of Σ) and can be implemented by a polynomial-size circuit. We remark that a similar strategy was also applied in [5]. For $n < \lambda(A_G)$, the upper bound on the order of the construction in [5] is greater than $15|V_G|$.

The properties of encoders obtained by the constructions of Sections 3 and 5 are summarized in Theorem 5 of Section 7.

6 Encoders at rates equaling capacity

In this section we consider constrained systems S with capacity $\log n$ for some integer n. The technique used in Section 5.2 does not allow for constructing an (S, n)-encoder for this case. Instead, we will first construct a (non-punctured) stethering (S^t, n^t)-encoder and then transform the resulting encoder into an (S, n)-encoder. The value of t and the ordering on the edges of the deterministic presentation of S^t that we use will guarantee losslessness of finite order.

Let G be a deterministic presentation of S with $\lambda(A_G) = n$. In [5], a construction was shown of (S, n)-encoders whose order is at most $9|V_G| + 6\lceil(\log |V_G|)/\log n\rceil$. The construction we show here is a variation of the construction of [5] using the stethering notation. By applying the stethering method on a power of G we obtain an (S, n)-encoder with a

polynomial-size implementation and with an upper bound on its order which is the same as in [5]. This upper bound on the order is based on a bound stated in Theorem 1 of [5].

6.1 Example

We start by illustrating the construction for the constrained system S presented by the deterministic labeled graph $G = (V, E, L)$ over $\Sigma = \{a, b, c\}$ of Figure 1. The adjacency matrix of G is given by

$$A_G \;=\; \begin{bmatrix} 1 & 1 & 1 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} .$$

Let $\mathbf{x}$ be the right eigenvector $[\,3\;2\;1\,]^\top$ associated with the eigenvalue $\lambda(A_G) = n = 2$. The encoder construction is based on finding a small positive integer t and a decomposition of each row in A_G^t as a sum

$$(A_G^t)_u \;=\; \mathbf{z}^{(u,0)} + \mathbf{z}^{(u,1)} + \ldots + \mathbf{z}^{(u,x_u-1)} \quad , \; u \in V \; ,$$

where each $\mathbf{z}^{(u,i)}$ is a nonnegative integer vector satisfying $\mathbf{z}^{(u,i)} \cdot \mathbf{x} = n^t$. Having such a decomposition at hand, we define an ordering on the set $E^2(u)$ of outgoing edges from each state u in G^t. The specific ordering we choose will enable us to obtain a stethering (S^t, n^t)-encoder $\mathcal{E}$ of order 2. Equivalently, the encoder $\mathcal{E}$ can be obtained by one round of the state-splitting algorithm applied to the graph G^t. Then, we transform the encoder $\mathcal{E}$ into an (S, n)-encoder, as was done in Section 5, to obtain an (S, n)-encoder with order $\leq 3t$.

Checking the value $t = 1$, it is easy to verify that only the third row of A_G can be decomposed as described. Consider now the case $t = 2$, where

$$A_G^2 \;=\; \begin{bmatrix} 2 & 2 & 2 \\ 1 & 2 & 1 \\ 1 & 0 & 1 \end{bmatrix} .$$

The first row of A_G^2 can be decomposed as $(A_G^2)_\alpha = \mathbf{z}^{(\alpha,0)} + \mathbf{z}^{(\alpha,1)} + \mathbf{z}^{(\alpha,2)}$, where $\mathbf{z}^{(\alpha,0)} = \mathbf{z}^{(\alpha,1)} = [\,1\;0\;1\,]$ and $\mathbf{z}^{(\alpha,2)} = [\,0\;2\;0\,]$. As for the second row, we have $\mathbf{z}^{(\beta,0)} = [\,1\;0\;1\,]$ and $\mathbf{z}^{(\beta,1)} = [\,0\;2\;0\,]$, and for the third row we have $\mathbf{z}^{(\gamma,0)} = [\,1\;0\;1\,]$. Each of the vectors $\mathbf{z}^{(u,i)}$ dots with $\mathbf{x}$ to yield $\lambda(A_G^2) = 2^2 = 4$.

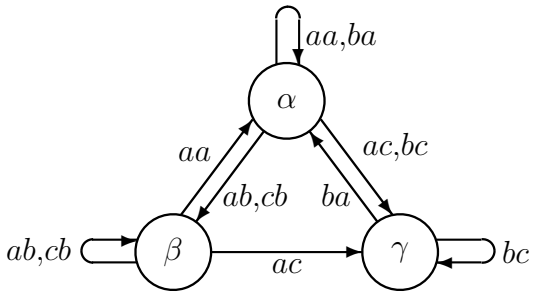

Figure 5: Graph G^2 for the example.

The second power S^2 of the constrained system S is presented by the labeled graph G^2 shown in Figure 5, with adjacency matrix $A_{G^2} = A_G^2$. For each state $u \in V$, we partition its set of outgoing edges $E^2(u)$ in G^2 into x_u subsets $E^2(u,0), E^2(u,1), \ldots, E^2(u, x_u - 1)$ so that the number of edges in $E^2(u,i)$ that terminate at state v is $(\mathbf{z}^{(u,i)})_v$. Now, fix some ordering on the symbols of Σ, say, $a < b < c$, and on the states in V, say $\alpha < \beta < \gamma$. The edges in $E^2(u)$, with their corresponding labels $L(E^2(u))$, are now ordered as follows: The edges in $E^2(u,i)$ precede those in $E^2(u,i')$ whenever $i < i'$; within each subset $E^2(u,i)$, the edges are ordered according to their terminal state v in G^2; and, finally, all edges in $E^2(u,i)$ with the same terminal state v are ordered according to the lexicographic ordering on their labels defined by the ordering on Σ.

The $(S^2, 4)$-encoder graph $\mathcal{E}$ is now defined as a stethering encoder obtained for this edge ordering.

We illustrate a particular choice of edge partition for $E^2(\alpha)$. Each edge $\alpha \xrightarrow{\mathbf{w}} v$ in G^2 is represented in Figure 6 by a rectangle labeled $\mathbf{w}$ whose length is equal to $x_v/n^2 = x_v/4$. The rectangles representing those edges with a common terminal state v are arranged in a single column headed by the name of v. The decomposition of $(A_G^2)_\alpha$ into [1 0 1]+[1 0 1]+[0 2 0] induces a partition of $E^2(\alpha)$ into subsets $E^2(\alpha, i)$ of edges labeled by $L(E^2(\alpha,0)) = \{aa, ac\}$, $L(E^2(\alpha,1)) = \{ba, bc\}$, and $L(E^2(\alpha,2)) = \{ab, cb\}$, as shown in Figure 7 (note that this is not the only partition induced by the above decomposition). Using the partition of Figure 7, we obtain the ordering $aa < ac < ba < bc < ab < cb$ on $L(E^2(\alpha))$.

The corresponding array $\mathcal{C}(\alpha)$ is shown in Table 17, which also determines the source tags on the edges outgoing from states $(\alpha, 0)$, $(\alpha, 1)$, and $(\alpha, 2)$ in $\mathcal{E}$. Each interval $\mathcal{I}(\alpha; \mathbf{w})$

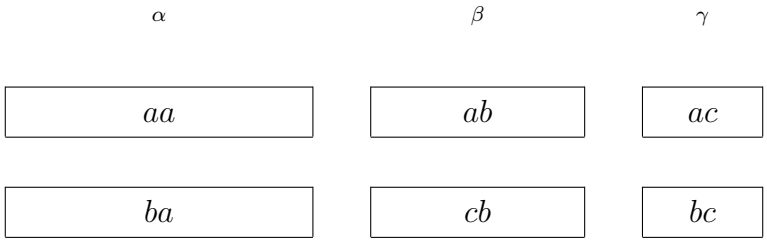

Figure 6: Edges outgoing from state α in G^2.

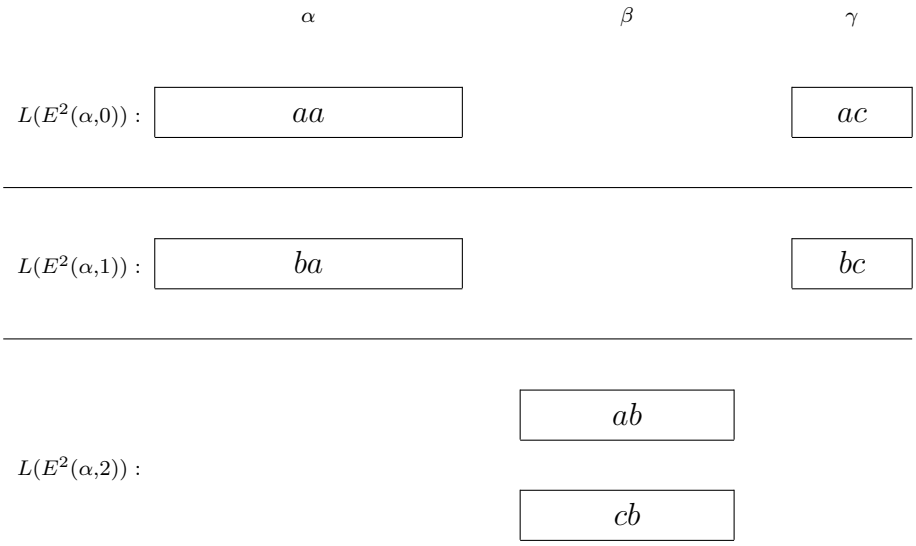

Figure 7: Partition of edges outgoing from state α in G^2.

0	1	2	3	0	1	2	3	0	1	2	3
$(\alpha,0)$				$(\alpha,1)$				$(\alpha,2)$			
$(\alpha,0)$	$(\alpha,1)$	$(\alpha,2)$	$(\gamma,0)$	$(\alpha,0)$	$(\alpha,1)$	$(\alpha,2)$	$(\gamma,0)$	$(\beta,0)$	$(\beta,1)$	$(\beta,0)$	$(\beta,1)$
aa	aa	aa	ac	ba	ba	ba	bc	ab	ab	cb	cb
$\longleftarrow$	$\mathcal{I}(\alpha;aa)$	$\longrightarrow$	$\mathcal{I}(\gamma;ac)$	$\longleftarrow$	$\mathcal{I}(\alpha;ba)$	$\longrightarrow$	$\mathcal{I}(\gamma;bc)$	$\leftarrow$	$\mathcal{I}(\beta;ab)$	$\rightarrow$	$\leftarrow$ $\mathcal{I}(\beta;cb)$ $\rightarrow$

Table 17: Array $\mathcal{C}(\alpha)$ associated with $\mathcal{E}$.

has the same length as the respective rectangle labeled $\mathbf{w}$ in Figures 6 and 7. The following observations are worth noting:

- The outgoing edges from state (α, i) are the edges in $\mathcal{E}$ that originate from the edges of $E^2(\alpha, i)$ in G^2; that is, for each edge $\alpha \xrightarrow{\mathbf{w}} v$ in $E^2(\alpha, i)$ and for each j, $0 \leq j \leq x_v - 1$, we have an edge $(\alpha, i) \xrightarrow{\mathbf{w}} (v, j)$ in $\mathcal{E}$. This follows from the equality $\mathbf{z}^{(u,i)} \cdot \mathbf{x} = \lambda(A_G^2) = 4$.

- Row 1 in $\mathcal{C}(\alpha)$ has a vertical double-line below each vertical single-line of row 0. This is a consequence of the fact that for every $u \in V$ and $0 \leq i \leq x_u - 1$, each unit-interval $\mathcal{C}_0(u, i)$ in row 0 lies above exactly $|E^2(u, i)| = \|\mathbf{z}^{(u,i)}\|_1$ (whole) super-intervals $\mathcal{I}(u; \mathbf{w})$ in row 1, where $\mathbf{w}$ ranges over all labels in $L(E^2(u, i))$. It thus follows that row 2 in the respective array $\mathcal{B}(u)$ (Table 2) has vertical double-lines at all points with abscissas of the form $m/n^2 = m/4$, $m = 0, 1, 2, \ldots$ (compare with Example 4). Hence, by Proposition 2 the encoder $\mathcal{E}$ has order 2. Equivalently, the existence of vertical double-lines in row 1 below each vertical single-line of row 0 suggests that $\mathcal{E}$ can be obtained by one round of the state-splitting algorithm applied to G^2 [5].

These features are demonstrated in Table 18 also for the array $\mathcal{C}(\beta)$ obtained by the ordering $aa < ac < ab < cb$ on $L(E^2(\beta))$ induced by the decomposition $(A_G^2)_\beta = \mathbf{z}^{(\beta,0)} + \mathbf{z}^{(\beta,1)}$. Similarly, the array $\mathcal{C}(\gamma)$ in Table 19 is obtained by the ordering $ba < bc$ on $L(E^2(\gamma))$.

We remark that the $(S^2, 4)$-encoder $\mathcal{E}$ we have just constructed is equivalent, up to tagging of edges, to the one obtained by squaring the $(S, 2)$-encoder $\mathcal{E}_3$ of Table 11 (or Figure 2).

	0	1	2	3	0	1	2	3
	$(\beta,0)$				$(\beta,1)$			
	$(\alpha,0)$	$(\alpha,1)$	$(\alpha,2)$	$(\gamma,0)$	$(\beta,0)$	$(\beta,1)$	$(\beta,0)$	$(\beta,1)$
	aa	aa	aa	ac	ab	ab	cb	cb
	$\longleftarrow$	$\mathcal{I}(\alpha;aa)$	$\longrightarrow$	$\mathcal{I}(\gamma;ac)$	$\leftarrow$	$\mathcal{I}(\beta;ab)$	$\rightarrow$	$\leftarrow$ $\mathcal{I}(\beta;cb)$ $\rightarrow$

Table 18: Array $\mathcal{C}(\beta)$ associated with $\mathcal{E}$.

	0	1	2	3
	$(\gamma,0)$			
	$(\alpha,0)$	$(\alpha,1)$	$(\alpha,2)$	$(\gamma,0)$
	ba	ba	ba	bc
	$\longleftarrow$	$\mathcal{I}(\alpha;ba)$	$\longrightarrow$	$\mathcal{I}(\gamma;bc)$

Table 19: Array $\mathcal{C}(\gamma)$ associated with $\mathcal{E}$.

6.2 Outline of the encoder construction

We now turn to a general constrained system S that is presented by a deterministic labeled graph $G = (V, E, L)$ such that $c(S) = \log \lambda(A_G) = \log n$ for an integer n. Let $\mathbf{x} = [x_u]_{u \in V}$ be a nonnegative integer (exact) right eigenvector of A_G associated with the eigenvalue n. As pointed out in Section 3, we can assume without loss of generality that all components of $\mathbf{x}$ are strictly positive. We further assume that the components of $\mathbf{x}$ do not have a common divisor, in which case $\|\mathbf{x}\|_\infty \leq n^{|V|-1}$ [5, Lemma 2].

As in the example of Section 6.1, the construction of an (S, n)-encoder involves finding an integer t and a decomposition of each row in A_G^t as a sum

$$(A_G^t)_u \;=\; \mathbf{z}^{(u,0)} + \mathbf{z}^{(u,1)} + \ldots + \mathbf{z}^{(u,x_u-1)} \quad , \; u \in V \; , \tag{6}$$

where each $\mathbf{z}^{(u,i)}$ is a nonnegative integer vector satisfying $\mathbf{z}^{(u,i)} \cdot \mathbf{x} = n^t$. We then obtain an (S^t, n^t)-encoder $\mathcal{E}$ of order 2 by applying the stethering method (or by applying one round of the state-splitting algorithm) to G^t, using a certain ordering on the set $E^t(u)$ of outgoing edges from state u in G^t; this ordering is defined as in Section 6.1 by means of a partition of $E^t(u)$ induced by the decomposition (6). The encoder $\mathcal{E}$, in turn, can be transformed into an (S, n)-encoder of order $\leq 3t$.

In [5], it was shown that a decomposition (6) can be obtained for $t = 3|V| + 2\lceil(\log|V|)/\log n\rceil$. The complexity of the task of finding a decomposition of A_G^t and representing a corresponding partition of $E^t(u)$, $u \in V$, was not addressed in [5]. This is the heart of the matter we address here. More specifically, we prove the following.

Theorem 4. *Let S be a constrained system presented by a deterministic labeled graph G such that $n = \lambda_G^q$ for positive integers q and n. Then there exists an (S^q, n)-encoder that can be implemented by a polynomial-size circuit, consisting of* $\mathrm{Poly}\big(|V_G|, q, \log|\Sigma|\big)$ *gates and $O\big(|V_G|\log n\big)$ memory bit-cells. Furthermore, the encoder has order $\leq 9|V_G| + 6\lceil(\log|V_G|)/(\log n)\rceil$ and can be decoded by a polynomial-size circuit.*

The polynomial-time algorithm to be described for implementing an (S^t, n^t)-encoder $\mathcal{E}$ consists of a preprocessing stage and an encoding stage. The preprocessing stage computes the vectors $\mathbf{z}^{(u,i)}$ in (6) and is carried out once and for all before any data is encoded.

Since the eigenvector components x_u can have size that is exponential in $|V|$, we cannot explicitly store each of the vectors $\mathbf{z}^{(u,i)}$. Rather, for each $u \in V$, we will choose a decomposition that is sufficiently regular to allow a polynomial-time computation of $\mathbf{z}^{(u,i)}$ as it is needed during the encoding process: Each of the vectors $\mathbf{z}^{(u,0)}, \mathbf{z}^{(u,1)}, \ldots, \mathbf{z}^{(u,x_u-1)}$ will actually be chosen from a set of at most $2|V| - 1$ distinct vectors $\mathbf{y}^{(u,0)}, \mathbf{y}^{(u,1)}, \ldots, \mathbf{y}^{(u,2|V|-2)}$. Furthermore, for each $u \in V$, the x_u vectors $\mathbf{z}^{(u,0)}, \mathbf{z}^{(u,1)}, \ldots, \mathbf{z}^{(u,x_u-1)}$ are ordered so that those equal to $\mathbf{y}^{(u,0)}$ come first, followed by those equal to $\mathbf{y}^{(u,1)}$, and so on. Thus, it will suffice to compute during the preprocessing stage and store just the vectors $\mathbf{y}^{(u,0)}, \mathbf{y}^{(u,1)}, \ldots, \mathbf{y}^{(u,2|V|-2)}$ and the multiplicity $m_{u,k}$ with which $\mathbf{y}^{(u,k)}$ occurs in the list $\mathbf{z}^{(u,0)}, \mathbf{z}^{(u,1)}, \ldots, \mathbf{z}^{(u,x_u-1)}$. We defer the argument that the preprocessing can be done in polynomial time until after a description of the encoding stage.

6.3 Encoding stage

For each edge from state u to state v in G^t, we assign an index θ, $0 \leq \theta \leq (A_G^t)_{u,v} - 1$, according to the lexicographic ordering of the words $\mathbf{w}$ that label the edges from u to v in G^t. It is easy to check that the index θ can be computed from the respective word $\mathbf{w}$, and vice versa, both in polynomial time (see Section 4).

For each state $u \in V$, we partition its set $E^t(u)$ of outgoing edges in G^t into the subsets $E^t(u,0), E^t(u,1), \ldots, E^t(u,x_u-1)$ so that the number of edges in $E^t(u,i)$ that terminate at state v is $(\mathbf{z}^{(u,i)})_v$. The subsets $E^t(u,i)$ are defined so that the $(\mathbf{z}^{(u,0)})_v$ edges in $E^t(u,0)$ leading to state v are lexicographically least, the $(\mathbf{z}^{(u,1)})_v$ edges in $E^t(u,1)$ leading to state v are lexicographically next, and so on. The ordering on $E^t(u)$ is defined using the subsets $E^t(u,i)$ as was illustrated in the example of Section 6.1. This way we obtain arrays $\mathcal{C}(u)$ which, in turn, define a stethering (S^t, n^t)-encoder $\mathcal{E}$ as follows: The states of $\mathcal{E}$ are given by (u,i), $u \in V$, $0 \le i \le x_u - 1$. Now, since $\mathbf{z}^{(u,i)} \cdot \mathbf{x} = n^t$, then the outgoing edges from (u,i) are exactly those edges in $\mathcal{E}$ that originate from the edges in $E^t(u,i)$ in G^t; namely, for each edge $u \xrightarrow{\mathbf{w}} v$ in $E^t(u,i)$ and for each j, $0 \le j \le x_v - 1$, we have an edge $(u,i) \xrightarrow{\mathbf{w}} (v,j)$ in $\mathcal{E}$. Furthermore, since $|E^t(u,i)| = \|\mathbf{z}^{(u,i)}\|_1$, then there is a vertical double-line in row 1 of $\mathcal{C}(u)$ below each vertical single-line in row 0 and, so, by Proposition 2, the encoder $\mathcal{E}$ has order 2.

It is also worth recalling that the (whole number of) super-intervals $\mathcal{I}(u;\mathbf{w})$ below each unit-interval $\mathcal{C}_0(u,i)$ in $\mathcal{C}(u)$ are arranged according to the ordering of $v = \tau(u;\mathbf{w})$ in V. In particular, the super-intervals $\mathcal{I}(u;\mathbf{w})$ underneath $\mathcal{C}_0(u,i)$ that correspond to edges $u \xrightarrow{\mathbf{w}} v \in E^t(u,i)$ with the same terminal state v in G^t appear next to each other in row 1 of $\mathcal{C}(u)$.

Given a present state (u,i) in $\mathcal{E}$ and a source tag s, $0 \le s \le n^t - 1$, we will describe the computation of the next state (v,j) in $\mathcal{E}$ and the labeling of the edge from state (u,i) tagged by s. The labeling of the edge will be described by the index θ, $0 \le \theta \le (A_G^t)_{u,v} - 1$, of the edge in G^t from state u to state v from which the edge in $\mathcal{E}$ originates. Note that we must have

$$\sum_{k=0}^{i-1} (\mathbf{z}^{(u,k)})_v \;\le\; \theta \;<\; \sum_{k=0}^{i} (\mathbf{z}^{(u,k)})_v \,.$$

Step 1. Find the smallest nonnegative index $\ell \le 2|V| - 2$ such that $i < \sum_{k \le \ell} m_{u,k}$. At this point we know $\mathbf{z}^{(u,i)} = \mathbf{y}^{(u,\ell)}$ and $\sum_{k=0}^{i-1} \mathbf{z}^{(u,k)} = \sum_{k<\ell} (m_{u,k}\mathbf{y}^{(u,k)}) + (i - \sum_{k<\ell} m_{u,k})\mathbf{y}^{(u,\ell)}$.

Step 2. Find the smallest state $v \in V$ such that $s < \sum_{v' \le v} (\mathbf{z}^{(u,i)})_{v'}\, x_{v'}$ and let $r = s - \sum_{v'<v} (\mathbf{z}^{(u,i)})_{v'}\, x_{v'}$. The next state is (v,j) where $j \equiv r \pmod{x_v}$ and $\theta = \lfloor r/x_v \rfloor + \sum_{k=0}^{i-1} (\mathbf{z}^{(u,k)})_v$.

Thus we can compute (v,j) and θ in polynomial time using the results stored during preprocessing. As for decoding, it can be readily verified that, given an initial state (u,i)

and the labeling, over Σ^t, of a path π of length 2 from (u,i) in $\mathcal{E}$, the first edge in π can be reconstructed in polynomial time.

6.4 Preprocessing stage

The preprocessing stage consists of three steps. In Step 1 we compute a set of $D \leq |V|^2$ nonnegative integer vectors $\mathbf{a}^{(0)}, \mathbf{a}^{(1)}, \ldots, \mathbf{a}^{(D-1)}$, each satisfying $\mathbf{a}^{(k)} \cdot \mathbf{x} = n^{t_1+|V|-1}$, where $t_1 = 2|V| + \lceil (\log |V|)/\log n \rceil$ and where the rows of A_G^ℓ are nonnegative combinations of the $\mathbf{a}^{(k)}$'s for every $\ell \geq |V|$. Step 1 appears in [5] and is included here for the sake of completeness. Then in Steps 2 and 3 we compute the vectors $\mathbf{y}^{(u,0)}, \mathbf{y}^{(u,1)}, \ldots, \mathbf{y}^{(u,2|V|-2)}$ and the multiplicities $m_{u,k}$ (see Section 6.2) out of the vectors $\mathbf{a}^{(0)}, \mathbf{a}^{(1)}, \ldots, \mathbf{a}^{(D-1)}$. All three steps can be carried out in polynomial time.

Step 1. Calculate the integer vectors $\mathbf{a}^{(0)}, \mathbf{a}^{(1)}, \ldots, \mathbf{a}^{(D-1)}$ as follows.

Fix $t_1 = 2|V| + \lceil (\log |V|)/\log n \rceil$ (so that $n^{t_1} \geq 2(|V|-1)\|\mathbf{x}\|_\infty^2$ [5, Lemma 2]). For each state $u \in V$, calculate a set $\mathcal{A}_u$ of nonnegative integer vectors whose convex hull contains the row vector $\mathbf{f}_u = (n^{t_1}/x_u)(A_G^{|V|-1})_u$ as follows.

First use the Euclidean Algorithm to find an integer lattice point $\mathbf{h}$ in the set

$$Q \;=\; \left\{ \mathbf{y} \in \mathbb{R}^{|V|} \;\middle|\; \mathbf{y} \cdot \mathbf{x} = n^{|V|-1} \quad \text{and} \quad (A_G^{|V|-1})_{u,v} = 0 \Rightarrow (\mathbf{y})_v = 0 \right\} .$$

We can guarantee that the components of $\mathbf{h}$ do not exceed $n^{|V|(|V|-1)}$.

Let $\mathbf{r}^{(1)}, \mathbf{r}^{(1)}, \ldots, \mathbf{r}^{(\ell)}$ be a rational basis for $Q - \mathbf{h}$ consisting of integer vectors $\mathbf{r}^{(j)}$ such that $\|\mathbf{r}^{(j)}\|_1 \leq 2\|\mathbf{x}\|_\infty$. For instance one can choose $\mathbf{r}^{(j)}$ to have only two nonzero coordinates x_v and $-x_{v'}$, and these at positions v' and v, respectively. Solve the equations

$$\mathbf{f}_u - n^{t_1}\mathbf{h} \;=\; \sum_{j=1}^{\ell} c_j \mathbf{r}^{(j)}$$

for the rational coefficients c_j. Set $\mathbf{b} = n^{t_1}\mathbf{h} + \sum_{j=1}^{\ell} \lfloor c_j \rfloor \mathbf{r}^{(j)}$.

Now $\mathbf{f}_u$ is in the parallelotope whose extreme points are the set of 2^ℓ points obtained by adding any $\{0,1\}$-combination of $\mathbf{r}^{(1)}, \mathbf{r}^{(2)}, \ldots, \mathbf{r}^{(\ell)}$ to $\mathbf{b}$. These extreme points are nonnegative integer lattice points. We find $\ell+1$ of these extreme points whose convex hull contains $\mathbf{f}_u$ as follows.

Set $d_j = c_j - \lfloor c_j \rfloor$. Sort the numbers $d_1, d_2, \ldots, d_\ell$, finding a permutation π such that $d_{\pi(1)} \geq d_{\pi(1)} \geq \ldots \geq d_{\pi(\ell)}$.

Set $\mathbf{a}^{(u,0)} = \mathbf{b}$ and $\mathbf{a}^{(u,i)} = \mathbf{b} + \sum_{k=1}^{i} \mathbf{r}^{(\pi(k))}$. Then the convex hull of $\mathcal{A}_u = \{\mathbf{a}^{(u,i)} \mid 0 \leq i \leq \ell\}$, contains $\mathbf{f}_u$.

Finally let $\{\mathbf{a}^{(0)}, \mathbf{a}^{(1)}, \ldots, \mathbf{a}^{(D-1)}\}$ be the union of $\mathcal{A}_u$ over all states $u \in V$.

Step 2. Set $t_2 = \lceil (\log |V|)/\log n \rceil$, $M = n^{t_2}$, and $t = t_1 + t_2 + |V| - 1$.

For each $u \in V$, use linear programming [18] to find a nonnegative rational vector $\mathbf{g}_u = [\, g_{u,0}\ g_{u,1}\ \ldots\ g_{u,D-1}\,]$ such that

$$(A_G^t)_u \;=\; \sum_{k=0}^{D-1} g_{u,k} M \mathbf{a}^{(k)} \;. \tag{7}$$

Furthermore, we can assume that $\mathbf{g}_u$ contains at most $|V|$ nonzero entries; otherwise we can eliminate entries in $\mathbf{g}_u$ by successively substituting $\mathbf{g}_u$ for $\mathbf{g}_u + \mathbf{s} \geq \mathbf{0}$, where $\mathbf{s} = [s_k]_{k=0}^{D-1}$ are rational vectors such that $\sum_{k=0}^{D-1} s_k \mathbf{a}^{(k)} = \mathbf{0}$.

Re-order the vectors $\mathbf{a}^{(k)}$ so that the first entries of $\mathbf{g}_u$ are the nonzero ones and set $m_{u,k} = \lfloor g_{u,k} \rfloor$ and $\mathbf{y}^{(u,k)} = M\mathbf{a}^{(k)}$ for $0 \leq k \leq |V| - 1$.

Notice that $\mathbf{y}^{(u,k)} \cdot \mathbf{x} = n^t$ for $u \in V$ and $0 \leq k \leq |V| - 1$. Also notice that

$$\sum_{k=0}^{|V|-1} g_{u,k} \;=\; (1/n^t) \sum_{k=0}^{|V|-1} g_{u,k} M \mathbf{a}^{(k)} \cdot \mathbf{x} \;=\; (1/n^t)\,(A_G^t)_u \cdot \mathbf{x} \;=\; x_u \quad \text{for every } u \in V \;.$$

Now, by (7),

$$(A_G^t)_u - \sum_{k=0}^{|V|-1} m_{u,k} \mathbf{y}^{(u,k)} \;=\; \sum_{k=0}^{|V|-1} (g_{u,k} - m_{u,k}) M \mathbf{a}^{(k)} \;. \tag{8}$$

Furthermore, for every $u \in V$, the number R_u defined by

$$R_u \;=\; \sum_{k=0}^{|V|-1} (g_{u,k} - \lfloor g_{u,k} \rfloor) \;=\; \sum_{k=0}^{|V|-1} g_{u,k} - \sum_{k=0}^{|V|-1} m_{u,k} \;=\; x_u - \sum_{k=0}^{|V|-1} m_{u,k} \tag{9}$$

is an integer smaller than $|V|$. Thus, it suffices to decompose the difference $(A_G^t)_u - \sum_{k=0}^{|V|-1} m_{u,k} \mathbf{y}^{(u,k)}$ into a sum of R_u nonnegative integer vectors $\mathbf{y}^{(u,|V|+k)}$, $0 \leq k \leq R_u - 1$, each satisfying $\mathbf{y}^{(u,k)} \cdot \mathbf{x} = n^t$. We do this in Step 3.

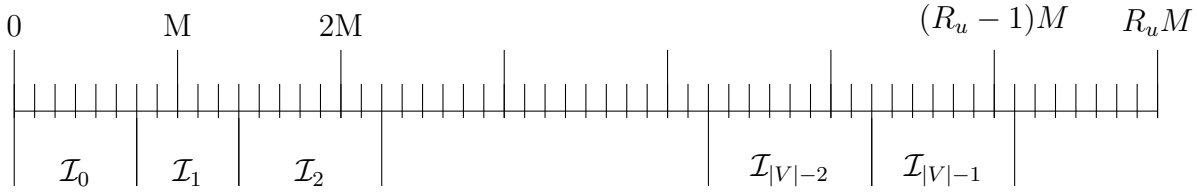

Figure 8: Intervals $\mathcal{I}_j$ on the integer number line.

Step 3. It is easiest to explain this step using the integer number line. Starting at 0, for each $u \in V$ lay down intervals $\mathcal{I}_0, \mathcal{I}_1, \ldots, \mathcal{I}_{|V|-1}$, where $\mathcal{I}_j$ has length $\lfloor (g_{u,j} - m_{u,j})M \rfloor$, and $\mathcal{I}_{j+1}$ begins where $\mathcal{I}_j$ ends (see Figure 8).

Define the nonnegative integer vectors $\mathbf{y}^{(u,|V|+k)}$, $0 \le k \le R_u - 2$, by

$$\mathbf{y}^{(u,|V|+k)} \;=\; \sum_{j=0}^{|V|-1} |\,\mathcal{I}_j \cap [kM, kM+M]\,|\,\mathbf{a}^{(j)}\;,$$

where $|\cdot|$ denotes length of intervals. Also, define the integer vector $\mathbf{y}^{(u,|V|+R_u-1)}$ by

$$\mathbf{y}^{(u,|V|+R_u-1)} \;=\; (A_G^t)_u \;-\; \sum_{k=0}^{|V|-1} (m_{u,k}\mathbf{y}^{(u,k)}) \;-\; \sum_{k=0}^{R_u-2} \mathbf{y}^{(u,|V|+k)}\;.$$

We complete the preprocessing by setting $m_{u,|V|+k} = 1$ for $0 \le k \le R_u - 1$.

We first verify that $\mathbf{y}^{(u,|V|+R_u-1)}$ is a nonnegative vector. Indeed, by (8),

$$\begin{aligned}
\mathbf{y}^{(u,|V|+R_u-1)} &= (A_G^t)_u \;-\; \sum_{k=0}^{|V|-1} (m_{u,k}\mathbf{y}^{(u,k)}) \;-\; \sum_{k=0}^{R_u-2} \mathbf{y}^{(u,|V|+k)} \\
&\ge \sum_{k=0}^{|V|-1} ((g_{u,k} - m_{u,k})M\mathbf{a}^{(k)}) \;-\; \sum_{j=0}^{|V|-1} \lfloor (g_{u,j} - m_{u,j})M \rfloor \mathbf{a}^{(j)} \\
&\ge 0\,.
\end{aligned}$$

Clearly, the vectors $\mathbf{y}^{(u,|V|+k)}$, $0 \le k \le R_u - 1$, can be computed in polynomial-time. It therefore remains to verify that each of these vectors dots with $\mathbf{x}$ to yield n^t. By (9),

$$\begin{aligned}
R_uM - \sum_{j=0}^{|V|-1} |\,\mathcal{I}_j\,| &= \sum_{k=0}^{|V|-1} ((g_{u,k} - m_{u,k})M - \lfloor (g_{u,k} - m_{u,k})M \rfloor) \\
&< |V| \;\le\; M\;,
\end{aligned}$$

implying that the right-hand end point of $\mathcal{I}_{|V|-1}$ is between $(R_u-1)M$ and R_uM. From this it follows that each interval $[kM, kM+M]$, $0 \le k \le R_u-2$, is covered by $\mathcal{I}_j$'s. So,

$$\mathbf{y}^{(u,|V|+k)} \cdot \mathbf{x} \;=\; |\,[kM, kM+M]\,|\, n^{t_1+|V|-1} \;=\; n^t\;, \quad 0 \le k \le R_u-2\;. \tag{10}$$

As for $\mathbf{y}^{(u,|V|+R_u-1)}$, by (8), (9), and (10), we have

$$\begin{aligned}\mathbf{y}^{(u,|V|+R_u-1)} \cdot \mathbf{x} \;&=\; (A_G^t)_u \cdot \mathbf{x} - \sum_{k=0}^{|V|-1} (m_{u,k}\mathbf{y}^{(u,k)} \cdot \mathbf{x}) - \sum_{k=0}^{R_u-2} \mathbf{y}^{(u,|V|+k)} \cdot \mathbf{x} \\ &=\; R_u\, n^t - (R_u-1)\, n^t \;=\; n^t\;,\end{aligned}$$

as desired.

7 Summary of attainable rates and orders

The following theorem classifies the properties of encoders obtained by the constructions of Sections 3, 5, and 6 according to their rate range.

Theorem 5. *Let S be a constrained system presented by a deterministic labeled graph G and let p and q be positive integers. Then, for the specified rate range, there exists a fixed-rate $p:q$ encoder over Φ into S such that —*

- *when $(p/q)\log|\Phi| \le c(S)$, the resulting encoder can be implemented by a polynomial-size circuit, consisting of $\mathrm{Poly}\big(|V_G|, q, \log|\Sigma|\big)$ gates and $O\big(|V_G|\,p\log|\Phi|\big)$ memory bit-cells (Theorem 3);*
- *when $(p/q)\log|\Phi| = c(S)$, the encoder has order which is bounded from above by $9|V_G| + 6\lceil(\log|V_G|)/(p\log|\Phi|)\rceil$ and can be decoded by a polynomial-size circuit (Theorem 4);*
- *when $(p/q)\log|\Phi| < c(S)$, the encoder has order $\le 12|V_G|$ and can be decoded by a polynomial-size circuit (Section 5.2);*
- *when $(p/q)\log|\Phi| \le c(S) - ((\log e)/(|\Phi|^p q))$, the upper bound on the order becomes $2|V_G|+2$ (Corollary 1);*

- *and when* $(p/q)\log|\Phi| \leq c(S) - ((\log e)/(|\Phi|^p q))$ *and* G *is of finite memory* $\mu(G)$*, the encoder is* $(\lceil \mu(G)/q \rceil, 2|V_G| + 1)$*-sliding-block decodable by a polynomial-size decoder (Corollary 2).*

Furthermore, there exists a program on a RAM which generates the layouts of all circuits in polynomial time.

8 Pseudo-stethering encoders

In this section, we discuss a natural generalization of stethering encoders called *pseudo-stethering*. This provides more flexibility in designing encoders.

8.1 Definition of pseudo-stethering encoders

We describe pseudo-stethering encoders as follows.

As before, we start with a deterministic presentation G of S, and we specify an (A_G, n)-approximate eigenvector $\mathbf{x}$ and an arbitrary pre-assigned ordering on each set of outgoing labels $L(E(u))$. These, in turn, define arrays $\mathcal{C}(u)$, $u \in V_G$, as shown in Table 1. Row 1 of $\mathcal{C}(u)$ contains $(A_G\mathbf{x})_u$ sub-intervals $\mathcal{C}_1(u; a; j)$ of length $1/n$, ordered lexicographically using the preassigned ordering on $a \in L(E(u))$ and the usual ordering on the integers j in the range $0 \leq j \leq x_{\tau(u;a)} - 1$.

Let $\mathcal{C}_1(u)$ denote the ordered set of sub-intervals of row 1 of $\mathcal{C}(u)$. Clearly, $|\mathcal{C}_1(u)| = (A_G\mathbf{x})_u \geq nx_u$. We now choose a subset $\mathcal{B}_1(u) \subseteq \mathcal{C}_1(u)$ of size $|\mathcal{B}_1(u)| = nx_u$. The lexicographic ordering on $\mathcal{C}_1(u)$ induces an ordering on $\mathcal{B}_1(u)$. Now, $\mathcal{B}_1(u)$ is naturally partitioned into subsets $\mathcal{B}_1(u,0), \mathcal{B}_1(u,1), \ldots, \mathcal{B}_1(u, x_u - 1)$, where subset $\mathcal{B}_1(u,0)$ consists of the first n elements of $\mathcal{B}_1(u)$, subset $\mathcal{B}_1(u,1)$ consists of the next n elements of $\mathcal{B}_1(u)$, and so on. Then, each $\mathcal{B}_1(u,i)$ naturally inherits an ordering.

The set of states of the pseudo-stethering encoder $\mathcal{E}$ is

$$V_{\mathcal{E}} \;=\; \Big\{(u,i) \;\Big|\; u \in V_G,\;\; 0 \leq i \leq x_u - 1\Big\}\,.$$

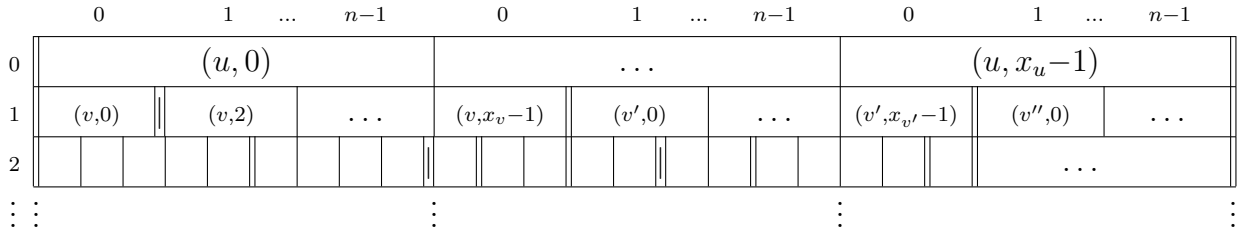

Table 20: Array $\mathcal{B}(u)$ for pseudo-stethering encoders.

For each $0 \leq i \leq x_u - 1$ and $\mathcal{C}_1(u; a; j) \in \mathcal{B}_1(u, i)$, set one edge labeled a from (u, i) to (v, j) where $v = \tau(u; a)$. We have now completely defined $\mathcal{E}$.

For each state $u \in V_G$, we define the infinite array $\mathcal{B}(u)$ as shown in Table 20. Row 0 in $\mathcal{B}(u)$ is identical to row 0 in $\mathcal{C}(u)$ and is divided into x_u unit-intervals $\mathcal{C}_0(u, i)$, $0 \leq i \leq x_u - 1$. Following our previous convention, these unit-intervals are represented in Table 20 as boxes of length 1 labeled by their respective states (u, i). Row 1 consists of the sub-intervals $\mathcal{C}_1(u; a; j) \in \mathcal{B}_1(u)$ of length $1/n$ which are laid according to their lexicographic order. Note that for each i, the n sub-intervals of $\mathcal{B}_1(u, i)$ are located right underneath the unit-interval $\mathcal{C}_0(u, i)$. Each sub-interval $\mathcal{C}_1(u; a; j)$ appears in Table 20 as a box of length $1/n$ labeled by (v, j), where $v = \tau(u; a)$ (the symbol a will sometimes be written underneath the box). For a fixed symbol $a \in L(E(u))$, the sub-intervals $\mathcal{C}_1(u; a; j) \in \mathcal{B}_1(u)$ form a contiguous super-interval $\mathcal{I}(u; a)$.

Subsequent rows in $\mathcal{B}(u)$ are subdivided and marked by down-scaling much as we described in Section 3.2 (see also the construction of $\mathcal{B}^*(u)$ in Section 5). Row r in $\mathcal{B}(u)$ is therefore of length x_u and is divided into $n^r x_u$ (down-scaled) unit-intervals, each of length $1/n^r$. These intervals are separated in each row of $\mathcal{B}(u)$ by vertical single-lines except that vertical double-lines are used to separate two (down-scaled) super-intervals $\mathcal{I}(u; a)$.

By a *gap* we mean a vertical single-line that separates a (down-scaled) sub-interval $\mathcal{C}_1(u; a; j)$ from $\mathcal{C}_1(u; a; j')$ where $j' \neq j \pm 1$. Gaps are marked in Table 20 by $\Big|\Big|$.

Note that every stethering encoder is a pseudo-stethering encoder in which $\mathcal{B}_1(u)$ is chosen to be the first nx_u elements of $\mathcal{C}_1(u)$. And by the description of punctured stethering encoders through the arrays $\mathcal{B}^*(u)$ of Section 5 it follows that every punctured stethering

encoder is also a pseudo-stethering encoder.

It is not hard to generalize the proof of losslessness (in Section 3) from stethering encoders to pseudo-stethering encoders. The polynomiality results (in Section 4) hold for those pseudo-stethering encoders provided we can count the number of sub-intervals in $\mathcal{C}_1(u)$ that were deleted up to any given point in row 1 of $\mathcal{B}(u)$ by a polynomial-time algorithm.

8.2 Losslessness of finite order of pseudo-stethering encoders

The following proposition formalizes a key feature of pseudo-stethering encoders that we used implicitly in the proof of Proposition 3.

Proposition 4. *Let S be a constrained system presented by a deterministic labeled graph G and let $\mathcal{E}$ be a pseudo-stethering (S, n)-encoder obtained from G using an (A_G, n)-approximate eigenvector $\mathbf{x}$. Then, there is an integer K such that for every path $u_0 \stackrel{w_1}{\to} u_1 \stackrel{w_2}{\to} \ldots \stackrel{w_\ell}{\to} u_\ell$ of length $\ell \geq K$ in G, any two respective paths*

$$\pi \;=\; (u_0, i_0) \stackrel{w_1}{\longrightarrow} (u_1, i_1) \stackrel{w_2}{\longrightarrow} \cdots \stackrel{w_\ell}{\longrightarrow} (u_\ell, i_\ell)$$

and

$$\pi' \;=\; (u_0, i'_0) \stackrel{w_1}{\longrightarrow} (u_1, i'_1) \stackrel{w_2}{\longrightarrow} \cdots \stackrel{w_\ell}{\longrightarrow} (u_\ell, i'_\ell)$$

in $\mathcal{E}$ must have $|i'_0 - i_0| \leq 1$.

Proof. Let $i < i'$, $u \in V_G$, and suppose that there is a label $a \in L(E(u))$ such that for some j and j' we have $\mathcal{C}_1(u; a; j) \in \mathcal{B}_1(u, i)$ and $\mathcal{C}_1(u; a; j') \in \mathcal{B}_1(u, i')$. Table 20 then makes it clear that $j < j'$ and that the distance between the unit-intervals $\mathcal{C}_0(u, i)$ and $\mathcal{C}_0(u, i')$ in row 0 is bounded from above by the distance between the sub-intervals $\mathcal{C}_1(u; a; j)$ and $\mathcal{C}_1(u; a; j')$ in row 1. It then follows that

$$i' - i - 1 \;\leq\; (j' - j - 1)/n \;.$$

This inequality can be re-written as

$$i' - i \;\leq\; (j' - j)/n \;+\; (n - 1)/n \;. \tag{11}$$

It follows inductively from (11) that

$$|i'_0 - i_0| \;\leq\; \|\mathbf{x}\|_\infty/n^\ell + (n-1)\sum_{j=1}^{\ell}(1/n^j) \;<\; (\|\mathbf{x}\|_\infty/n^\ell) + 1\;. \tag{12}$$

Hence, for $\ell \geq K = \lceil(\log \|\mathbf{x}\|_\infty)/\log\, n\rceil$ we have $|i'_0 - i_0| < 2$, as desired. □

The next proposition is an extension of Proposition 2 for pseudo-stethering encoders.

Proposition 5. *Let S be a constrained system presented by a deterministic labeled graph G and let $\mathcal{E}$ be a pseudo-stethering (S,n)-encoder. Then, $\mathcal{E}$ is lossless of finite order if and only if there is a positive integer N such that: for each array $\mathcal{B}(u)$ and for each vertical single-line $\mathcal{L}$ that appears in row 1 of $\mathcal{B}(u)$, but not immediately below a vertical line in row 0, if we extend $\mathcal{L}$ downward through the array, then we must hit either a vertical double-line or a gap in some row $r \leq N$.*

If such an integer N exists, then $\mathcal{O}(\mathcal{E}) \leq N + K$, where K is the integer guaranteed by Proposition 4.

Proof. The proof is similar to that of Proposition 2. Assume that such an integer N exists. Let

$$\pi \;=\; (u_0, i_0) \xrightarrow{w_1} (u_1, i_1) \xrightarrow{w_2} \cdots \xrightarrow{w_{N+K}} (u_{N+K}, i_{N+K})$$

and

$$\pi' \;=\; (u'_0, i'_0) \xrightarrow{w_1} (u'_1, i'_1) \xrightarrow{w_2} \cdots \xrightarrow{w_{N+K}} (u'_{N+K}, i'_{N+K})$$

be two paths of length $N+K$ in $\mathcal{E}$ with the same initial state $(u_0, i_0) = (u'_0, i'_0)$ that generate the same word $\mathbf{w} = w_1 w_2 \ldots w_{N+K}$. Since G is deterministic, we have $u_0 u_1 \ldots u_{N+K} = u'_0 u'_1 \ldots u'_{N+K}$. Furthermore, from the choice of K we have, from Proposition 4, $|i'_r - i_r| \leq 1$ for every $r = 1, 2, \ldots, N$.

We show that $(u_1, i_1) = (u'_1, i'_1)$. Suppose, to the contrary, that $i_1 \neq i'_1$. As in the proof of Proposition 2, let $\mathcal{L}$ denote the vertical single-line in row 1 of $\mathcal{B}(u_0)$ that separates the sub-interval $\mathcal{C}_1(u_0; w_1; i_1)$ from $\mathcal{C}_1(u_0; w_1; i'_1)$. The upward extension of $\mathcal{L}$ meets the interior of the unit-interval $\mathcal{C}_0(u_0, i_0)$ in row 0 of $\mathcal{B}(u_0)$. Let $r \leq N$ be an index of a row in $\mathcal{B}(u_0)$ in which the downward extension of $\mathcal{L}$ hits either a vertical double-line or a gap. Then, the (down-scaled) sub-intervals $\mathcal{C}_1(u_{r-1}; w_r; i_r)$ and $\mathcal{C}_1(u_{r-1}; w_r; i'_r)$ are separated in this row by a

vertical double-line or a gap. But this is impossible since $|i'_r - i_r| \leq 1$ and these sub-intervals belong to the same (down-scaled) super-interval $\mathcal{I}(u_{r-1}; w_r)$. Thus, $i_1 = i'_1$ as desired.

The proof of existence of an integer N given that $\mathcal{E}$ is lossless of finite order is essentially the same as that in Proposition 2. □

It follows from the proof of Proposition 4 that when $\mathcal{E}$ is a pseudo-stethering encoder based on an (A_G, n)-approximate eigenvector $\mathbf{x}$, we can take the integer K to be $\lceil (\log \|\mathbf{x}\|_\infty)/\log n \rceil$. And, when $\mathcal{E}$ is a punctured stethering (S, n)-encoder we can take $N = 2$. So, for punctured stethering encoders Proposition 5 implies the bound $\mathcal{O}(\mathcal{E}) \leq 2 + \lceil (\log \|\mathbf{x}\|_\infty)/\log n \rceil$ (compare with Proposition 3).

The following result shows that for irreducible stethering encoders (i.e., stethering encoders whose underlying graph is irreducible) the criterion for losslessness of finite order of Proposition 5 (or, rather, of Proposition 2) can be simplified.

Proposition 6. *Let $\mathcal{E}$ be an irreducible stethering (S, n)-encoder. Then $\mathcal{E}$ is lossless of finite order if and only if for each vertical single-line $\mathcal{L}$ that appears in row 0 of any array $\mathcal{B}(u)$, the downward extension of $\mathcal{L}$ must hit a vertical double-line.*

Proof. Each vertical single-line in row 1 of some array is a copy of a vertical single-line which appears in row 0 (of perhaps some other array). So, if the condition of the proposition holds, then it also holds for all vertical single-lines in row 1. Sufficiency of the condition then follows from Proposition 2.

For necessity, it suffices, by Proposition 2, to show that if $\mathcal{E}$ is lossless of finite order, then for each vertical single-line $\mathcal{L}$ in row 0 of any array $\mathcal{B}(u)$, there is a copy $\mathcal{L}'$ of $\mathcal{L}$ which appears in some row of some array $\mathcal{B}(u')$ such that the upward extension of $\mathcal{L}'$ meets the interior of some unit-interval in row 0 of $\mathcal{B}(u')$. Say that $\mathcal{L}$ separates the unit-intervals $\mathcal{C}_0(v, j)$ and $\mathcal{C}_0(v, j+1)$ in $\mathcal{B}(u)$. Since $\mathcal{E}$ is irreducible, it follows that for any state $(u', i) \in V_\mathcal{E}$, there is a positive integer k and at least two distinct paths in $\mathcal{E}$ from (u', i) to $(v, j+1)$ of length k (indeed, when $n > 1$, we can always find two distinct paths of the same length from state (u', i) to some state (u'', i'); then, by irreducibility, there is a path from (u'', i') to $(v, j+1)$). Thus, there are at least two distinct copies (down-scaled by a factor of n^k) of the unit-interval $\mathcal{C}_0(v, j+1)$ in row k of $\mathcal{B}(u')$ which are entirely contained within the downward extension

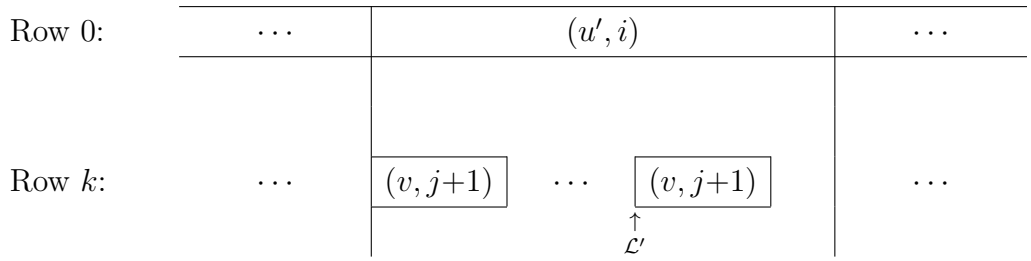

Table 21: Illustrating the proof of Proposition 6.

of the unit-interval $\mathcal{C}_0(u', i)$ in row 0, as shown in Table 21 (the copies of $\mathcal{C}_0(v, j+1)$ are labeled, as usual, by $(v, j+1)$). The right copy of $\mathcal{C}_0(v, j+1)$ must contain the desired copy $\mathcal{L}'$ of $\mathcal{L}$. □

8.3 Sliding-block decodability of pseudo-stethering encoders

In the framework of pseudo-stethering encoders, we show here how, for constrained systems of finite memory, there is a way of tagging the encoder such that it is lossless of finite order if and only if it is sliding-block decodable.

The *usual* way to tag an (S, n)-encoder $\mathcal{E}$ would be to assign the input tags $\{0, 1, \ldots, n-1\}$ to each $\mathcal{B}_1(u, i)$ in the order-preserving way (with respect to the ordering on $\mathcal{B}_1(u, i)$ inherited from $\mathcal{B}_1(u)$). Instead of doing that, we will assign the input tags alternately in the order-preserving way and the order-reversing way; to be definite, say that we use the order-preserving way when i is even and the order-reversing way when i is odd (see Table 22). When an encoder is tagged in this way, we say that it is *Gray-tagged*. Indeed, let $\mathbf{w} = w_1 w_2 \ldots w_\ell$ be a word of length ℓ generated by a path

$$\pi \;=\; (u_0, i_0) \xrightarrow{w_1} (u_1, i_1) \xrightarrow{w_2} \cdots \xrightarrow{w_\ell} (u_\ell, i_\ell)$$

in $\mathcal{E}$ and let $\mathbf{s} = s_1 s_2 \ldots s_\ell$ be the source Gray-tag sequence associated with the edges of π. We can associate with π a point Q_π in $[0, x_{u_0})$ as we did in Section 3.2: Using Gray tagging, the abscissa of Q_π will be given by $i_0 + q_\pi$, where q_π is the rational whose *Gray-code representation to base* n equals $0\,.\,s_1 s_2 \ldots s_\ell$ [19, Ch. 1].

While losslessness of finite order is a property of un-tagged encoders and sliding-block decodability is a property of tagged encoders, the latter property always implies the former.

	0	1 ... $n-1$	$n-1$	$n-2$ ... 0	0	1 ... $n-1$
0	$(u,0)$		...		(u,x_u-1)	
1	$(v,0)$	$(v,2)$ ...	(v,x_v-1)	$(v',0)$ ...	$(v',x_{v'}-1)$	$(v'',0)$ ...

Table 22: Gray tagging of pseudo-stethering encoders.

For constrained systems of finite memory and Gray-tagged pseudo-stethering encoders, the following result shows that the two properties are actually equivalent.

Proposition 7. *Let S be presented by a deterministic labeled graph G of finite memory and let $\mathcal{E}$ be a Gray-tagged pseudo-stethering (S,n) encoder obtained from G. Then $\mathcal{E}$ is lossless of finite order if and only if $\mathcal{E}$ is sliding-block decodable.*

Proof. Suppose that $\mathcal{E}$ is lossless of finite order. Let $m = \mu(G)$, N as in Proposition 5 and K as in Proposition 4. Let $\mathbf{w} = w_{-m+1}w_{-m+2}\dots w_{N+K}$ be a word of length $m+N+K$ in S. We will show that if

$$\pi \;=\; (u_{-m}, i_{-m}) \overset{w_{-m+1}}{\longrightarrow} (u_{-m+1}, i_{-m+1}) \overset{w_{-m+2}}{\longrightarrow} \cdots \overset{w_{N+K}}{\longrightarrow} (u_{N+K}, i_{N+K})$$

and

$$\pi' \;=\; (u'_{-m}, i'_{-m}) \overset{w_{-m+1}}{\longrightarrow} (u'_{-m+1}, i'_{-m+1}) \overset{w_{-m+2}}{\longrightarrow} \cdots \overset{w_{N+K}}{\longrightarrow} (u'_{N+K}, i'_{N+K})$$

are two paths in $\mathcal{E}$ which generate $\mathbf{w}$, then the edges in both paths which generate w_1 must have the same input tag; so, the encoder will then be $(m, N+K-1)$-sliding-block decodable.

Since $m = \mu(G)$ we have $u_0u_1\dots u_{N+K} = u'_0u'_1\dots u'_{N+K}$. Now, consider the following two cases:

Case 1: $i_0 = i'_0$. By Proposition 5, the order of $\mathcal{E}$ is bounded from above by $N+K$. Hence, we must have $i_1 = i'_1$ and, therefore, w_1 is generated by the same edge in both paths and thus we have the same input tag.

Case 2: $i_0 \neq i'_0$. ¿From the choice of K and the fact that π and π' generate the same word $\mathbf{w}$, we have $|i'_r - i_r| \leq 1$ for $0 \leq r \leq N$. We may assume $i'_0 = i_0 + 1$. Then $i'_1 = i_1 + 1$, the sub-interval $\mathcal{C}_1(u_0; w_1; i_1)$ must be the last element of $\mathcal{B}_1(u_0, i_0)$ and $\mathcal{C}_1(u_0; w_1; i'_1)$ must

be the first element of $\mathcal{B}_1(u_0, i'_0)$; since the encoder is Gray-tagged, it follows that the edge $(u_0, i_0) \stackrel{w_1}{\to} (u_1, i_1)$ in $\mathcal{E}$ has the same input tag (either 0 or $n-1$) as the edge $(u_0, i'_0) \stackrel{w_1}{\to} (u_1, i'_1)$. □

Proposition 8. *Let S be presented by a deterministic labeled graph G of finite memory and let $\mathcal{E}$ be an irreducible stethering (S, n)-encoder obtained from G and tagged in any way. Then $\mathcal{E}$ is lossless of finite order if and only if $\mathcal{E}$ is sliding-block decodable.*

Proof. As before, sliding-block decodability implies losslessness of finite order. If $\mathcal{E}$ is lossless of finite order, then, by Proposition 6, the downward extension of each vertical single-line must hit a vertical double-line. It follows that, with the notation in Proposition 7, we must have $(u_0, i_0) = (u'_0, i'_0)$ and $(u_1, i_1) = (u'_1, i'_1)$, and so all paths which generate the word $w_{-m+1}w_{-m+2}\dots w_{N+K}$ must occupy the same edge at time 1. Thus, $\mathcal{E}$ is sliding-block decodable with respect to any input tagging. □

9 Limitations

In this section we discuss some limitations of the stethering and pseudo-stethering techniques.

The following example shows that Propositions 6 and 8 need not hold for irreducible pseudo-stethering encoders; in particular, it is possible for an irreducible pseudo-stethering encoder to be lossless of finite order, but not sliding-block decodable with respect to the usual input tagging (although of course it will be sliding-block decodable with respect to the Gray-tagging).

Example 6. Let S be the constrained system which is presented by the labeled graph G of Figure 9. Here $V_G = \{\alpha, \beta, \gamma, \delta, \epsilon\}$ and all edges are labeled distinctly. Let $\mathcal{E}$ be a pseudo-stethering $(S, 2)$-encoder based on the $(A_G, 2)$-approximate eigenvector $[\ 3\ 2\ 1\ 1\ 1\ ]^\top$. The first two rows of the respective arrays $\mathcal{B}(u)$ are shown in Tables 23–27. It is straightforward to verify that $\mathcal{E}$ is irreducible.

It is evident from the figures that the vertical single-line $\mathcal{L}$ in row 1 of $\mathcal{B}(\alpha)$ that separates the sub-interval $\mathcal{C}_1(\alpha; a_1; 1)$ (labeled $(\alpha, 1)$) from the sub-interval $\mathcal{C}_1(\alpha; a_1; 2)$ (labeled $(\alpha, 2)$)

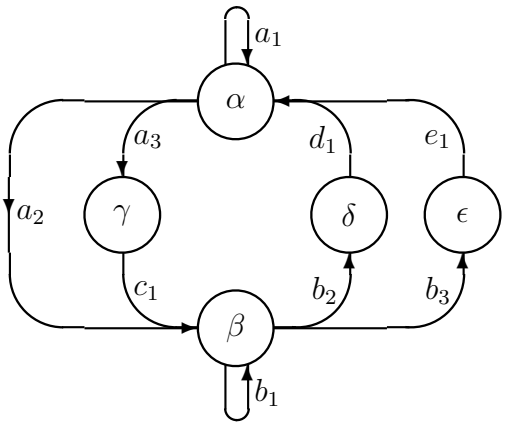

Figure 9: Graph G of Example 6.

	0		1		0		1		0		1
0	$(\alpha, 0)$				$(\alpha, 1)$				$(\alpha, 2)$		
1	$(\beta$,0)		$(\beta$,1)		$(\alpha$,0)		$(\alpha$,1)		$(\alpha$,2)		$(\gamma$,0)
	a_2		a_2		a_1		a_1	$\uparrow$ $\mathcal{L}$	a_1		a_3

Table 23: First two rows in $\mathcal{B}(\alpha)$ of Example 6.

	0		1		0		1
0	$(\beta, 0)$				$(\beta, 1)$		
1	$(\beta$,0)		$(\beta$,1)		$(\delta$,0)		$(\epsilon$,0)
	b_1		b_1		b_2		b_3

Table 24: First two rows in $\mathcal{B}(\beta)$ of Example 6.

	0		1
0	$(\gamma, 0)$		
1	$(\beta$,0)		$(\beta$,1)
	c_1		c_1

Table 25: First two rows in $\mathcal{B}(\gamma)$ of Example 6.

	0	1
0	$(\delta, 0)$	
1	$(\alpha,0)$	$(\alpha,2)$
	d_1	d_1

Table 26: First two rows in $\mathcal{B}(\delta)$ of Example 6.

	0	1
0	$(\epsilon, 0)$	
1	$(\alpha,0)$	$(\alpha,1)$
	e_1	e_1

Table 27: First two rows in $\mathcal{B}(\epsilon)$ of Example 6.

is the only vertical single-line in row 1 of any of the arrays whose downward extension does not meet any vertical double-lines or gaps. But the upward extension of $\mathcal{L}$ meets a vertical single-line in row 0 of $\mathcal{B}(\alpha)$. Thus, by Proposition 5, the encoder is lossless of finite order (and therefore, by Proposition 7, sliding-block decodable with respect to the Gray-tagging). However, the sequence $\cdots a_1 a_1 a_1 \cdots$ is generated in $\mathcal{E}$ by two distinct paths, namely, $\cdots \xrightarrow{a_1} (\alpha, 1) \xrightarrow{a_1} (\alpha, 1) \xrightarrow{a_1} \cdots$ and $\cdots \xrightarrow{a_1} (\alpha, 2) \xrightarrow{a_1} (\alpha, 2) \xrightarrow{a_1} \cdots$. With the usual input tagging, these paths are tagged $\cdots 111 \cdots$ and $\cdots 000 \cdots$. Thus, with respect to the usual input tagging, this encoder is not sliding-block decodable. •

In Example 7 below, we exhibit an irreducible constrained system S with $c(S) = \log n$ such that no stethering (S, n)-encoder can be lossless of finite order (and therefore no pseudo-stethering (S, n)-encoder can be lossless of finite order). The following results will help us establish this.

Proposition 9. *Let S be an irreducible constrained system with $c(S) = \log n$ and let $\mathcal{E}$ be a stethering (S, n)-encoder obtained from an irreducible deterministic presentation G of S. Then $\mathcal{E}$ is irreducible.*

Proof. Let (u, i) be a state in $\mathcal{E}$. The length of each down-scaled super-interval in row k of $\mathcal{B}(u)$ is at most $\|\mathbf{x}\|_\infty / n^k$. In particular, for $k = \lceil (1 + \log \|\mathbf{x}\|_\infty)/(\log n) \rceil$ this length is

at most $\frac{1}{2}$. Hence, the downward extension of the unit-interval $\mathcal{C}_0(u,i)$ onto row k of $\mathcal{B}(u)$ contains a whole down-scaled copy of row 0 of $\mathcal{B}(u')$ for some state $u' \in V_G$. Since G is irreducible, there is a path in G from u' to each state $v \in V_G$. It thus follows that there is a path in $\mathcal{E}$ from (u,i) to (v,j) for every $0 \leq j \leq x_v - 1$. Hence, $\mathcal{E}$ is irreducible. □

Corollary 3. *Let S be an irreducible constrained system with $c(S) = \log n$ and let $\mathcal{E}$ be a stethering (S,n)-encoder obtained from an irreducible deterministic presentation G of S. Then $\mathcal{E}$ is lossless of finite order if and only if for each vertical single-line $\mathcal{L}$ that appears in row 0, the downward extension of $\mathcal{L}$ must hit a vertical double-line.*

Proof. This is an immediate consequence of Propositions 6 and 9. □

When G is an irreducible graph with $\lambda(A_G) = n$, any (A_G, n)-approximate eigenvector is an exact eigenvector and is both positive and integral [23]. Thus, there is a unique 'smallest' (A_G, n)-approximate eigenvector which we refer to as the *smallest (A_G, n)-eigenvector*. All (A_G, n)-approximate eigenvectors are positive integral multiples of the smallest (A_G, n)-eigenvector.

Proposition 10. *Let S be presented by an irreducible deterministic labeled graph G with $c(S) = \log n$ and let $\mathbf{x}$ be the smallest (A_G, n)-eigenvector. Suppose that no stethering (S,n)-encoder based on $\mathbf{x}$ is lossless of finite order. Then no stethering (S,n)-encoder can be lossless of finite order.*

Proof. We prove this by contradiction. Suppose there is a stethering (S,n)-encoder $\mathcal{E}$ which is lossless of finite order. Such an encoder must be based on a positive integral multiple $k \cdot \mathbf{x}$ of $\mathbf{x}$. Consider the arrays $\mathcal{B}(u)$ in Table 20 (or Table 2) for $\mathcal{E}$. By Corollary 3, the downward extension of each vertical single-line in row 0 in every array $\mathcal{B}(u)$ must meet a vertical double-line. Now, delete all but every kth vertical line in each row. This defines a stethering (S,n)-encoder $\mathcal{E}'$ based on $\mathbf{x}$ instead of $k \cdot \mathbf{x}$. Observe that in this process we have not deleted any vertical double-lines. Thus, for $\mathcal{E}'$, we still have that the downward extension of each vertical single-line in row 0 must meet a vertical double-line. So, $\mathcal{E}'$ is lossless of finite order. □

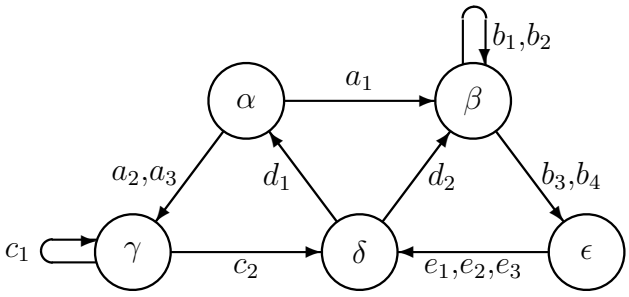

Figure 10: Graph G of Example 7.

Example 7. Let S be the constrained system presented by the irreducible labeled graph G of Figure 10. Again, the edges are all distinctly labeled.

One computes that the vector $\mathbf{x} = [\ 2\ 4\ 1\ 2\ 2\ ]^\top$ is a right eigenvector for A_G corresponding to eigenvalue 3. Thus, by the Perron-Frobenius Theorem [23], $c(S) = \log 3$ and $\mathbf{x}$ is the smallest $(A_G, 3)$-eigenvector. Now, by exhaustive search, one can check all of the stethering $(S, 3)$-encoders based on $\mathbf{x}$ (there are 72 of them) and see that none of these is lossless of finite order (an alternative argument, which requires less search, is sketched below). Therefore, by Proposition 10, there can be no stethering $(S, 3)$-encoder which is lossless of finite order. •

We now sketch the alternative argument. The state-splitting algorithm (see [1][22]) constructs encoders which are lossless of finite order (in fact, for finite memory constraints, it constructs encoders which are sliding-block decodable). This algorithm proceeds by splitting states of a deterministic presentation G in an iterative fashion until one arrives at a new presentation that is suitable for use as an encoder. The splittings are guided by choices based on an (A_G, n)-approximate eigenvector $\mathbf{x}$, and we then say that the splittings are *consistent* with respect to $\mathbf{x}$. As in stethering, the states of the encoders are descendants $(u, i),\ 0 \le i < x_u$ of states u in G. It is straightforward to see that, for Example 7, the first three rounds of any sequence of state splittings, consistent with respect to the smallest eigenvector $\mathbf{x} = [\ 2\ 4\ 1\ 2\ 2\ ]^\top$, are uniquely determined. Moreover, these splittings force the resulting encoder to have edges from each of the two states, $(\delta, 0), (\delta, 1)$ to exactly one descendant of α and two descendants of β. It is clear that no such encoder can be a stethering encoder. This, together with the following result, gives an alternative proof of the fact that

none of the stethering $(S, 3)$-encoders based on $\mathbf{x}$ can be lossless of finite order.

Proposition 11. *Let S be a constrained system presented by an irreducible deterministic labeled graph G with $c(S) = \log n$. Let $\mathcal{E}$ be a stethering (S, n)-encoder based on an eigenvector $\mathbf{x}$ of A_G. Then $\mathcal{E}$ is lossless of finite order if and only if $\mathcal{E}$ is obtained by a sequence of state splittings, starting from G, consistent with respect to $\mathbf{x}$.*

Proof. Consider the graph homomorphism from $\mathcal{E}$ to G which sends the state (u, i) to the state u and the edge $(u, i) \xrightarrow{a} (v, j)$ to the edge $u \xrightarrow{a} v$. This defines a mapping ψ from the set of semi-infinite paths in $\mathcal{E}$ to the set of semi-infinite paths in G. If $\mathcal{E}$ is lossless of finite order, then ψ satisfies the following two properties: (i) ψ is a *conjugacy* i.e., it is 1–1 and onto; this follows from Corollary 3; (ii) ψ is *left resolving* i.e., knowledge of (v, j), a, and u is enough to determine i: indeed, from (2) we have $i = \lfloor (\phi(u; a) + j)/n \rfloor$.

By [6][7], the existence of such a left-resolving conjugacy ψ from paths in $\mathcal{E}$ to paths in G implies that $\mathcal{E}$ is obtained from G by a sequence of state splittings consistent with respect to the vector $\mathbf{x}$ defined by

$$x_u = \text{ number states in } \mathcal{E} \text{ which map to } u.$$

Conversely, if $\mathcal{E}$ is obtained by a sequence of state splittings, starting from G, then $\mathcal{E}$ is lossless of finite order. □

It is not hard to modify this example to produce a constrained system $S = S(G)$ with $c(S) > \log 3$ and a *certain* (minimal) $(A_G, 3)$-approximate eigenvector $\mathbf{x}$ such that no stethering $(S, 3)$-encoder based on $\mathbf{x}$ can possibly be lossless of finite order. But we do not know of an example where $c(S) > \log 3$ and where we can prove that there is no stethering $(S, 3)$-encoder which is lossless of finite order.

Acknowledgment

The authors thank Moni Naor for helpful discussions.

References

[1] R.L. Adler, D. Coppersmith, M. Hassner, *Algorithms for sliding block codes — an application of symbolic dynamics to information theory, IEEE Trans. Inform. Theory,* IT-29 (1983), 5–22.

[2] R.L. Adler, J. Friedman, B. Kitchens, B.H. Marcus, *State splitting for variable-length graphs, IEEE Trans. Inform. Theory,* 32 (1986), 108–113.

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

[4] A.V. Aho, J.E. Hopcroft, J.D. Ullman, *The Design and Analysis of Computer Algorithms,* Addison-Wesley, Reading, Massachusetts, 1974.

[5] J.J. Ashley, *A linear bound for sliding block decoder window size, IEEE Trans. Inform. Theory,* IT-34 (1988), 389–399.

[6] J.J. Ashley, *LR conjugacies of shifts of finite type are uniquely so, Contemp. Math.,* 135 (1992), 57–84.

[7] J.J. Ashley, B.H. Marcus, *Canonical encoders for sliding block decoders,* submitted for publication.

[8] R. Fischer, *Sofic systems and graphs, Monats. fur Math.* 80 (1975), 179–186.

[9] R. Fischer, *Graphs and symbolic dynamics, Colloq. Math. Soc. János Bólyai, Topics in Information Theory,* 16 (1975), 229–243

[10] P.A. Franaszek, *On future-dependent block coding for input-restricted channels, IBM J. Res. Develop.,* 23 (1979), 75–81.

[11] P.A. Franaszek, *Synchronous bounded delay coding for input restricted channels, IBM J. Res. Develop.,* 24 (1980), 43–48.

[12] P.A. Franaszek, *A general method for channel coding, IBM J. Res. Develop.,* 24 (1980), 638–641.

[13] P.A. Franaszek, *Construction of bounded delay codes for discrete noiseless channels, IBM J. Res. Develop.,* 26 (1982), 506–514.

[14] C.D. Heegard, B.H. Marcus, P.H. Siegel, *Variable-length state splitting with applications to average runlength-constrained (ARC) codes, IEEE Trans. Inform. Theory,* 37 (1991), 759–777.

[15] K.A.S. Immink, *Coding Techniques for Digital Recorders,* Prentice Hall, New York, 1991.

[16] H. Kamabe, *Minimum scope for sliding block decoder mappings, IEEE Trans. Inform. Theory,* 35 (1989), 1335–1340.

[17] R. Karabed, B.H. Marcus, *Sliding-block coding for input-restricted channels, IEEE Trans. Inform. Theory,* 34 (1988), 2–26.

[18] N. Karmarkar, *A new polynomial-time algorithm for linear programming, Combinatorica,* 4 (1984), 373–395.

[19] Z. Kohavi, *Switching and Finite Automata Theory,* Second Edition, Tata McGraw-Hill, New Delhi, 1978.

[20] A. Lempel, M. Cohn, *Look-ahead coding for input-restricted channels, IEEE Trans. Inform. Theory,* 28 (1982), 933–937.

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

[22] B.H. Marcus, P.H. Siegel, J.K. Wolf, *Finite-state modulation codes for data storage, IEEE J. Sel. Areas Comm.,* 10 (1992), 5–37.

[23] H. Minc, *Nonnegative Matrices,* Wiley, New York, 1988.

[24] D.B. Shmoys, É. Tardos, *Computational complexity,* in *The Handbook of Combinatorics,* L. Lovász, R.L. Graham, M. Grötschel (Editors), North Holland, Amsterdam (to appear).

[25] P.H. Siegel, *Recording codes for digital magnetic storage, IEEE Trans. Magnetics,* 21 (1985), 1344–1349.

[26] R.S. Varga, *Matrix Iterative Analysis,* Prentice-Hall, Englewood Cliffs, New Jersey, 1962.