Entropy reduction, ordering in sequence spaces, and semigroups of non-negative matrices

Henk D.L. Hollmann * Peter Vanroose †

October 18, 1995

Abstract

We develop a mathematical framework to investigate classification or ordering of sequential input by means of finite-state algorithms, where the aim is to reduce the "diversity" at the output, that is, to achieve entropy reduction.

Our main interest is in optimal time-varying strategies; here, given a (finite) collection of algorithms sharing a common set of internal states, we consider ordering strategies represented by sequences of these algorithms, where the action on the t-th input is determined by the t-th algorithm in the sequence. So, in a sense we consider a programmable finite-state device and we are looking for the best program.

Surprisingly, there is a uniform method to handle questions of this type. Indeed, we first show how to transform such a problem to a problem on eigenvalues in a related semigroup of non-negative matrices, and then we present an approach to this eigenvalue problem which seems to succeed most of the time.

We apply our methods to a problem concerning ordering in sequence spaces introduced by Ahlswede, Ye, and Zhang (see the references) which motivated part of this work. In particular, we show that $\tau_2(0,2,1) = \frac{1}{3}{}^2\log(2+\sqrt{3})$, as conjectured by the second author some years ago.

Keywords: Entropy reduction, classification, ordering, sofic system, finite state machine, Perron-Frobenius, semigroups of non-negative matrices.

*Philips Research Laboratories, Prof. Holstlaan 4, 5656 AA Eindhoven, The Netherlands, hollmann@prl.philips.nl.

†Katholieke Universiteit Leuven, Kard. Mercierlaan 94, B–3001 Heverlee, Belgium, vanroose@esat.kuleuven.ac.be.

1 Introduction

In this paper, a *(time-invariant) finite-state entropy-reduction algorithm*, or briefly, an *algorithm*, is a synchronous finite-state input/output device; such a device takes inputs from a given source alphabet $\mathbf{B}$, and, depending on the input symbol and its internal state, produces an output symbol in an output alphabet $\mathbf{F}$ and moves to a new internal state.

In the context of channel codes (modulation codes), such a device is usually referred to as a (synchronous) finite-state encoder [6] and is used to translate or encode arbitrary sequences of source symbols into sequences that have certain desirable properties. In that context, it is of course required that decoding is possible.

Here, we think of such devices as performing some sort of data compression, and we will be interested in algorithms that have a "small" output space. A natural measure of the efficiency of an algorithm is thus the *entropy* of the output space, which measures the growth rate of the number N_n of output sequences of length n. (In the context of channel codes, the entropy is usually called the *(Shannon) capacity*.) We allow the case where the input sequences are restricted to a given constrained system over $\mathbf{B}$.

The use of the term "data compression" may cause confusion, since we do not consider the reconstruction problem. Instead, it might be better to speak of *ordering*, *classification*, or *entropy-reduction* of data sequences: an algorithm assigns an *index* to each data sequence (namely the corresponding output sequence) and the efficiency of the entropy-reduction is measured by the number of distinct indices produced by the algorithm.

As is well-known, the efficiency of such an algorithm, that is, the capacity of the constrained system formed by the collection of output sequences that can be produced by the algorithm, can be computed as the largest eigenvalue of a certain non-negative matrix.

Now let $\mathcal{F}$ be a finite collection of algorithms, all having the same source alphabet and sharing a common set of internal states. A *time-varying (entropy-reduction) algorithm over* $\mathcal{F}$ is a sequence

$$\mathbf{f} = f_1, f_2, f_3, \cdots$$

of algorithms $f_i \in \mathcal{F}$. We think of such a sequence $\mathbf{f}$ as an algorithm whose action at time t is directed by algorithm f_t. So at time t, $t = 1, 2, \ldots$, the algorithm $\mathbf{f}$ takes an input from the source alphabet and, depending on this input and its internal state, produces an output and moves to another internal state according to algorithm f_t. The collection of all entropy-reduction algorithms over $\mathcal{F}$ (all sequences over $\mathcal{F}$) will be denoted by $\mathcal{F}^\infty$.

It may happen that some time-varying algorithm in $\mathcal{F}^\infty$ performs better than the best algorithm in $\mathcal{F}$. (We will see examples of this later.) So the question now arises how to produce *lower bounds* for the efficiency of algorithms in $\mathcal{F}^\infty$ and how to find the best time-varying algorithm in $\mathcal{F}^\infty$. We will refer to this problem as the *optimal* *entropy-reduction problem* for $\mathcal{F}$.

The motivation to investigate time-varying entropy-reduction stems from a problem in [11] and [10] on ordering in sequence spaces, a subject introduced in [2] to study certain types of organization processes. In Section 2 we will outline these ordering problems, and we will show that they may be considered as special instances of the optimal entropy-reduction problem considered here.

Time-varying entropy-reduction algorithms and the optimal entropy-reduction problem are defined (in a slightly more general form) in Section 3.

In Section 4 we show how the optimal entropy-reduction problem can be transformed into a problem for a related finitely-generated semigroup of non-negative matrices. Briefly stated, we will show that with each algorithm f in $\mathcal{F}$ we may associate a non-negative matrix D_f such that the efficiency of an algorithm $\mathbf{f} = f_1, f_2, \ldots$ in $\mathcal{F}^\infty$ is measured by the number

$$\mu(\mathbf{f}) = \limsup_{t\to\infty} \lambda(D_{f_1} D_{f_2} \cdots D_{f_t})^{1/t},$$

where $\lambda(D)$ denotes the largest real (Perron-Frobenius) eigenvalue of a non-negative matrix D. The number

$$\mu(\mathcal{F}) = \liminf_{\mathbf{f}\in\mathcal{F}^\infty} \mu(\mathbf{f}),$$

which can be thought of as the minimum growth rate of matrix products in the semigroup generated by the matrices D_f, $f \in \mathcal{F}$, then provides the solution to the optimal entropy-reduction problem.

Section 5 is devoted to this semigroup problem. We present a method to obtain lower bounds for the optimal efficiency $\mu(\mathcal{F})$. (Note that, for each $\mathbf{f}$, $\mu(\mathbf{f})$ is an upperbound for $\mu(\mathcal{F})$, so finding upperbounds is easy.) In fact, we conjecture that under some mild conditions on the collection $\mathcal{F}$ our method will be able to determine the *exact* value of $\mu(\mathcal{F})$.

Later in this section, we return to the ordering problem. We will use our method to determine $\tau_2(0, 2, 1)$, the optimal efficiency of a time-varying binary ordering algorithm in the class $(0, 2, 1, T^+, O^-)$ [2]. Indeed, we show that $\tau_2(0, 2, 1) = \frac{1}{3}{}^2\log(2 + \sqrt{3})$, as conjectured in [10]. In addition, two other τ-values are determined, one of which is also new. Our approach suggests that (at least in principle) other values of $\tau_q(\pi, \beta, \phi)$ may be computed similarly.

Finally, we discuss our results in Section 6.

2 Ordering in sequence spaces

One of the motivations for this work stems from the following problem on ordering in sequence spaces [2]. An *ordering machine of type* $(\pi, \beta, \phi)_q$, $\pi \geq 0, \beta \geq 1, \phi \geq 1$, is a device to transform an arbitrary input sequence $\mathbf{x} = x_1, x_2, \cdots$ over a source alphabet

$\mathbf{B} = \{0, 1, \ldots, q-1\}$ of size q into an output sequence $\mathbf{y} = y_1, y_2, \cdots$ by *reordering* (See Figure 1). It consists of a *look-ahead shift register* of size $\phi - 1$, capable of holding $\phi - 1$ upcoming input symbols $x_{t+\beta+\phi-2}, \ldots, x_{t+\beta}$, a *memory box* of size β, capable of holding β *unordered* symbols from $\mathbf{B}$, and a *look-back shift register* of size π, which holds the last π previous output symbols $y_{t-1}, \ldots, y_{t-\pi}$. So the *internal state* of the device at time t consists of the (unordered) contents of the memory box and the (ordered) contents of the two shift registers, and can be represented by a tuple

$$s^{(t)} = (x_{t+\beta+\phi-2}, \ldots, x_{t+\beta}; b_0^{(t)}, \ldots, b_{q-1}^{(t)}; y_{t-1}, \ldots, y_{t-\pi}),$$

where $b_i^{(t)}$, $i \in \mathbf{B}$, represents the number of symbols i in the memory box. Note that the $b_i^{(t)}$ are nonnegative integers, and, for all t, $\sum_i b_i^{(t)} = \beta$.

The functioning of the ordering device can be described as follows. At time t, the ordering machine first reads the next upcoming input symbol $x_{t+\beta+\phi-1}$, then, depending on this symbol, its internal state $s^{(t)}$, and possibly the time t, chooses a symbol y_t from one of the symbols in the memory box. Next, the new symbol $x_{t+\beta+\phi-1}$ is shifted into the look-ahead shift register, and the output $x_{t+\beta}$ of the shift register is transfered into the memory box to replace y_t. The output y_t of the memory box is then shifted into the look-back shift register, and the output $y_{t-\pi}$ of the shift register is the next output of the entire device.

Let us denote the collection of all possible internal states of the device by S. We may describe the possible *actions* of our device with the aid of a labelled digraph. This digraph will have S as its set of states, and labelled transitions of the form

$$s^{(t)} \stackrel{x_{t+\beta+\phi-1}/y_{t-\pi}}{\longrightarrow} s^{(t+1)}.$$

In this terminology, a *time-invariant ordering algorithm* is a *subset* of these actions, with the property that for each state s and each input x this subset contains a *unique* action of the form $s \stackrel{x/y}{\longrightarrow} s'$. For later use, we observe that such an algorithm is fully described by a function

$$f : S \times \mathbf{B} \to \mathbf{F}$$

which, for a given internal state and input, stipulates which symbol is taken out of the memory box. (Note that this information is sufficient to determine both the next state and the output of the ordering machine.)

Example 2.1. For later use, we consider the simplest non-trivial case, where $(\pi, \beta, \phi)_q = (0, 2, 1)_2$. So we consider binary sequences, and hence the contents of the memory box, which consist of two binary symbols, are determined by the number $b = b_1$ of ones in the box. The two registers in Figure 1 have zero length. Consequently, the set of internal states is given by

$$S = \{0, 1, 2\}.$$

The labelled digraph that describes the possible actions of the ordering device is given in Figure 2.

It is easily verified that there are precisely four time-invariant algorithms, characterized by their action when in state 1 (i.e., when the memory box contains both a zero and a one). These algorithms will be referred to as f_{2i+j}, $i, j \in \{0,1\}$, where

$$f_{2i+j} : 1 \xrightarrow{0/i} 1-i; \qquad 1 \xrightarrow{1/j} 2-j.$$

Informally, these four algorithms can be described as follows. Algorithm f_0 (f_3) produces as output a zero (a one) when possible, and algorithm f_1 (f_2) produces an output equal to the input (the complement of the input) when possible.
It is not difficult to verify that each of the four time-invariant algorithms can produce (upon suitable inputs) any given output sequence. So no ordering is achieved by repeated application of a fixed algorithm. □

For a fixed ordering machine, of type $(\pi, \beta, \phi)_q$, say, we can make a (finite) list of all possible time-invariant algorithms and (at least in principle) compute the entropy (growth rate) of their output spaces, and hence the minimal entropy $\nu_q(\pi, \beta, \phi)$ of a time-invariant algorithm, just as was done in the above example.

Now think of the ordering machine as a *programmable* device, on which each of the (finitely many) algorithms can be implemented. Then, if the device could keep track of time (e.g., by counting the number of inputs that it has handled), it could vary the algorithm that it applies in time (but independent of the inputs). For example, if f and f' are two distinct algorithms, the ordering machine could be programmed to apply algorithm f at even times and algorithm f' at odd times. Surprisingly, such a strategy sometimes results in better "compression". The minimal entropy of the output space achievable by such time-varying strategies is denoted by $\tau_q(\pi, \beta, \phi)$.

Example 2.2. Consider again the case where $(\pi, \beta, \phi)_q = (0, 2, 1)_2$. In Example 2.1 we saw that none of the four algorithms f_k, $k = 0, \ldots, 3$, achieves any compression, that is, $\nu_2(0, 2, 1) = 1$.
Now suppose that the ordering machine applies the sequence of algorithms

$$f_0, f_1, f_0, f_1, \ldots.$$

(So time starts with $t = 1$, and the device applies algorithm f_0 at odd times and algorithm f_1 at even times.) It can be verified that no choice of input sequence and initial state results in an output sequence that contains 01110 as consecutive symbols. Therefore, at least *some* compression is achieved. □

In view of this example, it is natural to ask for the *optimal* strategy in the case $(\pi, \beta, \phi)_q = (0, 2, 1)_2$, and the optimal compression that can be achieved. (It is also not evident that this optimum can actually be attained.) These questions seem difficult; they ask for a statement on infinitely many strategies, and there seem to be no purely combinatorial techniques available to resolve them. To stress this last point: we invite the reader to come up with a *combinatorial* argument why no optimal strategy can involve algorithm f_2. (We could not find one.) This would be a logical first step in an alternative solution to these problems.

3 Time-varying entropy-reduction algorithms

In this section, we will develop a general model in which problems of the kind as discussed in the previous section can be formulated and analysed. Such a model should address the following points.

Input sequences (consisting of objects or data) are transformed into output sequences, with as aim to reduce the diversity of the sequences. The diversity of the output space is measured by its *(topological) entropy*, in the symbolic-dynamics sense.

Basically, possible transformations are described by (time-invariant) finite-state algorithms. The states represent the (limited) knowledge about the system (i.e., some knowledge about the past and about upcoming inputs). Possible actions of these algorithms are limited, and only finitely many algorithms are available. Our interest is in time-varying algorithms: in that case, we may apply different algorithms at different times, but the order in which the algorithms are applied is *fixed*, that is, does not depend on the input sequence.

Our model starts from a finite collection S (to be considered as internal states), a source (or input) alphabet $\mathbf{B}$, an output alphabet $\mathbf{F}$, and a finite collection E of *actions*, that is, labelled transitions on S, each of the form

$$s \xrightarrow{x/y} s'$$

for some states $s, s' \in S$, source symbol $x \in \mathbf{B}$ and output symbol $y \in \mathbf{F}$. In our model, the set E represents possible actions of algorithms.

A *time-invariant entropy-reduction algorithm* or briefly, an *algorithm*, is just a subset of E. An algorithm $f \subseteq E$ is said to be *deterministic* (or *complete*) if for each pair (s, x) with $s \in S$ and $x \in \mathbf{B}$ there is *at most one* (or *at least one*) transition in f leaving s with source label x. In most practical cases, algorithms will be both complete and deterministic, that is, an algorithm f will represent a *deterministic* input/output device: for each internal state s and each input x, the algorithm f produces a unique output y while moving to a unique next state s'. However, in what follows we do not make any special assumptions about the algorithms.

Now let $\mathcal{F}$ be a finite collection of such algorithms. Write $\mathcal{F}^\infty$ to denote the collection of infinite sequences

$$\mathbf{f} = f_1, f_2, f_3, \cdots$$

with $f_t \in \mathcal{F}$ for all t. We will refer to members of $\mathcal{F}^\infty$ as *time-varying (entropy-reduction) algorithms*, or *strategies*, over $\mathcal{F}$.

In our interpretation, the action of strategy $\mathbf{f}$ at time t is determined by algorithm f_t. That is, a strategy $\mathbf{f} \in \mathcal{F}^\infty$ can produce an output sequence $\mathbf{y} = y_1, y_2, \ldots$ if there exists an input sequence $\mathbf{x} = x_1, x_2, \ldots$, a sequence of states $s_0, s_1, \ldots$, and a *walk* (sequence of successive actions) $\epsilon_1, \epsilon_2, \ldots$ in E, with each ϵ_t of the form $\epsilon_t : s_{t-1} \xrightarrow{x_t/y_t} s_t$, such that ϵ_t is contained in f_t, for all $t \geq 1$.

The *output space* $\mathcal{L}(\mathbf{f})$ of a strategy $\mathbf{f} \in \mathcal{F}^\infty$ is the collection of all output sequences that can thus be produced by $\mathbf{f}$. The *efficiency* or *capacity* $\tau(\mathbf{f})$ of $\mathbf{f}$ is defined as

$$\tau(\mathbf{f}) = \limsup_{t\to\infty} \frac{1}{t} \log |\mathcal{L}_t(\mathbf{f})|, \tag{1}$$

where $\mathcal{L}_t(\mathbf{f})$ denotes the collection of all words of length t in $\mathcal{L}(\mathbf{f})$ and log denotes the q-logarithm, where $q = |\mathbf{F}|$ is the size of the output alphabet. (So $\tau(\mathbf{f})$ measures the maximum *growth rate* of sequences of length t in $\mathcal{L}(\mathbf{f})$. This notion is similar to the notion of (topological) entropy in symbolic dynamics, see e.g. [1].)

The *optimal efficiency* $\tau(\mathcal{F})$ of a strategy over $\mathcal{F}$ is defined as

$$\tau(\mathcal{F}) = \inf_{\mathbf{f}\in\mathcal{F}^\infty} \tau(\mathbf{f}).$$

The problem to determine $\tau(\mathcal{F})$, which is the central problem in this paper, will be referred to as the *optimal entropy-reduction problem* for $\mathcal{F}$.

Let $\mathcal{F}^+$ denote the collection of all finite-length sequences

$$\mathbf{f} = f_1, f_2, \ldots, f_n,$$

$n \geq 1$, with $f_t \in \mathcal{F}$, $1 \leq t \leq n$. We think of such a sequence as representing the *periodic* strategy $\mathbf{f}$ with $f_t = f_{t \pmod n}$ for all t. By abuse of notation, we write $\mathcal{L}(f_1, \ldots, f_n)$ and $\tau(f_1, \ldots, f_n)$ for the output space and the efficiency, respectively, of the periodic algorithm represented by $f_1, \ldots, f_n$. Observe that the periodic strategy $f, f, \ldots$ acts as the time-invariant algorithm $f \in \mathcal{F}$. The optimal efficiency $\nu(\mathcal{F})$ of a time-invariant algorithm in $\mathcal{F}$ is defined as

$$\nu(\mathcal{F}) = \min_{f\in\mathcal{F}} \tau(f).$$

We will show later that, in order to compute $\tau(\mathcal{F})$, we only need to consider *periodic* strategies. Moreover, we will show that, in the case where $\mathbf{f}$ is periodic, the "lim sup" in (1) can be replaced by "lim".

As was noted before, it may happen that $\tau(\mathcal{F}) < \nu(\mathcal{F})$, i.e. varying the algorithm to be used, independent of the input data, may result in better compression.

We end this section with a few comments on the model introduced here. (The reader may skip the remainder of this section on first reading.)

First we remark that our model is sufficiently general to incorporate the case where input sequences are restricted to a given constrained system $\mathcal{S}$ with alphabet $\mathbf{B}$ (see e.g. [1]). Indeed, suppose that $\mathcal{S}$ is generated by a labelled directed graph (or finite-state transition diagram) with set of states Σ and a collection A of labelled transitions, each of the form

$$\sigma \xrightarrow{x} \tau,$$

with $\sigma, \tau \in \Sigma$, $x \in \mathbf{B}$.

Define a new collection E' of labelled transitions on the set of states $\Sigma \times S$, where E' contains a transition

$$(\sigma, s) \xrightarrow{x/y} (\tau, s')$$

for each pair of a transition

$$\sigma \xrightarrow{x} \tau$$

in A and a transition

$$s \xrightarrow{x/y} s'$$

in E. Now the algorithm f, when acting on input sequences from $\mathcal{S}$ only, is described by the collection of transitions $(\sigma, s) \xrightarrow{x/y} (\tau, s')$ with $\sigma \xrightarrow{x} \tau$ in A and $s \xrightarrow{x/y} t$ in f.

Our second remark is that in fact inputs are not essential in our model, and may be dispensed with altogether. The idea is that each algorithm is in fact characterized by the labelled directed graph that generates its output sequences. (This graph is obtained from the actions of the algorithm by dropping the input-part of the labels.)

From this point of view, a time-varying algorithm (strategy) is a *sequence* $G_1, G_2, \ldots$ of labelled directed graphs, where each G_t corresponds to one of the algorithms. Let the labelled directed graph G be the *union* (superposition) of all the graphs G_t. Then the output space generated by this sequence of graphs is the collection of all sequences generated by walks $\{\epsilon_t\}$ in G for which ϵ_t is contained in G_t for all t. Note that this output space can also be considered as a *code generated by a time-varying trellis*.

The above may thus be considered as a *mixing operation* for constrained channels, and we ask which mixing gives the lowest capacity. (We remark that the question which mixing gives the *highest* capacity can be handled by techniques similar to those developed in this paper.) Further details concerning this more general model will be given elsewhere. (See also [5] and the references given there.) It would also be interesting to consider the corresponding mixing operation for *Markov sources*, and now ask which mixing gives the lowest entropy in the information-theoretical sense.

4 A semigroup related to $\mathcal{F}$

We now turn to the problem how to determine $\tau(\mathcal{F})$. In this section, we show how this problem can be transformed into a problem on eigenvalues of non-negative matrices in a semigroup. The latter problem will be discussed in the next section.

First of all, we need a more transparent description of how a given strategy $\mathbf{f}$ generates its output sequences. The problem is of course that a fixed output sequence $\mathbf{y} \in \mathcal{L}(\mathbf{f})$ may be generated by many different pairs $s, \mathbf{x}$ of starting state s and input sequence $\mathbf{x}$. This difficulty can be handled as follows.

Let us investigate in how many ways a given sequence $y_1, \ldots, y_t \in \mathcal{L}_t(\mathbf{f})$ can be extended to a sequence $y_1, \ldots, y_{t+1} \in \mathcal{L}_{t+1}(\mathbf{f})$. So let S_t denote the collection of all states in which the algorithm $\mathbf{f}$ can be at time t after generating output $y_1, \ldots, y_t$. We will refer to non-empty subsets of S of this form as *superstates*. Note that, in particular, $S_0 = S$ is a superstate. Now, $\mathbf{f}$ can generate $y_1, \ldots, y_{t+1}$ at time $t+1$ precisely when algorithm f_{t+1} can produce output y_{t+1} from some state in S_t, under a suitable input. This observation motivates the following approach.

For each non-empty subset S' of states in S, each output symbol y, and each algorithm f in $\mathcal{F}$, we define a labelled transition

$$\alpha : S' \xrightarrow{f/y} S'',$$

where S'' is the set of all states s'' in S for which algorithm f contains a transition from a state s' in S' to s'' that produces output y (that is, with label x/y, for some input x).

We could represent these transitions in the form of a directed graph. Then it makes sense to speak of a *walk* (directed path) in this graph. Now let $\mathcal{A}$ denote the collection of all such transitions α with S'' non-empty that can be reached by a walk starting in superstate S. Moreover, let $\mathcal{V}$ denote the collection of all subsets S' of S that are involved in transitions α in $\mathcal{A}$. So $\mathcal{V}$ is precisely the collection of all superstates. Accordingly, we might refer to the transitions in $\mathcal{A}$ as *superactions*. Note that $\mathcal{V}$ and $\mathcal{A}$ are the vertices and arcs of a *subgraph* of the directed graph referred to above. The reason behind these definitions is given by the following theorem.

Theorem 4.1. *Let $\mathbf{f}$ in $\mathcal{F}^\infty$ be fixed. Then there is a one-to-one correspondence between*
(i) walks in $\mathcal{A}$ of length t that start in superstate S, with label sequence $f_1/y_1, \ldots, f_t/y_t$, and
(ii) output sequences $y_1, \ldots, y_t$ of length t that can be generated by $\mathbf{f}$.

Proof. First we observe that $\mathcal{A}$ is *deterministic*, that is, there is at most one transition in $\mathcal{A}$ from any given superstate with any given label. Consequently, given $\mathbf{f}$ and $\mathbf{y}$, there is at most one walk in $\mathcal{A}$ starting in S with label sequence $f_1/y_1, f_2/y_2, \ldots$.

On the other hand, it is easily seen that for each $\mathbf{f}$ in $\mathcal{F}^\infty$ and each output sequence $y_1, \ldots, y_t$ that can be generated by $\mathbf{f}$, there exists a walk in $\mathcal{A}$ with the required properties. □

We will use the above result to count the number of output sequences of a given length that can be generated by a given strategy. To that end, we will associate with each algorithm $f \in \mathcal{F}$ a $|\mathcal{V}| \times |\mathcal{V}|$-matrix D_f, where, for each pair $S', S'' \in \mathcal{V}$, we take $D_f(S', S'')$ to be the number of transitions in $\mathcal{A}$ from superstate S' to superstate S'' carrying label f/y for some y. Observe that D_f is a non-negative matrix, a property that will play an important role later on. Moreover, it is easily seen that, for a given sequence $f_1, f_2, \ldots, f_t$ of algorithms, the (S', S'')-entry of the matrix product

$$D_{f_1} D_{f_2} \cdots D_{f_t}$$

counts the number of distinct walks in $\mathcal{A}$ from S' to S'' with label sequence of the form $f_1/y_1, f_2/y_2, \ldots, f_t/y_t$. So in view of Theorem 4.1, we have the following.

Theorem 4.2.
$$|\mathcal{L}_t(\mathbf{f})| = \sum_{S''\in\mathcal{V}} D_{f_1}D_{f_2}\cdots D_{f_t}(S,S'').$$

For a non-negative square matrix D, we will denote by $\lambda(D)$ the largest real (Perron-Frobenius) eigenvalue of D. We now have the following result.

Theorem 4.3. *(i) For each periodic strategy $f_1, f_2, \ldots, f_m$ in $\mathcal{F}^+$, we have that*
$$\tau(f_1,\ldots,f_m) = \log\Big(\lambda(D_{f_1}D_{f_2}\cdots D_{f_m})^{1/m}\Big) = \frac{1}{m}\log\lambda(D_{f_1}D_{f_2}\cdots D_{f_m}).$$
(ii) The optimal efficiency $\tau(\mathcal{F})$ of a strategy over $\mathcal{F}$ is given by
$$\tau(\mathcal{F}) = \inf\log\Big(\lambda(D_{f_1}D_{f_2}\cdots D_{f_m})^{1/m}\Big),$$
where the infimum is taken over all periodic strategies $f_1, f_2, \ldots, f_m$, $m \geq 1$, in $\mathcal{F}^+$.

Proof. We willl only sketch the proof, leaving some details to the reader.
(i) Write $D = D_{f_1}D_{f_2}\cdots D_{f_m}$. Using well-known properties of non-negative matrices (see e.g. [9]), we may conclude that
$$\Big(\sum_{S''\in\mathcal{V}} D^k(S,S'')\Big)^{1/k} \to \lambda(D), \tag{2}$$
for $k\to\infty$. (Here we use the fact that each $S''\in\mathcal{V}$ can be reached by a walk from S.) Obviously, the same statement holds if, in (2), we replace D^k by $D^kD_{f_1}\cdots D_{f_i}$. Hence, from Theorem 4.2 we conclude that
$$|\mathcal{L}_t(f_1,f_2,\ldots,f_m)|^{1/t} \to \lambda(D)^{1/m},$$
for $t\to\infty$, from which part (i) follows.

(ii) We begin with a few observations. Let D be any non-negative matrix, and let 1 denote the all-one vector. Then we have that
$$1^\top D1 \geq \lambda(D). \tag{3}$$
Indeed, by well-known properties of non-negative matrices, there is a non-negative vector v for which $Dv = \lambda(D)v$ and $1^\top v = 1$. Then $v \leq 1$ (componentwise inequality), hence
$$1^\top D1 \geq 1^\top Dv = \lambda(D)1^\top v = \lambda(D).$$
Let $\mathbf{f}\in\mathcal{F}^\infty$, and put $D^{(m)} = D_{f_1}\cdots D_{f_m}$, $m\geq 1$. Let e_S denote the vector with S-entry equal to 1 and all other entries equal to 0. Since, by definition, each superstate S'' can be reached from S, we have that
$$(e_S^\top D^{(m)}1)^{1/m} \approx (1^\top D^{(m)}1)^{1/m}. \tag{4}$$
Now by Theorem 4.2 and (4), we find that
$$|\mathcal{L}_m(\mathbf{f})|^{1/m} = (e_S^\top D^{(m)}1)^{1/m} \approx (1^\top D^{(m)}1)^{1/m},$$
so, in view of the results of part (i), part (ii) follows from (3). □

Example 4.4. Consider again the ordering machine of type $(\pi, \beta, \phi)_q = (0, 2, 1)_2$ investigated in Example 2.1. The following facts can be verified. The set of "superstates" $\mathcal{V}$ reachable from "superstate" $S = \{0, 1, 2\}$ has six elements, namely $\{0\}, \{1\}, \{2\}, \{0, 1\}, \{1, 2\}$, and $\{0, 1, 2\}$. With respect to this ordering of the states, the matrices D_0, D_1, D_2, and D_3 corresponding to algorithms $f_{0,0}, f_{0,1}, f_{1,0}$, and $f_{1,1}$ are found to be

$$D_0 = \begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \end{pmatrix}, \qquad D_1 = \begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 \end{pmatrix},$$

$$D_2 = \begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 2 \end{pmatrix}, \qquad D_3 = \begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 \end{pmatrix}.$$

For later use we observe that if we interchange rows and columns indexed by $\{0\}$ and $\{2\}$, and also those indexed by $\{0, 1\}$ and $\{1, 2\}$, we have that $D_0 \leftrightarrow D_3$, $D_1 \leftrightarrow D_1$, and $D_2 \leftrightarrow D_2$. This symmetry can be understood as resulting from interchanging the two source symbols 0 and 1.
It is easily verified that $\lambda(D_i) = 2$, $i = 1, \ldots, 4$. Hence the optimal efficiency $\nu_2(0, 2, 1)$ of a time-invariant algorithm equals 1, that is, in the time-invariant case no entropy reduction can be achieved. (In fact, each of the four algorithms can produce any given binary output sequence.) On the other hand, in [10] it was shown that $\lambda(D_0 D_1 D_3) = 2 + \sqrt{3}$, hence the optimal efficiency $\tau_2(0, 2, 1)$ of a strategy satisfies $\tau_2(0, 2, 1) \leq \frac{1}{3} \log(2 + \sqrt{3}) \approx 0.633$.
Later on, we will indicate how this can be verified and we will show that in fact equality holds. □

5 The semigroup problem

Motivated by the results of the previous section, we investigate the following problem, which is also interesting in its own right.

Let $\mathcal{D} = \{D_1, \ldots, D_n\}$ be a collection of non-negative $d \times d$ matrices over the real numbers. For each word $\mathbf{w} = w_1 \cdots w_k$ of length $|\mathbf{w}| = k \geq 1$ over $\{1, \ldots, n\}$, we let

$$D(\mathbf{w}) = D_{w_1} \cdots D_{w_k}$$

be the matrix product of $D_{w_1}, \ldots, D_{w_k}$. We write $\mathcal{D}^+$ to denote the semigroup generated by $\mathcal{D}$, that is, the collection of all these matrices $D(\mathbf{w})$. Since $\mathcal{D}^+$ consists entirely of non-negative matrices, the following definitions make sense.

Let the *normalized eigenvalue* $\mu(\mathbf{w})$ of a matrix $D(\mathbf{w})$ be defined as $\mu(\mathbf{w}) = \mu(D(\mathbf{w})) = \lambda(D(\mathbf{w}))^{1/|\mathbf{w}|}$, the $1/|\mathbf{w}|$-th power of the largest real eigenvector of $D(\mathbf{w})$, and put

$$\mu(\mathcal{D}) = \inf_{\mathbf{w}} \mu(\mathbf{w}).$$

We may think of $\mu(\mathcal{D})$ as the *minimum growth rate* of the entries of matrices in $\mathcal{D}^+$. We will sometimes refer to the quantity $\tau(\mathcal{D}) = \log \mu(\mathcal{D})$ as the *entropy* of $\mathcal{D}$. Note that if $\mathcal{F}$ is a finite collection of algorithms and if $\mathcal{D}$ consists of the collection of matrices D_f associated with algorithms $f \in \mathcal{F}$, then $\tau(\mathcal{F}) = \tau(\mathcal{D}) = \log \mu(\mathcal{D})$.

The problem that we investigate is how to obtain $\mu(\mathcal{D})$. In particular, we require a method to compute lower bounds for this number. Our approach to this problem is the following. Let $u_1, \ldots, u_m$, $m \geq 1$, be a collection of non-zero, nonnegative d-dimensional real vectors, and let $\mu \geq 0$. Suppose that, for each $i = 1, \ldots, n$ and each $j = 1, \ldots, m$, the vector $\mu^{-1} D_i u_j$ dominates some convex combination of $u_1, \ldots, u_m$, that is, there are numbers $\lambda_{ijk} \geq 0$ with $\sum_{k=1}^m \lambda_{ijk} = 1$ such that

$$\mu^{-1} D_i u_j \geq \sum_{k=1}^{m} \lambda_{ijk} u_k,$$

where the inequality signifies componentwise inequality. Then, as we will show, we have $\mu(\mathcal{D}) \geq \mu$. Before we prove this, we first give some reformulations of the above assumptions.

Put $U = [u_1 \cdots u_m]$ (so U is a $d \times m$ matrix with as columns the vectors u_k), and let the $m \times m$ matrices L_i, $i = 1, \ldots, n$, be defined by

$$L_i(k, j) = \lambda_{ijk}.$$

Then our assumptions translate as follows. There are matrices U and L_i, $i = 1, \ldots, n$, with

$$U = [u_1 \cdots u_m] \geq 0, u_j \neq 0, L_i \geq 0, 1^\top L_i = 1^\top, \tag{5}$$

$j = 1, \ldots, m$, $i = 1, \ldots, n$, such that

$$\mu^{-1} D_i U \geq U L_i \tag{6}$$

holds for $i = 1, \ldots, n$. (The above inequality now signifies inequality for each entry. The vector 1 denotes the all-one vector of length m.)

We can also translate the above into more geometrical terms. For a collection U of non-negative vectors $u_1, \ldots, u_m$, let

$$U^\uparrow = \{x \mid x \geq \sum_{k=1}^{m} \lambda_k u_k \text{ holds for some } \lambda \geq 0 \text{ with } 1^\top \lambda = 1\}.$$

So $U^\uparrow$ consists of all vectors that dominate some convex combination of the vectors u_k. It is a convex set, and $\mathrm{ext}(U^\uparrow)$, the collection of extremal vectors in $U^\uparrow$, is contained in U. In fact, it is easy to see that we only need to consider sets U for which

$\mathrm{ext}(U^{\uparrow}) = U$, that is, we may assume w.l.o.g. that no vector $u \in U$ dominates a convex combination of vectors in $U \setminus \{u\}$. (To see this, it is sufficient to remark that for all $D \geq 0$ we have $Du \leq Dv$ whenever $u \leq v$.) For any collection of vectors X, define

$$\mathcal{D}(X) = \{D_i x \mid i = 1, \ldots, n; x \in X\}.$$

Then (5) and (6) are equivalent to the existence of a collection U, consisting entirely of non-zero nonnegative vectors, for which

$$\mathcal{D}(U) \subseteq \mu U^{\uparrow}. \tag{7}$$

Remark that $\mathcal{D}(U) \subseteq \mu U^{\uparrow}$ holds if and only if $\mathcal{D}(U)^{\uparrow} \subseteq \mu U^{\uparrow}$, an observation that will be used without comment later on.

In what follows, we will require the following result. (It is surely well-known, but we could not find a reference.)

Proposition 5.1. *If $D \geq 0$ and $Du \geq \lambda u$ holds for some $\lambda \geq 0$ and some vector $u \geq 0, \neq 0$, then $\lambda(D) \geq \lambda$.*

Proof. The result is trivial if D is irreducible. Indeed, in that case there is a vector $v > 0$ with $v^{\top} D = \lambda(D) v^{\top}$, and our assumption implies

$$\lambda(D) v^{\top} u = v^{\top} D u \geq \lambda v^{\top} u.$$

Since $v^{\top} u > 0$, this in turn implies $\lambda(D) \geq \lambda$.

For general $D \geq 0$, standard theory (see i.e. [9], [7]) shows that, after a suitable numbering of the rows and columns, D has a block right-triangular structure with all blocks on the diagonal irreducible (with the possible exception of the last block, which may be 0). Now the result can be derived from the irreducible case. □

Now we come to one of the main results of this paper.

Theorem 5.2. *Let U be a finite collection of non-zero, nonnegative vectors, and let $\mu \geq 0$.*
(i) If $\mathcal{D}(U) \subseteq \mu U^{\uparrow}$, then $\mu(\mathcal{D}) \geq \mu$.
(ii) If $\mathcal{D}(U)^{\uparrow} = \mu U^{\uparrow}$, then $D(\mathbf{w})u = \mu^{|\mathbf{w}|} u$ holds for some vector $u \in U$ and some matrix $D(\mathbf{w}) \in \mathcal{D}^{+}$. If also $\lambda(D(\mathbf{w})) = \mu^{|\mathbf{w}|}$ (which is certainly true if $u > 0$), then $\mu(\mathcal{D}) = \mu$.

Proof. We will use the assumptions in (i) in the form given by (5) and (6).
(i) For each word $\mathbf{w}$, let $L(\mathbf{w}) = L_{w_1} \cdots L_{w_k}$, where $k = |\mathbf{w}|$. From (6) we find that

$$D(\mathbf{w}) U \geq \mu^{|\mathbf{w}|} U L(\mathbf{w}). \tag{8}$$

Since $1^{\top} L(\mathbf{w}) = 1^{\top}$, the largest real eigenvalue of $L(\mathbf{w})$ equals 1, hence there is a vector $v \geq 0$, $v \neq 0$, such that $L(\mathbf{w}) v = v$. Then we conclude from (8) that

$$D(\mathbf{w}) U v \geq \mu^{|\mathbf{w}|} U v,$$

and since the vector Uv is non-zero and nonnegative, by Proposition 5.1 this inequality implies that $\lambda(D(\mathbf{w})) \geq \mu^{|\mathbf{w}|}$, whence $\mu(\mathbf{w}) \geq \mu$. Since this holds for each word $\mathbf{w}$, the first part of the theorem follows.
(ii) Without loss of generality, we may assume that $\mathrm{ext}(U^{\uparrow}) = U$. We conclude that if equality holds in (7), then each vector μu, $u \in U$, is the image Dv of some $v \in U$ under some $D \in \mathcal{D}$. But U is finite, so this implies that there are $u \in U$ and matrices $D_{w_1}, \ldots, D_{w_k}$, $k \geq 1$, in $\mathcal{D}$ such that

$$D_{w_1} \cdots D_{w_k} u = \mu^k u,$$

that is, u is an eigenvector of $D_{w_1} \cdots D_{w_k}$, with (real) eigenvalue μ^k. Now (ii) is evident. □

A set $U^{\uparrow}$ generated by a set of vectors U as in Theorem 5.2 for which

$$\mathcal{D}(U) \subseteq \mu(\mathcal{D}) U^{\uparrow} \tag{9}$$

holds will be termed an *eigenregion* for $\mathcal{D}$. If moreover $D(\mathbf{w})u = \mu(\mathcal{D})^{|\mathbf{w}|}u$ holds for some vector $u \in U$ and some matrix $D(\mathbf{w}) \in \mathcal{D}^+$, then we will say that $\mathcal{D}$ is *well-behaved*. This last definition may be superfluous: we know of no example of a collection $\mathcal{D}$ that has an eigenregion but is not well-behaved.

The above theorem may be considered as a generalisation on various aspects of the Perron-Frobenius theory for non-negative matrices. Indeed, suppose that $\mathcal{D}$ consists of a single matrix D. Then obviously $\mu(\mathcal{D}) = \lambda(D)$, the largest real eigenvalue of D. If U consists of a single vector u, then part (i) of the theorem states that $Du \geq \mu u, u \geq 0, u \neq 0$, implies $\lambda(D) \geq \mu$, which is a well-known result (see Proposition 5.1 above). Moreover, the equality in (ii) holds if and only if u is an eigenvector of D with corresponding eigenvalue μ. If D is also *irreducible* (that is, for each i, j there exists $N > 0$ such that $D^N_{i,j} > 0$), then D has a *unique* non-negative eigenvector (to constant multiples). So in that case, $\mathcal{D}$ has a *unique* (to a constant multiple) eigenregion.

Unfortunately, such a uniqueness result is not to be expected in the case where $\mathcal{D}$ contains more than one matrix, as the following example shows.

Example 5.3. Let $\mathcal{D} = \{D_0, D_1\}$, where

$$D_0 = \begin{pmatrix} 4 & 1 \\ 10 & 1 \end{pmatrix}, \qquad D_1 = \begin{pmatrix} 1 & 10 \\ 1 & 4 \end{pmatrix}.$$

Put $u_0 = [1,2]^{\top}$, $u_1 = [2,1]^{\top}$, and let $U = U(\theta) = \{u_0, \theta u_1\}$. Observe that u_i is the unique non-negative eigenvector of D_i, $i = 0, 1$, with eigenvalue 6.
It is an easy exercise to verify that $\mathcal{D}(U)^{\uparrow} = 6U^{\uparrow}$ holds precisely when $2/3 \leq \theta \leq 3/2$. So we conclude that $\mu(\mathcal{D}) = 6$, but the set U is not unique.
Moreover, U need not even be "projectively unique". Indeed, if $u_2 = \frac{1}{6}[21, 9]^{\top} = \frac{1}{6} D_1 u_0$ and $U = \{u_0, u_2\}$, then again we have that $\mathcal{D}(U)^{\uparrow} = 6U^{\uparrow}$. □

The usefulness of Theorem 5.2 is demonstrated by the following examples.

Example 5.4. Let $\mathcal{D} = \{D_0, D_1, D_2, D_3\}$, where the D_i are as given in Example 4.4. We now set out to determine $\tau_2(0,2,1) = {}^2\log \mu(\mathcal{D})$. An easy computation shows that

$$D_3D_1D_0 = \begin{pmatrix} 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 2 & 2 \\ 0 & 0 & 0 & 0 & 2 & 1 \\ 0 & 0 & 0 & 0 & 3 & 2 \end{pmatrix}, \qquad D_0D_1D_3 = \begin{pmatrix} 0 & 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 2 & 0 & 1 \\ 0 & 0 & 0 & 2 & 0 & 2 \\ 0 & 0 & 0 & 3 & 0 & 2 \end{pmatrix}.$$

Let $\lambda = 2 + \sqrt{3}$ (so λ satisfies $\lambda^2 = 4\lambda - 1$), and define two vectors u_0 and u_1 by

$$u_0 = [\lambda - 3, \lambda - 3, 12 - 3\lambda, 2\lambda - 6, 1, \lambda - 2]^\top,$$

$$u_1 = [12 - 3\lambda, \lambda - 3, \lambda - 3, 1, 2\lambda - 6, \lambda - 2]^\top.$$

Then u_0 and u_1 are the (unique) positive eigenvectors of $D_3D_1D_0$ and $D_0D_1D_3$, respectively. (Note that the sign of an expression $a + b\lambda$ can be obtained from the signs of b, $a + 2b$, and $3b^2 - (a + 2b)^2$.)
We now claim that $\mu(\mathcal{D}) = \mu = \lambda^{1/3} = (2 + \sqrt{3})^{1/3}$. This can be verified as follows. For a word $\mathbf{w} \in \{0,1,2,3\}^*$, let

$$u_i(\mathbf{w}) = \mu^{-|\mathbf{w}|} D(\mathbf{w}) u_i.$$

Define the sets of words W_0 and W_1 as

$$W_0 = \{\wedge, 0, 1, 10, 010, 110, 0110, 3110, 03110, 13110\},$$

$$W_1 = \{\wedge, 3, 1, 13, 313, 113, 3113, 0113, 30113, 10113\},$$

and put

$$U = \{u_i(\mathbf{w}) \mid i = 0, 1; \mathbf{w} \in W_i\}.$$

(Here $\wedge$ denotes the empty word, and $u_i(\wedge) = u_i$.) So U contains twenty vectors. By abuse of notation, we also use U to denote the 6×20-matrix with as columns the vectors $u \in U$. For this set U, it can be shown that $\mathcal{D}(U)^\uparrow = \mu U^\uparrow$. In order to verify this, we have to show that for each vector u in U and each i, $i = 0, \ldots, 3$,

$$\max\{1^\top x \mid x \geq 0; \mu U x \leq D_i u\} \geq 1. \tag{10}$$

Checking (10) is easy with the aid of a computer. (In fact, in most cases we have that $D_i u \geq u'$, for some $u' \in U$, and in most cases, the maximum of the above LP-problem is greater than one.) We also verified (10) *exactly*, by hand, using only arithmetic in $\mathbf{Q}(\lambda)$. (Remark that the work can be reduced somewhat by using the symmetry induced by interchanging the two source symbols 0 and 1, as discussed in Example 4.4. Note that under this symmetry, we have that $u_0 \leftrightarrow u_1$ and $A_0 \leftrightarrow A_1$, so the symmetry leaves U invariant.)
We do not claim that this set U is the smallest set of vectors that works; however, we have not found a smaller one. $\square$

In the next two examples, we determine two further values of $\tau_2(\pi, 2, \phi)$ (the first of which was already known). In contrast to the above example, we will not provide all the details. (That is hardly possible due to the shere size of these problems.) Nevertheless, our aim has been to provide sufficient information to enable the reader (preferably assisted by computer) to check our results.

Example 5.5. In a similar fashion, it is possible to show that $\tau_2(1,2,1) = 1/2$. This result is not new. In fact, it has been shown [2] that when $\pi \geq 1$ and $\phi \geq \beta - 1$, then $\tau_q(\pi,\beta,\phi) = 1/\beta$. Nevertheless, we think that this case is a good illustration of our method. First, we determine the time-invariant algorithms in this case. From Section 2 we have that the set of states S consists of all pairs (b, y) with $0 \leq b \leq 2$ and $y = 0, 1$. So there are precisely six states. Let S' denote all states (b, y) with $b = 1$. Note that since $\beta = 2$, we need only consider the behaviour of algorithms when the memory box contains a zero and a one, that is, the behaviour of algorithms on S'. Moreover, as observed in Section 2, the action of an algorithm f, when in state s' in S', upon input x, is fully determined once we know which symbol is taken out of the memory box.
These observations lead to the following conclusion. There are precisely 16 distinct time-invariant algorithms. Algorithm f_i, $0 \leq i \leq 15$, will be represented as

$$f_i :< (1,0)0, (1,0)1, (1,1)0, (1,1)1 > \longrightarrow < i_0, i_1, i_2, i_3 >,$$

$i = 8i_0 + 4i_1 + 2i_2 + i_3$, which is to be interpreted as follows: algorithm f_i, when in state $(1, y)$, upon input x, takes the symbol i_{2y+x} out of the memory box.
The next step is to construct, for each i, the matrix D_i associated with algorithm f_i. There are 63 non-empty subsets of states. Investigation of the directed graph obtained from the superactions reveals that all walks starting in superstate S eventually end up in a strongly connected subgraph spanned by only 10 "essential" superstates, namely

$\{(1,0)\}$, $\{(0,0),(1,0)\}$, $\{(2,0)\}$, $\{(1,0),(2,0)\}$, $\{(0,0),(1,0),(2,0)\}$, $\{(0,1)\}$, $\{(1,1)\}$, $\{(0,1),(1,1)\}$, $\{(1,1),(2,1)\}$, and $\{(0,1),(1,1),(2,1)\}$.

Consequently, when counting walks we may restrict attention to those walks that involve only these 10 essential superstates. So we let $\mathcal{D}$ consist of sixteen 10×10-matrices.
It turns out that the two matrixproducts D_5D_3 and D_3D_5 have the smallest possible normalised eigenvalue in $\mathcal{D}^+$, namely $\sqrt{2}$. Let

$$u = (2,2,1,2,2,1,2,2,2,2)^\top$$

denote the (unique) positive eigenvector of D_5D_3, and for each word $\mathbf{w} \in \{0,\ldots,15\}^*$, let

$$u(\mathbf{w}) = \mu^{-|\mathbf{w}|} D(\mathbf{w}) u.$$

Define the set of words W as

$$W = \{\wedge, 1, 4, 13, 16, 1\ 16, 16\ 1\},$$

and put

$$U = \{u(\mathbf{w}) \mid \mathbf{w} \in W\}.$$

(Here $\wedge$ denotes the empty word, and $u(\wedge) = u$.) So U contains seven vectors. It is then easily checked that $U^\uparrow$ constitute an eigenregion for $\mathcal{D}$, thus proving that $\mu(\mathcal{D}) = \sqrt{2}$. So we conclude that $\tau_2(1,2,1) = 1/2$. □

Example 5.6. Finally, we will use our method to show that $\tau_2(0,2,2) = \frac{1}{6}\log(\lambda)$, where λ is the largest real zero of $x^3 - 12x^2 - 4x - 1$.
As in Example 5.5, we first determine the time-invariant algorithms. Here, the set of states S consists of all pairs (x, b) with $0 \leq b \leq 2$ and $x = 0, 1$. So there are again precisely six states. Let S' denote all states (x, b) with $b = 1$.

Again, there are precisely 16 distinct time-invariant algorithms. Adopting a similar numbering scheme as in Example 5.5, we represent algorithm f_i, $0 \leq i \leq 15$, as

$$f_i :< (0,1)0, (0,1)1, (1,1)0, (1,1)1 > \longrightarrow < i_0, i_1, i_2, i_3 >,$$

$i = 8i_0 + 4i_1 + 2i_2 + i_3$, which is to be interpreted as follows: algorithm f_i, when in state $(z, 1)$, upon input x, takes the symbol i_{2z+x} out of the memory box.

We now construct the matrices D_i. In this case, it turns out that there are 47 "essential" superstates. So $\mathcal{D}$ consist of sixteen 47×47-matrices.

It appears that the matrix product $D = D_{15}D_1D_7D_0D_7D_1$, together with its cyclic shifts, have the smallest possible normalised eigenvalue in $\mathcal{D}^+$, namely $\mu(D) = \lambda^{1/6} = 1.51996\ldots$. Here $\lambda = \lambda(D)$ is the number defined above. To show this, we will describe an eigenregion for $\mathcal{D}$.

Put $\mu = \lambda^{1/6}$. Let u denote the (unique) positive eigenvector of D, and for each word $\mathbf{w} \in \{0, \ldots, 15\}^*$, let

$$u(\mathbf{w}) = \mu^{-|\mathbf{w}|} D(\mathbf{w}) u.$$

Define the set of words W' as

$$W' = \{\wedge, 1, 7\ 1, 0\ 7\ 1, 7\ 0\ 7\ 1, 1\ 7\ 0\ 7\ 1, 15\ 1\ 7\ 0\ 7\ 1\},$$

and put

$$U' = \{u(\mathbf{w}) \mid \mathbf{w} \in W'\}.$$

(Note that the six vectors in U' are precisely the eigenvectors of D and its "cyclic shifts" $D_1D_{15}D_1D_7D_0D_7, \ldots$.) These six vectors, together with a further set of 130 vectors, all of the form $u(\mathbf{w})$ for suitable words $\mathbf{w}$, make up a set of 136 vectors U for which $U^{\uparrow}$ constitute an eigenregion for $\mathcal{D}$. (These vectors can be obtained by repeated computation of images $\mu^{-1}D_iv$ of vectors v obtained earlier, and elimination of vectors in the set that dominate convex combinations of other vectors in the set. This elimination process involves solving LP-problems.)

As a consequence, we find that $\mu(\mathcal{D}) = \mu$, and hence $\tau_2(0,2,2) = {}^2\log \mu$. Strictly speaking, the above does not constitute a *proof* that $\tau_2(0,2,2)$ has the indicated value; this example is too large to check by hand, so all computations where done by computer. Nevertheless, in theory everything could be checked by hand (or assisted by computer), using only arithmetic in $\mathbf{Q}(\lambda)$.

We add a few comments on the optimal (periodic) strategy

$$f_{15}, f_1, f_7, f_0, f_7, f_1$$

for this problem. This strategy can of course be considered as a time-invariant ordering strategy, now acting on six-tuple inputs. With this interpretation, this strategy (eventually) involves only the six superstates

$\{(1,1)\}$, $\{(0,1),(1,1)\}$, $\{(0,0),(1,1),(0,2)\}$, $\{(0,0),(1,0),(0,1),(1,1),(0,2)\}$, and $\{(0,1),(1,1),(0,2),(1,2)\}$.

The restriction of D to these superstates is given by the matrix

$$\begin{pmatrix} 1 & 3 & 3 & 0 & 2 \\ 1 & 2 & 3 & 1 & 3 \\ 1 & 2 & 3 & 1 & 3 \\ 2 & 4 & 6 & 2 & 6 \\ 2 & 4 & 5 & 1 & 4 \end{pmatrix},$$

which has characteristic polynomial $x^5 - 12x^4 - 4x^3 - x^2$.
Further details concerning this example, notably on the precise nature of the set U, can be obtained from the authors. $\square$

From a practical point-of-view, Theorem 5.2 is very satisfactory. Based on this theorem, we have developed an iterative method which has enabled us to find eigenregions and entropies for every "natural" collection $\mathcal{D}$ that we investigated. (Example 5.6 in particular shows that even rather large problems can still be handled.)

However, many theoretical questions are left unanswered, and in that direction we have not achieved much. The main open problem is to find conditions which assure that $\mathcal{D}$ has an eigenregion (or is well-behaved) when $n > 1$. Although we strongly believe that $\mathcal{D}$ is well-behaved for example if all matrices D_i are positive, we have not been able to prove this. We have proved that under some rather general conditions (including the case of positive matrices referred to above) there always exists an eigenregion generated by *some* (possibly infinite) collection of nonnegative vectors. Further work on this problem, and on the iterative method mentioned above, will be reported elsewere.

For the sake of completeness, we include some negative results below. Let us say that $\mathcal{D}$ is *irreducible* if for each i, j, $i, j = 1, \ldots, d$, there is a matrix $D \in \mathcal{D}^+$ such that $D_{ij} > 0$. It might be tempting to conjecture that $\mathcal{D}$ is well-behaved if $\mathcal{D}$ is irreducible, but this conjecture is false, as is shown by the following example.

Example 5.7. Let $\mathcal{D} = \{D_0, D_1\}$, where

$$D_0 = \begin{pmatrix} 2 & 1 \\ 0 & 1/2 \end{pmatrix}, \qquad D_1 = \begin{pmatrix} 1/2 & 1 \\ 0 & 2 \end{pmatrix}.$$

It is easily seen that $\mu(\mathcal{D}) = 1$. However there is no finite set $U \geq 0$, $0 \notin U$, such that $\mathcal{D}(U)^{\uparrow} \subseteq U^{\uparrow}$. Indeed, suppose that there is such a set U. Then repeated application of D_0 shows that $U^{\uparrow}$ contains vectors with arbitrary small second component (and large first component). If U is finite, this is only possible if U contains some multiple of the vector $(1\ 0)^{\top}$. But this vector is eigenvector of D_1, with eigenvalue $1/2$, so 0 is contained in $U^{\uparrow}$, and $0 \in U$, contradicting our assumptions.
Now $\mathcal{D}$ is not irreducible. However, if we consider $\mathcal{D}' = \mathcal{D} \cup \{2J\}$, where J is the all-one matrix, then $\mathcal{D}'$ is irreducible, but we still have that $\mu(\mathcal{D}') = 1$, and the same reasoning as before shows that no finite set $U \geq 0$, $0 \notin U$, satisfies $\mathcal{D}'(U)^{\uparrow} \subseteq U^{\uparrow}$. (To see that $\mu(\mathcal{D}') = 1$, note that $2D_iJ, 2JD_i \geq D_1D_0$ and use the well-known fact that $\lambda(X) \geq \lambda(Y)$ if $X \geq Y$.) $\square$

Our last example shows that $\mu(\mathcal{D})$ need not be attained for a finite matrix product in $\mathcal{D}^+$.

Example 5.8. Let $\mathcal{D} = \{D_0, D_1\}$, where

$$D_0 = \begin{pmatrix} 1 & 1 \\ 0 & 2 \end{pmatrix}, \qquad D_1 = \begin{pmatrix} 3 & 1 \\ 0 & 1 \end{pmatrix}.$$

It is not difficult to show that $\mu(\mathcal{D}) = 3^{1/(1+{}^2\log 3)}$. (To this end, remark that $\mu(D_0^pD_1^q) = \max(2^p, 3^q)^{1/(p+q)}$, which is minimal if $2^p = 3^q$.) A simple argument then shows that no matrix D in $\mathcal{D}^+$ exists for which $\mu(D) = \mu(\mathcal{D})$. $\square$

6 Conclusions

We have introduced time-varying entropy-reduction algorithms (or strategies) over a finite collection $\mathcal{F}$ of finite-state input/output algorithms, and stated the optimal entropy-reduction problem for $\mathcal{F}$ as finding the minimal growth rate of the output space of a strategy over $\mathcal{F}$.

We have shown that the optimal entropy-reduction problem is equivalent to the problem of finding the smallest "normalized" Perron-Frobenius eigenvalue of a matrix in a related finitely-generated semigroup of non-negative matrices. A method is presented to derive lower bounds on these normalized eigenvalues, which in practice allows to determine the true minimum.

As an example of our approach, we solve a conjecture concerning ordering in sequence spaces. In this context, it would be interesting if our algebraic approach could be given some combinatorial interpretation.

Perron-Frobenius theory on non-negative matrices has found many applications. There is a vast literature on one-parameter semigroups of non-negative matrices, but as far as we know the problem considered here is new. Our results extend (part of) Perron-Frobenius theory to finitely-generated semigroups of non-negative matrices, but some important problems are left unresolved. It is our hope that this paper stimulates further interest in these problems, and that the methods presented here will find further applications.

References

[1] R. Adler, D. Coppersmith, and M. Hassner, "Algorithms for sliding-block codes", IEEE Trans. Inf. Theory, vol. IT-29, no. 1. pp. 5–22, Jan. 1983.

[2] R. Ahlswede, J-P. Ye, and Z. Zhang, "Creating order in sequence spaces with simple machines", Inform. and comp. 89, pp. 47-94, 1990.

[3] G. Birkhoff, "Lattice theory" (third edition), Amer. Math. Soc. Colloq. Publicns., Vol. XXV, Providence, R.I., 1967.

[4] G. Birkhoff, "Extensions of Jentzsch's theorem", Trans. Amer. Math. Soc., 85, pp. 219–227, 1957.

[5] J.C. Lagarias, Y. Wang, "The finiteness conjecture for the generalized spectral radius of a set of matrices", Linear Algebra Appl., vol. 214, pp. 17–42, 1995

[6] B.H Marcus, P.H. Siegel, and J.K. Wolf, "Finite-state modulation codes for data storage", IEEE J-SAC, vol. 10, no. 1, pp. 5–37, Jan. 1992.

[7] H. Minc, "Nonnegative matrices", John Wiley & Sons, 1988.

[8] G. Pólya and G. Szegö, "Aufgaben und lehrsätze aus der analysis I", aufgabe 98, Springer Verlag, 1970.

[9] E. Seneta, "Non-negative matrices and Markov chains", Springer Verlag, 1981.

[10] P. Vanroose, "Een ordeningsresultaat voor de situatie $(0, 2, 1, T^+)$" (in Dutch), report, Katholieke Universiteit Leuven, September 1989.

[11] J-P. Ye, "Towards a theory of ordening in sequence spaces", thesis, Universität Bielefeld, July 1988.

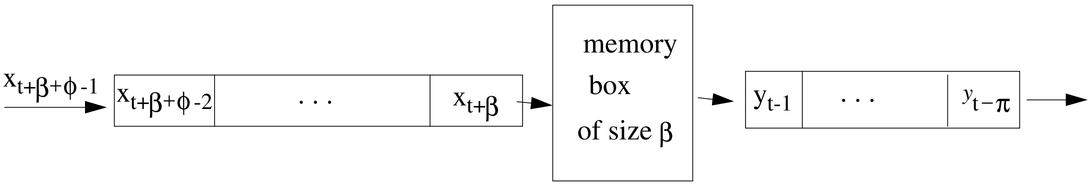

Figure 1: An ordering machine of type $(\pi, \beta, \phi)_q$

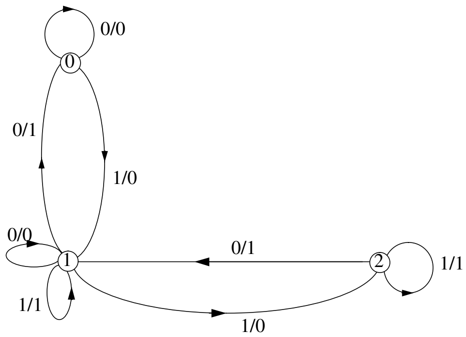

Figure 2: Admissible transitions in the state space