Extensions of the method of poles for code construction

Marie-Pierre Béal[*]

Abstract

The method of poles is a method introduced by P. A. Franaszek for constructing a rate $1:1$ finite state code from k-ary data into a constrained channel of finite type whose capacity is strictly greater than $\log(k)$. The method is based on the computation of a set of states called poles. To each pole is associated a set of paths going from this pole to others. Each set verifies an entropy condition. The code produced by the method of poles has a sliding block decoder if each set of paths satisfies moreover an optimization condition based on the sum of the path lengths of the set. In this paper we give a new optimization condition which guarantees the sliding block window decoding property and has a lower computational complexity than the previous one. We also extend the method of poles to the more general case of sofic constrained channels.

Index Terms: method of poles, sliding-block decoder, encoder, shift of finite type, sofic system, strongly synchronizing state.

1 Introduction

We are interested in the problem of encoding digital data into a constrained set of sequences. Typical applications are coding for storage systems or transmission systems. For digital magnetic storage for instance, constraints are run-length constraints on the sequences of bits stored. They are due to physical limitations of the storage systems. The constraints that can be modeled by a finite state machine are called rational constraints or sofic constraints. The problem is then to encode a free source of data, usually sequences of bits 0 and 1, into an available sequence, that is a sequence

[*]Institut Gaspard Monge, Université de Marne-la-Vallée, 77454 Marne-la-Vallée Cedex 2, France. `http://www-igm.univ-mlv.fr/~beal`.

that can be read as the label of a path of the automaton that models the constraints. With a fixed rate $p : q$ coding strategy, each block of p bits is encoded in a block of q bits, where p and q are small integers. After changing the alphabets, that is after recoding each source block of length p by one letter and each constrained block of length q by one letter, the coding (seen with these new alphabets), is a $1 : 1$ rate coding from an unconstrained source into a constrained channel.

In this paper we will consider first the class of constraints of finite type that we define as the constraints that can be recognized or represented by a deterministic local (or definite) automaton. An automaton is composed of a finite set of states and a finite set of edges. Each edge is defined by its origin state, end state, and a letter called its label. The automaton is deterministic if there is at most one edge going out of a given state and with a given label (the end state is then determined). The automaton is deterministic and local if it has in addition the following property : there is an integer n such that all equally labeled paths of length n end at the same state. The final state depends then only on the label. The method of poles is one method of channel coding for channels modeled by finite type constraints. The method constructs a rate $1 : 1$ encoder from arbitrary k-ary data into a given finite type channel S with a topological entropy $h(S)$, also called the Shannon capacity of the channel, strictly greater than $\log(k)$. The log is usually taken base 2. The algorithm starts with a finite local automaton $\mathcal{A}$ representing the channel, and builds a transducer, that is a finite automaton labeled by pairs of letters, whose input labeling is right closing (or deterministic with a finite delay), and whose output labeling is local. Since the input labeling is right closing, the encoding is sequential and since the output labeling is local the decoder is a sliding-block decoder and therefore propagates a symbol error in an encoded S-string by at most the decoder window length. The decoding is sometimes called state independent because the network of the decoder is then a purely combinatorial one. Final automata reductions tend to minimize the size of the transducer and hence the size of the decoding window. The construction of the output of the transducer is based on the computation of a set of states called principal states. It is possible to associate to each principal state a set of paths of the automaton $\mathcal{A}$. It consists in all paths going out of a given principal state and ending at another one. All possible sets of paths satisfy a Kraft condition on the lengths of the paths. For each principal state, the chosen set of paths is one that minimizes the set length (that is the sum of the path lengths of the set) among all possible sets. Finally the principal states that are not reached by any path of any set are removed and the remainders are called poles. This natural optimization condition is essential for the important feature of state independent decoding.

The method of poles and the computation of the set of principal states has been introduced by P. A. Franaszek in [14]. The above optimization condition and the proof of its influence on the sliding-window property were given in [8], (see also [9]). This method does not work in general in the case of equality of capacities of the source and of the constrained channel [4]. A different method, which allows also in the case of equality of capacities, has been obtained in [1] (see also [24], [25]). It is called the state splitting method or the ACH algorithm and involves state splittings of some states of the representation of the channel. Many other methods based on state splitting are described in [27] (see also [3], [6], [20]). These channel coding problems are closely related with the mathematical theory of symbolic dynamics [25]. In the case of a channel of finite type whose Shannon capacity is strictly greater than the capacity of the source, both methods lead in practice to transducers that can have about the same complexity. By complexity, we mean the size of the length of the window of the decoder and the size of the encoder. In the case of equality of capacities, the state splitting method is more powerful since the method of poles does not apply in general for these channels.

The aim of the paper is to improve the method of poles for constraints of finite type by choosing a better optimization condition. We also show that the method can be extended to more general sofic constraints. These two points can be merged and one can make the improved optimization condition work in the sofic case.

The first improvement is a complexity improvement on the optimization condition that ensures the state independent decoding property. The method of poles computes sets of paths of length at most an integer M which depends on the entropy of the constrained channel and of the geometry of the automaton. Let us consider an n-state local automaton which represents the constraints. Let us also assume that the Shannon capacity of the channel is strictly greater than $\log(k)$. In most applications, the integer n is small. The bound M of the length of the paths can asymptotically be k^n (see [4] for instance) and is nevertheless not too big in practice. For each principal state p, a set of paths is computed as a finite tree T_p of height M and whose arity is the maximal outdegree of the graph representing the channel. An algorithm for computing the set of principal states and of one possible tree associated to each principal state, is given by Franaszek in [14]. Its time complexity is exponential in M. A computational problem appears when we have to apply the optimization condition given in [8]. Indeed, if t_p is the number of nodes of T_p, the search of an optimal tree for this minimization condition requires to check all trees representing prefix sets of paths being prefixes of paths of T_p. This computation is exponential in the sum of the sizes t_p. We present here a new optimization condition which allows us to compute optimal trees in a linear time in the sum of the sizes t_p. We prove that the optimal trees obtained with this new optimization condition still lead to a sliding-block window decoder.

In the second part of the paper, we show how to extend the method of poles to sofic constraints that are no more of finite type. This is possible after a preliminary transformation of the finite state machine that models the constraints, in order to make the representation contain some special states called strongly synchronizing states. We describe this transformation than can be done with easy rounds of state splittings or also with a fibered product of two automata. This preprocessing is very different and much simpler than the ACH algorithm [1]. We then show that the method of poles can be performed on this sofic representation and still builds a $1:1$ encoder with sliding block decoder.

Another method to solve the case of sofic constraints has been described by R. Karabed and B. Marcus in [21]. It is based on the state splitting process of [1]. Compared to their method, our method does not allow us, in general, to treat the case of equality of capacities for the class of almost of finite type constraints (see [21]). But the method of poles for sofic shifts gives a practical alternative to state splitting methods that are rather complicated in the sofic case.

In Section 2 we give the primary definitions and we briefly recall the method of poles. As it is used in Section 3, we describe the computation of the set of principal states. The algorithm of construction of the trees obtained with the new optimization condition is given in Section 3. We prove here that we do not lose the state independent decoding property. The extension of the method of poles to the case of sofic constraints is described in Section 4. In this paper, we will sometimes describe algorithms as programs written in a pseudo code. We adopt some conventions given in [12, p. 4].

2 Definitions and background

Let A be an alphabet, that is, a finite set of symbols called letters.

A *finite state automaton* $\mathcal{A} = (Q, E)$ on the alphabet A is composed of two finite sets: Q, the set of states, and E, the set of edges. The set of edges is included in $Q \times A \times Q$. A finite path is a finite sequence of consecutive edges $((q_i, a_{i+1}, q_{i+1}))_{0 \le i < n}$, the word $a_1 a_2 \dots a_n$ being the label of the path.

A finite automaton is *deterministic* if and only if, whenever there are two edges (p, a, q) and (p, a, r) then $q = r$.

A finite automaton is *local* if there are three nonnegative integers n, m, a with $m+a = n$ such that whenever two finite paths $((q_i, a_{i+1}, q_{i+1}))_{0 \le i < n}$ and $((q'_i, a_{i+1}, q'_{i+1}))_{0 \le i < n}$ have the same label $w = a_1 a_2 \dots a_n$, then $q_m = q'_m$. The integer m is for memory and a for anticipation. If $\mathcal{A}$ is moreover deterministic, it is possible to have a null anticipation, or, equivalently, to take $m = n$. Local automata are also called *definite automata* [28]. An automaton is said to be *irreducible* if its graph is strongly connected. A constraint channel S is said to be recognized or represented by the automaton $\mathcal{A}$ if it is the set of labels of bi-infinite paths of the automaton. Such a constrained channel is called a *sofic channel.* If it can be represented by a local automaton, the constraint or the channel is said *of finite type.* It is characterized by a finite set of finite blocks avoided by any bi-infinite sequence of the channel. This set is called a set of forbidden words for the channel. It is possible to represent the channel by an automaton which is both deterministic and local.

A word w of length $m + a$, where m and a are nonnegative integers, is said to be (m, a)-*synchronizing* if there is a state p such that each path $(q_i, a_{i+1}, q_{i+1})_{0 \le i < m+a}$ labeled by w satisfies $q_m = p$. We say in this case that w synchronizes onto the state p. A word w is said to be *synchronizing* if it is (m, a)-synchronizing for some m and a.

We introduce the notion of strongly synchronizing states that will be useful in the last section to extend the method of poles to sofic constraints. For each state p and each nonnegative integers m, a, we define the set $E_p^{(m,a)}$ of finite words uv, where u is the label of a path ending at p, and v the label of a path starting at p. A state p of an automaton $\mathcal{A}$ is said to be (m, a)-*strongly synchronizing*, where m and a are nonnegative integers, if for any state q distinct from p,

$$E_p^{(m,a)} \cap E_q^{(m,a)} = \emptyset.$$

A state p is said to be *strongly synchronizing* if it is (m, a)-strongly synchronizing for some m and a. A state p of an automaton $\mathcal{A}$ is thus strongly synchronizing if and only there are not two distinct equally labeled bi-infinite paths such that the first one goes through a state p at some index, and the second one goes through a state $q \neq p$ at the same index. This property is computable in a polynomial time in the number of states of $\mathcal{A}$ (see [9] page 72).

Example. All states of a local automaton are strongly synchronizing.

We are going to encode a free k-ary source into the constraint channel S represented by an irreducible, deterministic and local automaton. We assume that the topological entropy, or the Shannon capacity, $h(S)$ of S satisfies $h(S) > \log(k)$. The entropy is computed as the log of the spectral radius of the adjacency matrix of the graph of the automaton $\mathcal{A}$. It is defined as the limit of $1/n \ \log(\text{card}(A^n \cap S_n)$, where S_n is the set of blocks of length n that can appear as a subblock of a bi-infinite sequence of S.

The first step in the method of poles consists in finding a subset P of states of Q called *principal states*, such that there exists a positive integer M (as small as possible) such that one can associate to each principal state p a finite prefix set Z_p of finite paths that satisfy the following properties

1. each path of Z_p is a path of $\mathcal{A}$ that begins at state p;
2. each path of Z_p ends at some state of P;
3. the length of each path of Z_p is less than or equal to M;
4. the set Z_p satisfies the Kraft inequality
$$\sum_{z \in Z_p} \frac{1}{k^{l(z)}} \geq 1,$$
where $l(z)$ denotes the length of the path z.

We recall that a set of path is a prefix set[1] if no path is the strict beginning of another one. A maximal subset of Q satisfying these above conditions is unique and is called the *set of principal states* of the automaton.

It is shown in [8] that if $h(S) > \log(k)$, then, for a sufficiently large integer M, such a nonempty set P of principal states always exists. The search begins with $M = 1, 2, \cdots$, (M is incremented by 1 at each step when the previous one has failed). Once a nonempty set P of principal states of $\mathcal{A}$ is found, the maximal size M of the lengths of the paths of any possible set Z_p is fixed.

Franaszek's algorithm gives then, for each principal state p of P, a prefix set of paths Z_p satisfying the above conditions, which first maximizes the sum
$$\sum_{z \in Z_p} \frac{1}{k^{l(z)}},$$

[1] also called a prefix free set.

and, in addition to this maximization condition, minimizes what we call the length $l(Z_p)$ of Z_p defined by

$$l(Z_p) = \sum_{z \in Z_p} l(z). \tag{1}$$

We can here remark that the last minimization condition is a local minimization condition in the sense that it does not imply that a chosen set Z_p has a minimal length among all sets that satisfy conditions 1 to 4.

We describe below Franaszek's algorithm in a pseudo code. The computation of the set of principal states can be performed as follows

Computation of the set of principal states
begin
  $P \leftarrow Q$ //where Q is the set of states of $\mathcal{A}$
  while ($P \neq \emptyset$ and there is a state $p \in P$ with $S_P(p) < 1$)
  do $P \leftarrow P - \{p\}$
end,
where $S_P(p)$ is the maximum of the sums

$$\sum_{z \in Z_p} \frac{1}{k^{l(z)}},$$

for all possible choices of prefix sets Z_p of paths satisfying conditions 1 to 4.

We now describe the computation of the predicate $\{S_P(p) < 1\}$ for a state p in P. Recall that M is a fixed integer that bounds the lengths of the paths considered. We first build a tree T of height M whose nodes represent the paths in $\mathcal{A}$ of length less than or equal to M starting at p. The nodes of T are labeled by states of $\mathcal{A}$ and we denote by r the root labeled by p. If n is a node, its height is the length of the path from the root to the node. Each node at height at most $M-1$ labeled by a state q admits a son labeled by s for each edge (q, a, s) in $\mathcal{A}$. This completely defines a tree that is a covering tree starting at p and of height M, of the automaton $\mathcal{A}$. The size of the tree is its number of nodes.

We assign a boolean mark to each node of the tree T. A node is `marked` or `unmarked`. It is `marked` if its label belongs to set P and `unmarked` otherwise. We then associate to each node a rational value. The value of a node n is denoted by $v(n)$ in the pseudo code below. The computations necessary to calculate v are linear in the size t of the tree. They are performed bottom-up from the leaves to the root of T.

COMPUTATION OF $\{S_P(p) > 1\}$

begin

if (n is a leaf)

  then if (n is `marked`) **then** $v(n) \leftarrow 1$ **else** $v(n) \leftarrow 0$

  else $//n$ is not a leaf

    if (n is `marked` and distinct from the root)

      then $v(n) \leftarrow \max(1, \sum_{s \text{ sons of } n} \frac{v(s)}{k})$

      else $v(n) \leftarrow \sum_{s \text{ sons of } n} \frac{v(s)}{k}$

end

It is easy to verify that $\{S_P(p) \geq 1\}$ if and only if $v(r) \geq 1$. We point out that this computation depends on the current set P and can thus be performed several times for a same state p during the search of the principal states. Once the set of principal states is obtained, the same algorithm can be slightly modified to produce the Franaszek's sets of paths Z_p of Equation (1). We more precisely compute, from the covering tree T of $\mathcal{A}$ starting at p and of height M, a tree whose set of paths from the root to the leaves is Z_p.

COMPUTATION OF THE FRANASZEK TREE OF THE STATE p

begin

if (n is a leaf)

  then if (n is `marked`) **then** $v(n) \leftarrow 1$ **else** $v(n) \leftarrow 0$

  else begin $//n$ is not a leaf

    let $v = \sum_{s \text{ sons of } n} \frac{v(s)}{k}$

    if (n is `marked` and distinct from the root)

      then if ($v \leq 1$)

        then $v(n) \leftarrow 1$ and cut all branches under the node n

        else $v(n) \leftarrow v$ and cut the link between n and each of its sons s such that $v(s) = 0$

      else $v(n) \leftarrow v$ and cut the link between n and each of its sons s such that $v(s) = 0$

    end

end

In the sequel, we denote by T_p the final tree associated to each principal state p computed by the above algorithm. We refer to it as the Franaszek tree associated to the principal state p, or also as the principal tree associated to p.

Example. We give below an example of computation of the set of principal states and of the trees T_p, where p is a principal state. We consider the local automaton of Figure 1 which represents a constrained channel of entropy greater than $\log(2)$. We choose $k = 2$ and compute the set of principal states for a maximal path length M equal to 1. The search fails by computing an empty set. We try again with $M = 2$. The set of states P is initialized to $\{1, 2, 3\}$. The marked nodes are circled and the values of the function v are given in the squares beside each node of the tree.

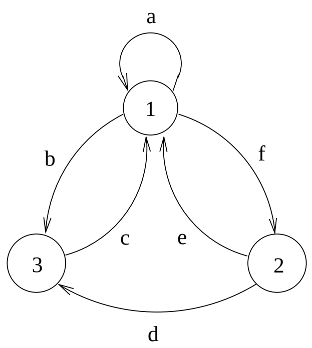

Figure 1: A local automaton

First step We have $P = \{1, 2, 3\}$. We check if $\{S_P(3) \geq 1\}$. The computation of v is done on the tree of Figure 2 whose root is denoted by r. We get $v(r) = 3/4$ and we remove state 3 from the set P.

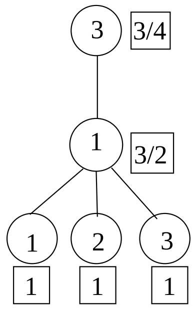

Figure 2: First step.

Second step We have $P = \{1, 2\}$. We check if $\{S_P(2) \geq 1\}$. The computation of v is done on the tree of Figure 3. We again get $v(r) = 3/4$ and we remove state 2 from the set P. In this computation, some branches have been cut during the process. This is symbolized by a dashed line.

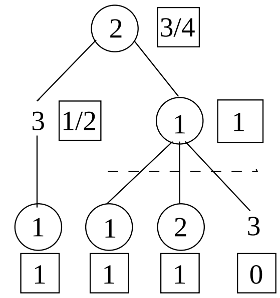

Figure 3: Second step.

Third step We have $P = \{1\}$. We check $\{S_P(1) \geq 1\}$. The computation of v is done on the tree of Figure 4. We now get $v(r) = 1$ and then $S_P(1) = 1$. The set of principal states is $\{1\}$. Some branches have been cut during the process. This is symbolized by a dashed line. We obtain the Franaszek tree T_1.

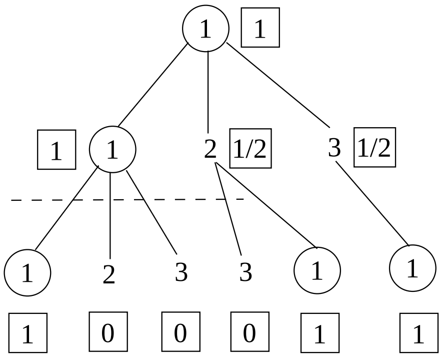

Figure 4: Third step.

To each tree T_p, where p is a principal state, is associated in a natural way a set of paths Z_p satisfying the conditions 1 to 4 defined as the set of paths of the tree going from the root to a leaf. Each such path corresponds to a path in $\mathcal{A}$. Since condition 4 is satisfied, that is,

$$\sum_{z \in Z_p} \frac{1}{k^{l(z)}} \geq 1,$$

it is possible to extract from Z_p a subset Z'_p such that the above Kraft inequality becomes a Kraft equality

$$\sum_{z \in Z_p} \frac{1}{k^{l(z)}} = 1.$$

In order to minimize the number of notations, we still call this new set Z_p. We moreover assume, by possibly removing some unuseful states in P, that the following condition is satisfied: for each pair of principal states p, p', there is a concatenation of paths of $\bigcup_{p \in P} Z_p$ going from p to p'.

Now a transducer $\mathcal{T}$, used to encode and decode, can be constructed as follows. For each principal state p, we choose a prefix code X_p on a k-letter alphabet, that has the same length distribution as Z_p. We choose a length-preserving bijection ρ_p from Z_p to X_p. We define a state $\hat{p}$, called a pole, for each principal state p. For each path z in Z_p, the transducer has a path $\hat{z}$ of length $l(z)$ from state $\hat{p}$ to state $\hat{p'}$, where p' is the terminal state of the path in $\mathcal{A}$. One can imagine $l(z) - 1$ dummy states of $\mathcal{T}$ strung along the path $\hat{z}$. To the path $\hat{z}$ is assigned as input label $\rho_p(z)$, and as output label, the label of the path z in $\mathcal{A}$.

As shown in [9, p. 172], this process does not ensure that the output automaton of the transducer is a local automaton, and therefore that the decoding is sliding block. This is based upon the fact that the sets Z_p have to satisfy a length optimization condition different from the condition of Equation (1), in order to get the state independent decoding property.

3 A new optimization condition in the method of poles

In [8] is given a length optimization condition that ensures that the transducer has a local output, and then a sliding block window decoder. The condition is to choose, among all possible sets Z_p of paths satisfying conditions 1 to 4, a set that minimizes the sum of the path lengths. This optimization condition is a global optimization condition compared to the local one described in the definition of Franaszek's sets Z_p (see the remark below the definition of the sets Z_p in the second section). This requires a search of all prefix sets of paths satisfying conditions 1 to 4, the paths being prefixes of paths of the Franaszek tree T_p obtained in the previous section. If we denote by t_p the size of the tree T_p, that is the number of nodes of T_p, this exhaustive search has an exponential time cost in t_p.

In this section, we give a new optimization condition that can be computed from the Franaszek trees in a linear time.

We call *subtree* of a tree T the part of T formed by a node of T with all its descendants. We call *principal subtree* of a Franaszek tree T a subtree of T whose root is not a leaf of T and is labeled by a principal state.

We now give a family of new optimization conditions. We assume that the principal subtrees are preordered with a preorder, denoted by ord, which satisfies the following condition:

- if A and B are two distinct principal subtrees of a Franaszek tree,

$$A \text{ is a strict subtree of } B \Longrightarrow \operatorname{ord}(A) < \operatorname{ord}(B). \tag{2}$$

We choose, for each principal state p, a principal subtree of any Franaszek tree, which is rooted by a node labeled by p, and minimizes the preorder ord. We denote it by B_p. We point out that the trees $(B_p)_{p \in P}$ satisfy the Kraft inequality condition since they are principal subtrees of some tree T_q. This is due to the fact that each node labeled by a principal state of a Franaszek tree has a final value v, computed during the computation of the Franaszek trees, which is greater than or equal to 1. This property is equivalent to the fact that the principal subtree satisfies the Kraft inequality. A final extraction consists in removing some branches of the principal subtrees $(B_p)_{p \in P}$ to get finally sets of paths which satisfy the Kraft equality. The transducer used to encode and decode is built from the new sets Z_p as explained in Section 2. It has a strongly connected graph.

The following proposition states that the important property of state independent decoding is guaranteed.

Proposition 1 *The output labeling of the transducer constructed from the trees $(B_p)_{p \in P}$ is a local automaton.*

Proof : We prove that the transducer constructed from the trees $(B_p)_{p \in P}$ that we get before the final extraction has a local output. Since this automaton is obtained from the final pruned trees $(B_p)_{p \in P}$ by removing some paths from it, it will prove the result.

Recall that an irreducible automaton is local if and only if it does not admit two distinct equally labeled cycles. Let us thus consider two distinct equally labeled cycles of the output automaton of the transducer $\mathcal{T}$, denoted by $((p_i, a_{i+1}, p_{i+1})_{0 \leq i < n}$ and $((p'_i, a_{i+1}, p'_{i+1})_{0 \leq i < n}$. The addition on the set of indices $\{0, 1, \ldots, n-1\}$ of the cycles, has to be understood modulus n. We first consider the case where there is an index i such that p_i and p'_i are both poles of the transducer. The two cycles of the output of $\mathcal{T}$ project onto two cycles of the automaton $\mathcal{A}$ and the two poles p_i and p'_i project onto two principal states. Since the principal states of $\mathcal{A}$ are strongly synchronizing, the two projected principal states are equal, and then the poles p_i and p'_i also. It follows that the two cycles of $\mathcal{T}$ are also equal, by construction of the transducer.

We can now assume that for each index i, p_i and p'_i are not simultaneously poles. Since each cycle of $\mathcal{T}$ goes through a pole, there is at least one index

i such that p_i is a pole and p'_i is not one. The two cycles of the transducer project onto two cycles in $\mathcal{A}$, one going through the projection of p_i, the other one going through the projection of p'_i at the same time. Since the projection of p_i is a principal state which is strongly synchronizing, the projections of p_i and p'_i are equal. We call it the projection of the pair (p_i, p'_i). Since we are only interested in the pairs (p_j, p'_j) such that p_j or p'_j is a pole, we can assume, after renumbering, that all pairs are like this, two states of consecutive indices being linked by a path of length at least one. We divide these indices into two disjoint sets, one set A where p_j is a pole and one set A' where p'_j is a pole. We restrict then our attention only to the boundary set I consisting of all indices j such that

$$(j \in A \text{ and } (j+1) \in A') \text{ or } (j \in A' \text{ and } (j+1) \in A).$$

Let us assume that the set I is $\{0, 1, \dots, r-1\}$ and that the addition on I is modulus r. We consider then the circular sequence $(e_j)_{0 \leq j < r}$ of the projections of pairs of states (p_i, p'_i) indexed by successive points of I modulus r (see Figure 5 where the poles are circled).

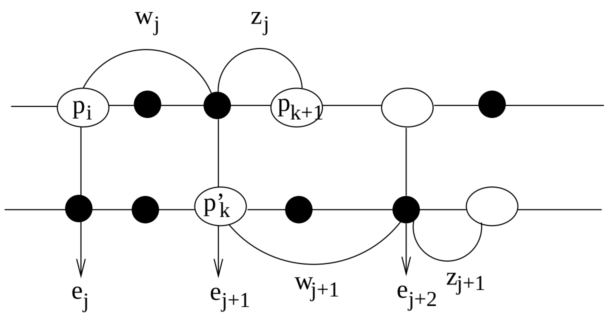

Figure 5: Paths encroaching upon each other

Let e_j be one element of this sequence which is the projection of a pair (p_i, p'_i). Let us assume that p_i is a pole (the case where p'_i is a pole is symmetric). Then e_{j+1} is the projection of a pair (p_k, p'_k) where p'_k and p_{k+1} are poles. Let us denote by w_j (resp. z_j the path from p_i to p_k (resp. from p_k to p_{k+1}) in the first cycle. The projected path onto $\mathcal{A}$ of $w_j z_j$ belongs to the set of paths associated to the tree B_{e_j}. Since the states p_k and p'_k project onto the same principal state of $\mathcal{A}$, the state e_{j+1}, the tree B_{e_j} has a proper principal subtree B rooted by e_{j+1}. By definition of the optimized trees $(B_p)_{p \in P}$, we then have

$$\operatorname{ord}(B_{e_{j+1}}) \leq \operatorname{ord}(B) < \operatorname{ord}(B_{e_j}).$$

This implies that for each index j in I, $\mathrm{ord}(B_{e_{j+1}}) < \mathrm{ord}(B_{e_j})$. It ends the proof by a contradiction since the sequence $(e_j)_{0 \leq j < r}$ is finite. $\square$

We now give some examples of possible preorders on principal subtrees satisfying condition (2):

- the height of a tree.
- the length of a tree, that is the sum of the lengths of all paths going from the root to each leaf.

For practical applications, we adopt this second preorder as optimization condition. This new optimization condition can be stated as follows. We compute and associate to each principal state p a principal subtree B_p, rooted by a node labeled by p, which has a minimal length among all principal subtrees of all Franaszek trees. This choice is due to two reasons. First, the size of the encoding transducer depends on the length of the trees $(B_p)_{p \in P}$. The decoding window length is also bounded above by a function of the size of this transducer. Second, the computation of the trees $(B_p)_{p \in P}$ is linear in the sum of the sizes of the Franaszek trees.

We precise below the computation of the trees $(B_p)_{p \in P}$ from the Franaszek trees. A bottom up computation of these trees is possible as follows. We first compute the lengths of all subtrees of the principal trees. This is easily done by computing together for a node n the pair (l_n, x_n) of the length l_n of the subtree rooted by this node, and the number x_n of leaves of this subtree. If n is itself a leaf, we have $(l_n, x_n) = (0, 1)$. If n is a node which has s sons denoted by $1, 2, \ldots, s$, we have the following trivial equalities:

$$\begin{aligned} x_n &= \sum_{i=1}^{s} x_i, \\ l_n &= \sum_{i=1}^{s} (l_i + x_i) = \sum_{i=1}^{s} l_i + x_n \end{aligned}$$

Let us now denote by N the set of roots of all principal subtrees of all Franaszek trees and by $c(n)$ the label of a node n in N. We define B_p as the subtree rooted by n where n is a node of the forest $(B_p)_{p \in P}$ such that

$$l_n = \min\{l_p \mid p \in N \text{ and } c(n) = p\}.$$

A bottom-up exploration of the forest $(T_p)_{p \in P}$ allows simultaneous computations of pairs (l_n, x_n) and of the forest $(B_p)_{p \in P}$. Since each tree is explored once, the time complexity of the computation is $O(\sum_{p \in P} t_p)$.

Example. We consider the example of the channel of entropy greater than log(2) pictured in Figure 6. The trees $(T_p)_{p\in P}$ are given in Figure 7. The set of principal states with $M = 2$ is $P = \{1, 2, 3, 4\}$. The trees $(B_p)_{p\in P}$ obtained at the end of the computation have their root pointed in the figure. The value l_n for a node n is given beside the node. The nodes are labeled by their projection state onto the automaton of Figure 6 that represents the channel. A final operation, made in order to get trees that satisfy the Kraft equality, consists in removing some branches of the trees obtained at the previous step. This is shown in Figure 8.

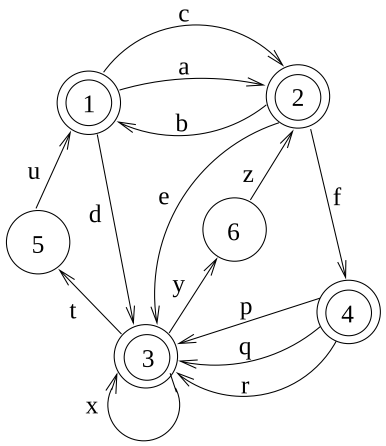

Figure 6: Channel of entropy greater than log(2)

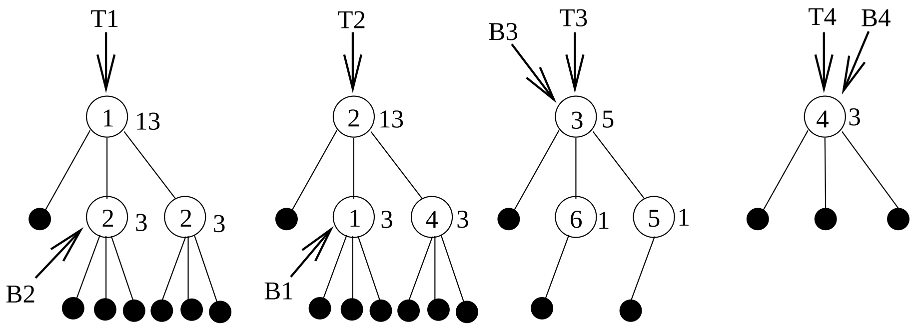

Figure 7: Computation of the forest $(B_p)_{p\in P}$

Here we can remark that we have $B_p = T_p$ for at least one principal state p, since otherwise, we could choose a smaller integer M that bounds the path lengths during the computation of the principal states.

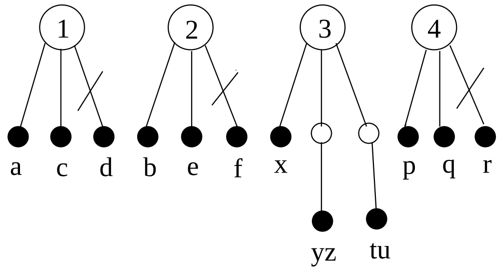

Figure 8: Final extraction

4 Extension of the method of poles to sofic constraints

In this section, we extend the method of poles to the more general class of constraints represented by a not necessarily local automaton. We consider a *transitive* sofic channel S, that is, a channel that can be represented by an irreducible automaton. If it is not the case, it is always possible to consider a subset of the channel that has this property and the same capacity. We also assume that the entropy of S is strictly greater than $\log(k)$, where k is a positive integer. In order to extend the method, we will use state splittings of states of the representation of the channel. We first give the definition of the notion of state splitting, which comes from symbolic dynamics.

We define the operation of *output state splitting* in an automaton $\mathcal{A} = (Q, E)$. Let q be a vertex of Q and let I (resp. O) be the set of edges coming in q (resp. going out of q). Let $O = O' + O''$ be a partition of O. The operation of *(output) state splitting* relative to (O', O'') transforms $\mathcal{A}$ into the automaton $\mathcal{B} = (Q', E')$ where $Q' = (Q \setminus \{q\}) \cup \{q'\} \cup \{q''\}$ is obtained from Q by splitting state q into two states q' and q'', and where E' is defined as follows (see Figure 9 and 10)

1. All edges of E that are not incident to q are left unchanged.

2. The states q' and q'' have the same input edges as q.

3. The output edges of q are distributed between q' and q'' according to the partition of O into O' and O''. We denote U' and U'' the sets of output edges of q' and q'' respectively
   $U' = \{(q', x, p) \mid (q, x, p) \in O'\}$ and $U'' = \{(q'', x, p) \mid (q, x, p) \in O''\}$.

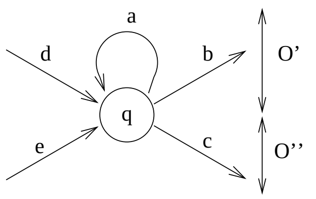

Figure 9. Automaton $\mathcal{A}$

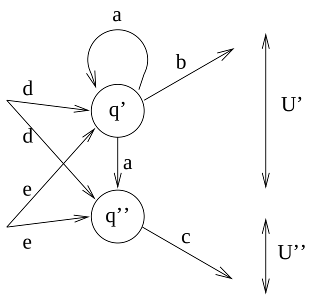

Figure 10. Automaton $\mathcal{B}$

The notion of *input state splitting* is defined similarly.

We now transform the automaton that represents the constraints into another one that has at least one strongly synchronizing state.

Proposition 2 *A transitive sofic channel admits a representation that has at least one strongly synchronizing state.*

Proof : It is known that a transitive sofic channel has a unique minimal deterministic representation. This representation admits a synchronizing word. It is called the minimal automaton in automata theory and the Fischer cover in the symbolic dynamics theory (see for instance [10, p. 478], [23] or [9]).

Let $\mathcal{A}$ be such a representation and let w be a (m,a)-synchronizing word onto a state p. We associate to each state q the set $E_q^{(m,a)}$ of pairs (u,v) of finite words, where u is the label of a path ending at q, and v the label of a path starting at q. The pairs (u,v) are also denoted by $u \cdot v$. One can remark that if $u \cdot v$ and $u' \cdot v'$ belongs to a set $E_q^{(m,a)}$, then $u \cdot v'$ and $u' \cdot v$ also. We now consider for each state q the longest prefix z_q (possibly equal to the empty word) of all words v such that there is a word u with $u \cdot v \in E_q^{(m,a)}$. We choose a state r such that z_r has a minimal length among all $(z_q)_{q\in Q}$. If the length of z_r is strictly less than the anticipation a, the set $E_r^{(m,a)}$ contains two pairs $u \cdot z_r bv$ and $u' \cdot z_r cv'$, where u, u', v, v' are words, and b, c distinct letters. If z_r is not the empty word, let d be its first letter. We define the word x_r by $z_r = dx_r$. We do an output state splitting of state r by partitioning the outgoing edges of r in the ones ending at a state s such that $x_r b$ is a prefix of z_s and the other ones. If z_r is the empty word, we do an output state splitting of state r by partitioning the outgoing edges of r in the ones labeled by b and the other ones. This state splitting process of the automaton is iterated from the new automaton obtained. This process always stops since if a state q is split in q_1 and q_2, the cardinalities of $E_{q_1}^{(m,a)}$

and of $E_{q_2}^{(m,a)}$ are strictly less than the cardinality of $E_q^{(m,a)}$. The automaton computed at the last step is such that all words z_q have a length equal to a.

We do the symmetrical operations with the longest suffixes y_q of all words u such that there is a word v with $u \cdot v \in E_q^{(m,a)}$. We use this time input state splitting. The final automaton that we get is such that, for all of its states q, the word y_q has length m, and the word z_q has length a. This means that each set $E_q^{(m,a)}$ of the final automaton is reduced to one pair $y_q \cdot z_q$. Since w is a synchronizing word of the initial automaton, y_p is the prefix of length m of w, and z_p is its suffix of length a. This state is a strongly synchronizing state of the final automaton. □

We mention that another proof of the previous result can be obtained by doing a direct (or fibered product) of the initial automaton that recognizes the channel and a (m, a)-local universal De Bruin automaton, (we refer for instance to [9] for this notion). In the above proof, the order chosen to treat the past and the future can be changed. The sequences of input and output state splittings can be merged. The interest of the state splitting way versus the product of automata is that one can stops the process as soon as we have obtained enough strongly synchronizing states.

Let S be a transitive sofic channel recognized by an automaton $\mathcal{A}$ with an entropy $h(S) > \log(k)$. By the previous proposition we can assume that $\mathcal{A}$ has a nonempty set of strongly synchronizing states. A set of principal states for an integer M is obtained like in Section 3 by starting this time the computation with a set P reduced to the strongly synchronizing states only.

Computation of the set of principal states
begin
  $P \leftarrow$ the set of strongly synchronizing states
  while ($P \neq \emptyset$ and there is a state q with $S_P(q) < 1$)
  do $P \leftarrow P - \{q\}$.
end,
where $S_P(q)$ is the maximum of the sums:

$$\sum_{z \in Z_q} \frac{1}{k^{l(z)}},$$

for all possible choices of prefix sets Z_q of paths satisfying conditions 1 to 4.

Since the set of strongly synchronizing states is not empty and since the Shannon capacity of the channel is strictly greater that $\log(k)$, one can prove like for constraints of finite type (see Section 2), that a nonempty set of principal states is found by increasing M if the search fails with an empty set.

The construction of a coding and decoding transducer in then done exactly like in the previous section. The proof that its output automaton is a local automaton is the same. It is due to the strongly synchronizing property of the principal states.

Example Let us consider the transitive sofic system recognized by the automaton $\mathcal{A}$ of Figure 11. Its entropy is strictly greater than $\log(2)$. This automaton is the minimal deterministic representation of the system. It admits at least one synchronizing word: the word bb, which is $(2,0)$-synchronizing. It also has a strongly synchronizing state: the state 4, which is $(2,0)$-strongly synchronizing. In order to get as many strongly synchronizing states as possible, we do a sequence of input state splittings and get the automaton $\mathcal{B}$ of Figure 12 where the set $E_p^{(2,0)}$ is represented inside each state p.

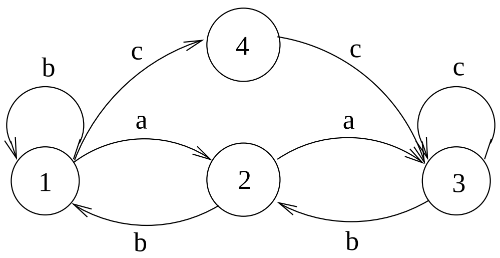

Figure 11: Automaton $\mathcal{A}$

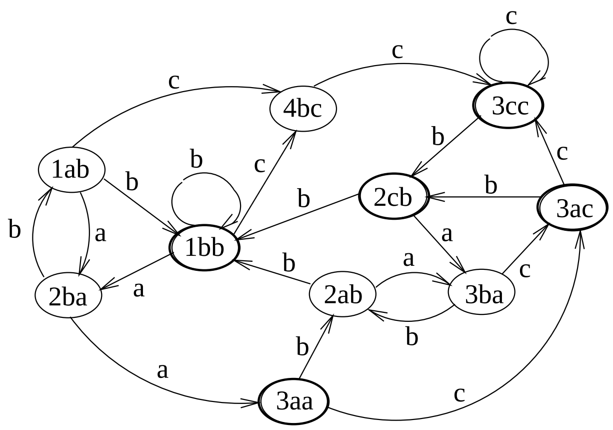

Figure 12: Automaton $\mathcal{B}$

The set of strongly synchronizing states of the automaton $\mathcal{B}$ is

$$S = \{(1, bb), (4, bc), (2, cb), (3, cc), (3, ac), (3, aa)\},$$

where a state p is denoted here by its projection state onto $\mathcal{A}$ and the left component of the unique pair of $E_p^{(2,0)}$. A nonempty set of principal states is obtained for $M = 5$. The set of poles is then

$$P = \{(1, bb), (2, cb), (3, cc), (3, ac), (3, aa)\}.$$

The labels of the paths Z_p associated to each pole p are:

$$\begin{aligned}
Z_{(3,aa)} &= \{c, bac, bbb, bbaa, bbcc, babb, babac, bbabb\} \\
Z_{(2,cb)} &= \{ac, bb, abb, baa, bcc, abac, babb\} \\
Z_{(1,bb)} &= \{b, cc, aa\} \\
Z_{(3,ac)} &= C_{(3,cc)} = \{b, c\}
\end{aligned}$$

Since the optimized trees associated to the poles $(3, ac)$ and $(3, cc)$ are the same, one can merge these two poles in the transducer. The transducer obtained has 4 poles and 41 states for the integer $M = 5$. A better transducer can be obtained with an initial transformation of the automaton $\mathcal{B}$ of Figure 12. If the state $(1, ab)$ is removed for instance, the channel represented has an entropy which is still greater than $\log(2)$. The number of strongly synchronizing states is then advantageously increased in

$$S = \{(1, bb), (4, bc), (2, cb), (3, cc), (3, ac), (3, aa), (2, ab)\},$$

and the following set of poles is obtained with $M = 2$ only

$$P = \{(1, bb), (2, cb), (3, cc), (3, ac), (3, aa), (2, ab)\}.$$

With a final automata reduction (state merging), we get the very small encoding transducer of Figure 13. Its sliding block decoding window length is only 2.

5 Acknowledgment

We thank anonymous referees for helpful comments.

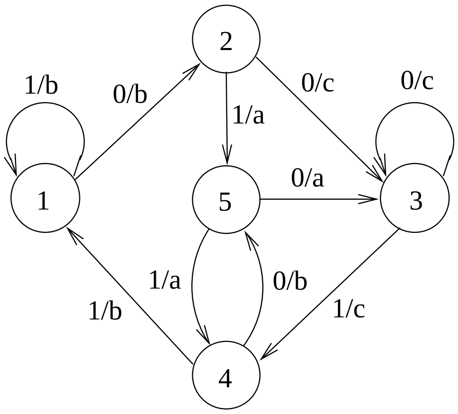

Figure 13: Encoding Transducer

References

[1] Adler, R., Coppersmith, D., and Hassner, M. Algorithms for sliding block codes. *IEEE Trans. Inform. Theory 29*, 1 (1983), 5–22.

[2] Ashley, J. A linear bound for sliding block decoder window size. *IEEE Trans. Inform. Theory 34*, 3 (1988), 389–399.

[3] Ashley, J. A linear bound for sliding block decoder window size (II). *IEEE Trans. Inform. Theory 42*, 6 (1996), 1913–1924.

[4] Ashley, J., and Béal, M.-P. A note on the method of poles for code construction. *IEEE Trans. Inform. Theory 40*, 2 (1994), 512–517.

[5] Ashley, J., Karabed, R., and Siegel, P. H. Complexity and sliding block decodabibity. *IEEE Trans. Inform. Theory 42*, 6 (1996), 1925–1947.

[6] Ashley, J., Marcus, B. H., and Roth, R. M. Construction of encoders with small decoding look-ahead for input-constrained channels. *IEEE Trans. Inform. Theory 41*, 1 (1996), 55–76.

[7] Ashley, J., Marcus, B. H., and Roth, R. M. On the decoding delay of encoders for input constrained channels. *IEEE Trans. Inform. Theory 42*, 6 (1996), 1948–1956.

[8] Béal, M.-P. The method of poles : a coding method for constrained channels. *IEEE Trans. Inform. Theory 36*, 4 (1990), 763–772.

[9] Béal, M.-P. *Codage symbolique.* Masson, 1993.

[10] Béal, M.-P., and Perrin, D. Symbolic dynamics and finite automata. In *Handbook of Formal Languages*, G. Rozenberg and A. Salomaa, Eds., vol. 2. Springer-Verlag, 1997, ch. 10.

[11] Berstel, J., and Perrin, D. *Theory of codes.* Academic Press, 1985.

[12] Cormen, T., Leiserson, C., and Rivest, R. *Algorithms.* MIT Press, McGraw Hill, 1990.

[13] Franasazek, P. A., and Thomas, J. A. On the optimization of constrained channel codes. preprint.

[14] Franaszek, P. On synchronous variable length coding for discrete noiseless channels. *Inform. Control 1-J* (1969), 155–164.

[15] Franaszek, P. Run-length-limited variable length coding with error propagation limitation. U.S. Patent,3,689,899, 1972.

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

[17] Franaszek, P. Coding for constrained channel: a comparison of two approaches. *IBM J. Res. Dev. 33*, 6 (1989), 602–608.

[18] Heegard, C., Marcus, B., and Siegel, P. Variable length state splitting with applications to average runlength constrained (ARC) codes. *IEEE Trans. Inform. Theory 37*, 3 (1991), 759–777.

[19] Hollman, H. D. L. On the construction of bounded-delay encodable codes for constrained channels. *IEEE Trans. Inform. Theory IT-41*, 5 (1995), 1354–1378.

[20] Hollman, H. D. L. Bounded-delay-encodable, block-decodable codes for contrained systems. *IEEE Trans. Inform. Theory 42*, 6 (1996), 1957–1970.

[21] Karabed, R., and Marcus, B. Sliding block coding for input-restricted channels. *IEEE Trans. Inform. Theory 34*, 1 (1988), 2–26.

[22] Khayrallah, Z. A., and Neuhoff, D. Subshift models and finite-state modulation codes for input constrained channels: a tutorial. Tech. Rep. 90-9-1, Idel-EE, 1990.

[23] Lind, D., and Marcus, B. *An Introduction to Symbolic Dynamics and Coding.* Cambridge, 1995.

[24] Marcus, B. Factors and extensions of full shifts. *Monats. Math 88* (1979), 239–247.

[25] Marcus, B. Sofic systems and encoding data. *IEEE Trans. Inform. Theory IT-31*, 1 (1985), 366–377.

[26] Marcus, B., Siegel, P., and Wolf, K. Finite-state modulation codes for date storage. *IEEE Journal on Selected areas in Communications 10* (1992), 5–37.

[27] Marcus, B. H., Roth, R. M., and Siegel, P. H. Constrained systems and coding for recording channels. In *Handbook of Coding Theory*, V. Pless and W. Huffman, Eds., vol. II. North Holland, 1998, ch. 20, pp. 1635–1764.

[28] Perles, M., Rabin, M. O., and Shamir, E. The theory of definite automata. *IEEE Trans. Electr. Comp. EC-12* (1963), 233–243.

[29] Perrin, D. Finite automata. In *Handbook of Theoretical Computer Science*, J. V. Leeuven, Ed., vol. B. Elsevier, 1990, ch. 1.