Chapter 6
Capacity of Codes

Abstract. This chapter presents personal research on an application of the joint spectral radius to a problem in constrained coding: the computation of the capacity of codes submitted to forbidden differences constraints. We first present how the joint spectral radius appears to be the good tool to compute the capacity in this particular problem. We show how the quantity $\hat{\rho}_t$ provides bounds on the joint spectral radius that are tighter than in the general case. We show how the situation is even better in some particular situations. We then provide a polynomial time algorithm that decides if the capacity is positive. We introduce a closely related problem that we prove to be NP-hard. We then prove the existence of extremal norms for sets of matrices arising in this coding problem.

6.1 Introduction

In certain coding applications one is interested in binary codes whose elements avoid a set of forbidden patterns[1]. This problem is rather classical and has been widely studied in the past century [75]. In order to minimize the error probability of some particular magnetic-recording systems (see for instance [81]), a more complicated problem arises when it is desirable to find code words whose *differences* avoid forbidden patterns. We now describe this problem formally.

Let $\{0,1\}^t$ denote the set of words of length t over $\{0,1\}$ and let $u,v \in \{0,1\}^t$. The difference $u-v$ is a word of length t over $\{-1,0,+1\}$ (as a shorthand we shall use $\{-,0,+\}$ instead of $\{-1,0,+1\}$). The difference $u-v$ is obtained from u and v by symbol-by-symbol subtraction so that, for example, $0110-1011 = -+0-$. Consider now a finite set D of words over $\{-,0,+\}$; we think of D as a set of *forbidden difference patterns*. A set (or *code*) $C \subseteq \{0,1\}^t$ is said to *avoid* the set D if none of the differences of words in C contain a word from D as subword, that is, none of the differences $u-v$ with $u,v \in C$ can be written as $u-v = xdy$ for $d \in D$ and some (possibly empty) words x and y over $\{-,0,+\}$.

[1] The chapter presents research work that has been published in [11, 12].

R. Jungers: The Joint Spectral Radius: Theory and Applications, LNCIS 385, pp. 85–99.
springerlink.com

We are interested in the largest cardinality, which we denote by $\delta_t(D)$, of sets of words of length t whose differences avoid the forbidden patterns in D.

$$\delta_t(D) = \max_{W \subset \{0,1\}^t : W \text{ avoids } D} |W|.$$

If the set D is empty, then there are no forbidden patterns and $\delta_t(D) = 2^t$. We will see that when D is nonempty, $\delta_t(D)$ grows exponentially with the word length t and is asymptotically equal to $2^{\text{cap}(D)t}$ where the scalar $0 \leq \text{cap}(D) \leq 1$ is the *capacity* of the set D. The capacity is thus a measure of how constraining a set D is; the smaller the capacity, the more constraining the forbidden difference patterns are.

As an illustration consider the set of forbidden patterns $D = \{+-, ++\}$. Differences between two words in $C = \{u_1 0 u_2 0 \cdots 0 u_k : u_i \in \{0,1\}\}$ will have a "0" in any succession of two characters and will therefore not contain any of the forbidden patterns. From this it follows that $\delta_t \geq 2^{\lceil t/2 \rceil}$ and so $\text{cap}(D) \geq 1/2$. One can show that in fact $\text{cap}(D) = 1/2$. This follows from the next proposition combined with the simple observation that the capacity of the set $D = \{+-, ++\}$ is identical to the capacity of the set $D = \{+-, ++, -+, --\}$, that we denote $D = \{+,-\}^2$ as usual.

Proposition 6.1. *The capacity of the set $\{+,-\}^m$ is given by $(m-1)/m$.*

Proof. Let C_{km} be a code of length km avoiding D. In any given window of length m, the set of words appearing cannot contain both u and $\bar{u}$ (we use $\bar{u}$ to denote the word obtained by inverting the ones and the zeros in u). This implies that there are at most 2^{m-1} different words in any given window of size m. Let us now consider words in C_{km} as a concatenation of k words of length m. There are at most $2^{(m-1)k}$ words in C_{km} and so $\text{cap}(D) \leq (m-1)/m$.

Now consider the code

$$C_{km} = \{z_1 0 z_2 0 \cdots 0 z_k : z_i \in \{0,1\}^{m-1}\}. \tag{6.1}$$

This code satisfies the constraints, and the bound $(m-1)/m$ is reached.

The computation of the capacity is not always that easy. As an example it is proved in [82] that the capacity of $\{+++\}$ is given by $\log_2((1+(19+3\sqrt{33})^{1/3}+(19-3\sqrt{33})^{1/3})/3) = .8791\ldots$ and the same reference provides numerical bounds for the capacity of $\{0+-+\}$ for which no explicit expression is known.

The capacity of codes that avoid forbidden difference patterns was first introduced and studied by Moision, Orlitsky and Siegel. In [82], these authors provide explicit values for the capacity of particular sets of forbidden patterns and they prove that, in general, the capacity of a forbidden set D can be obtained as the logarithm of the joint spectral radius of a set of matrices that have binary entries. The size of the matrices constructed in [82] for computing the capacity is not polynomial in the size of the forbidden set D and so even the construction of the set of matrices is an operation that cannot be performed in polynomial time. Since moreover the computation of the joint spectral radius is NP-hard even if the matrices have binary entries, computing the capacity of codes seems at first sight to be a challenging task. However,

as pointed out in [82], the matrices that arise in the context of capacity computation have a particular structure and so the capacity could very well be computable in polynomial time.

In this chapter we first present this in details. We then provide several results ; all are related to the capacity computation and its complexity.

We first provide new bounds that relate the capacity of a set of forbidden patterns D with the values $\delta_t(D)$, the maximum size of a code of length t avoiding D. These bounds depend on parameters that express the number and positions of zeros in the patterns of D. These new bounds allow us to compute the capacity of any set to any given degree of accuracy by numerically evaluating $\delta_t(D)$ for some value of t. The approximation algorithm resulting from these bounds has exponential complexity but provides an a-priori guaranteed precision, and so the computational effort required to compute the capacity to a given degree of accuracy can be evaluated before the calculations are actually performed. As an example, it follows from the bounds we provide that the capacity of a set of forbidden patterns that does not contain any 0s can be computed with an accuracy of 90% by evaluating $\delta_t(D)$ for $t = 10$ (see Corollary 6.3 below).

In a subsequent section, we provide explicit necessary and sufficient conditions for a set to have positive capacity and we use this condition for producing a polynomial time algorithm that decides whether or not the capacity of a set is positive. These conditions are directly based on theoretical results presented in Chapter 3.

We then consider the situation where in addition to the forbidden symbols $-, 0$ and $+$ the forbidden patterns in D may also include the symbol $\pm$, where $\pm$ stands for both the symbols $+$ and $-$. We prove that in this case the problem of computing the capacity, or even determining if this capacity is positive, becomes NP-hard.

Finally, we show that sets of matrices constructed in order to compute the capacity always have an extremal norm.

These results allow us to better delineate the capacity computation problems that are polynomial time solvable from those that are not. We do however not provide an answer to the question, which was the original motivation for the research reported here, as to whether or not one can compute the capacity of sets of forbidden patterns over $\{-, 0, +\}$ in polynomial time. This interesting question that was already raised in [82], remains unsettled.

6.2 Capacity and Joint Spectral Radius

Let D be a set of forbidden patterns over the alphabet $\{-, 0, +\}$ and consider for any $t \geq 1$ the largest cardinality, denoted by $\delta_t(D)$, of sets of words of length t whose pairwise differences avoid the forbidden patterns in D. The capacity of D is defined by

$$\mathrm{cap}(D) = \lim_{t \to \infty} \frac{\log_2 \delta_t(D)}{t}. \tag{6.2}$$

The existence of this limit is a simple consequence of Fekete's Lemma (Lemma 1.1). We skip the formal proof, since it will be clear after the formulation of the problem with a joint spectral radius.

Moision et al. show in [82] how to represent codes submitted to a set of constraints D as products of matrices taken in a finite set $\Sigma(D)$. The idea of the proof is to make use of De Bruijn graphs. De Bruijn graphs were introduced in [38]; for an introduction, see for instance [75]. Let us construct the De Bruijn graph of binary words of length T equal to the lengths of the forbidden patterns. Edges in these graphs represent words of length T, and since some pairs of words cannot appear together, a subgraph of the De Bruijn graph is said *admissible* if it does not contain two edges that represent words of length T whose difference is forbidden. Figure 6.1 (a) represents a De Bruijn graph that is admissible for the forbidden pattern $D = \{++-\}$. An efficient way of drawing these graphs is to represent them as cascade graphs (see Chapter 3) as in Figure 6.1 (b). In order to construct longer codes, one just has to juxtapose admissible cascade graphs, such that each path from left to right represents an admissible word.

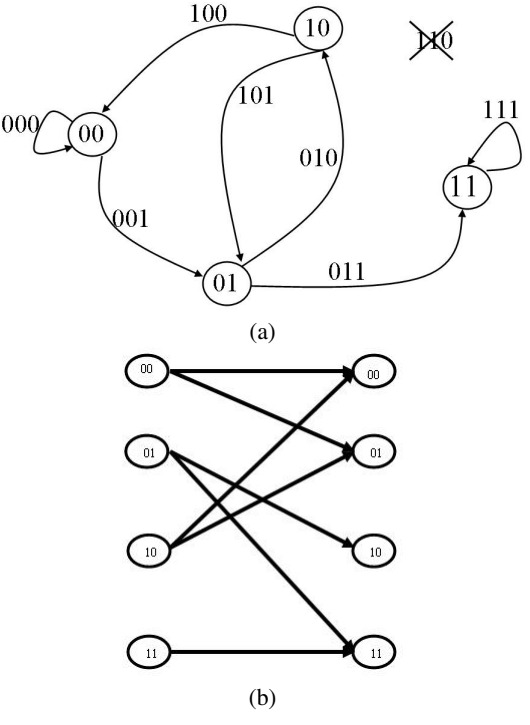

Fig. 6.1 An admissible De Bruijn graph for $D = \{++-\}$ (a), and the same graph under its cascade graph form (b)

In such a construction, the edges in the leftmost cascade graph represent words of length T, and each subsequent edge represents the addition of one letter to the word. A cascade graph for words of length 5 that are admissible for $D = \{++-\}$ is represented in Figure 6.2. Since we have a bijection between the paths of length t in an admissible cascade graph and the words in an admissible code of length

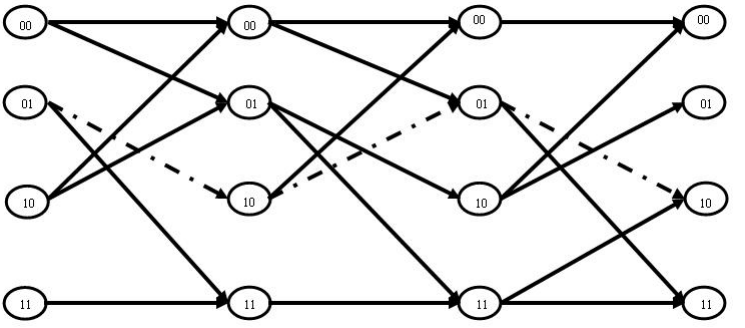

Fig. 6.2 An admissible cascade graph that represents a maximal set of admissible words of length 5 for $D = \{++-\}$. For example, the path on the top represents the word 00000 and the dashed path represents the word 01010. Such graphs are maximal in the sense that no word can be added to the corresponding code, but perhaps another choice of elementary cascade graphs would generate more paths

$T+t-1$, the maximal size of a code of length $T+t-1$ is given by the cascade graph of length T that maximizes the number of paths from left to right. We have seen in Chapter 3 how the joint spectral radius of binary matrices represents the asymptotics of the maximum number of paths in long cascade graphs. This reasoning leads to the following theorem:

Theorem 6.1. *Associated to any set D of forbidden patterns of length at most m, there exists a finite set $\Sigma(D)$ of binary matrices for which*

$$\delta_{m-1+t} = \hat{\rho}_t^t(\Sigma(D)) = \max\{\|A_1 \dots A_t\| : A_i \in \Sigma(D)\}. \tag{6.3}$$

In this expression, the matrix norm used is the sum of the absolute values of the matrix entries. The main result of this section is then a direct consequence of the definition of the joint spectral radius:

Corollary 6.1. *Let D be a set of forbidden patterns and $\Sigma(D)$ be the set of binary matrices constructed as described above, then*

$$cap(D) = \log_2(\rho(\Sigma(D))).$$

Example 6.1. *Let $D = \{++-\}$. The set $\Sigma(D)$ contains two matrices :*

$$A_0 = \begin{pmatrix} 1&1&0&0 \\ 0&0&1&1 \\ 1&1&0&0 \\ 0&0&0&1 \end{pmatrix}, \quad A_1 = \begin{pmatrix} 1&0&0&0 \\ 0&0&1&1 \\ 1&1&0&0 \\ 0&0&1&1 \end{pmatrix}.$$

One can check that the cascade graph in Figure 6.2 represents the product $A_0A_0A_1$ (the sum of the entries equals the number of paths).

The joint spectral radius of the set Σ is $\rho(\Sigma) = 1.75\ldots$ [82], and the product that ensures this value is $A_0A_0A_1A_1$, that is, $\rho(\Sigma) = \rho(A_0^2A_1^2)^{1/4}$, and $cap(D) = \log_2 1.75\ldots = 0.8113\ldots$.

Example 6.2. *Let $D = \{+++-\}$. The set $\Sigma(D)$ contains two matrices:*

$$A_0 = \begin{pmatrix} 1&1&0&0&0&0&0&0\\ 0&0&1&1&0&0&0&0\\ 0&0&0&0&1&1&0&0\\ 0&0&0&0&0&0&1&1\\ 1&1&0&0&0&0&0&0\\ 0&0&1&1&0&0&0&0\\ 0&0&0&0&1&1&0&0\\ 0&0&0&0&0&0&0&1 \end{pmatrix},$$

$$A_1 = \begin{pmatrix} 1&0&0&0&0&0&0&0\\ 0&0&1&1&0&0&0&0\\ 0&0&0&0&1&1&0&0\\ 0&0&0&0&0&0&1&1\\ 1&1&0&0&0&0&0&0\\ 0&0&1&1&0&0&0&0\\ 0&0&0&0&1&1&0&0\\ 0&0&0&0&0&0&1&1 \end{pmatrix}.$$

We will see that $cap(D) = 0.9005\ldots$ and that the product that ensures this value is A_0A_1 (see Example 6.5).

Let us comment here on the number and size of the matrices in $\Sigma(D)$; these issues are relevant for the questions raised hereafter: If the forbidden patterns in D have identical length m, then the number of matrices in $\Sigma(D)$ can be doubly exponential in m and all matrices in $\Sigma(D)$ have dimension $2^{m-1} \times 2^{m-1}$. If the forbidden patterns in D have different lengths, then one can construct a set D' whose forbidden patterns have equal length and for which $\mathrm{cap}(D) = \mathrm{cap}(D')$. Unfortunately, the number of patterns in D' can grow exponentially with the size of D so that the number of matrices in the set $\Sigma(D)$ is in fact even worse than in the former case. Capacity approximation algorithms based on the direct computation of the set $\Sigma(D)$ will therefore not be tractable even for small sets D.

6.3 Upper and Lower Bounds

In this section, we derive bounds that relate the capacity of a set D with $\delta_t(D)$. Consider some set D of forbidden patterns and denote by r_1 (respectively r_2) the maximal k for which 0^k is the prefix (respectively suffix) of some pattern in D: No pattern in D begins with more than r_1 zeros and no pattern in D ends with more than r_2 zeros. We also denote by r the maximal number of consecutive zeros in any

pattern in D; obviously, $r \geq \max(r_1, r_2)$. In the next theorem we provide upper and lower bounds on the capacity $\mathrm{cap}(D)$ in terms of $\delta_t(D)$.

Theorem 6.2. *For any $t \geq r_1 + r_2$ we have*

$$\frac{\log_2 \delta_t(D) - (r_1 + r_2)}{t + r + 1 - (r_1 + r_2)} \leq \mathrm{cap}(D) \leq \frac{\log_2 \delta_t(D)}{t}. \tag{6.4}$$

Proof. Let us first consider the upper bound. The following equation is straightforward, given any positive integers k, t, and any set of forbidden patterns D :

$$\delta_{kt} \leq \delta_t^k.$$

Indeed, considering any word of length kt as the concatenation of k subwords of length t, for each of these subwords we have at most δ_t possibilities. Taking the $\frac{1}{kt}$th power of both sides of this inequality and taking the limit $k \to \infty$, we obtain :

$$\rho(\Sigma) = 2^{\mathrm{cap(D)}} \leq \delta_t^{1/t}.$$

Now let us consider the lower bound. The optimal code of length t contains at least $\lceil 2^{-r_1-r_2}\delta_t(D) \rceil$ words that coincide in the first r_1 bits and in the last r_2 bits (because there are in total $2^{r_1+r_2}$ different words of length $r_1 + r_2$). Denote the set of strings of all these words from (r_1+1)st bit to $(t-r_2)$th bit by C'. This set contains at least $\lceil 2^{-r_1-r_2}\delta_t(D) \rceil$ different words of length $t - r_1 - r_2$. Then for any $l \geq 1$ the code

$$C = \left\{ u_1 0^{r+1} u_2 0^{r+1} \cdots 0^{r+1} u_l 0^{r+1},\ u_k \in C',\, k = 1, \ldots, l \right\} \tag{6.5}$$

avoids D. The cardinality of this code is at least $\lceil 2^{-r_1-r_2}\delta_t(D) \rceil^l$ and the length of its words is $T = l(t - r_1 - r_2 + r + 1)$. Therefore, for any l we have

$$\delta_T(D) \ \geq\ \lceil 2^{-r_1-r_2}\delta_t(D) \rceil^l.$$

Taking the power $1/T$ of both sides of this inequality, we get

$$\left[\delta_T(D)\right]^{1/T} \ \geq\ \lceil 2^{-r_1-r_2}\delta_t(D) \rceil^{1/(t-r_1-r_2+r+1)},$$

which as $T \to \infty$ yields

$$\rho \ \geq\ \lceil 2^{-r_1-r_2}\delta_t(D) \rceil^{1/(t-r_1-r_2+r+1)}.$$

Now after elementary simplifications we arrive at the lower bound on $\mathrm{cap}(D)$.

Both bounds in Theorem 6.2 are sharp in the sense that they are both attained for particular sets D. The upper bound is attained for the set $D = \emptyset$ and the lower bound is attained, for instance, for the set $D = \{0^{m-1}+\}$. Indeed, in this case $r = r_1 = m-1, r_2 = 0$ and $\mathrm{cap}(D) = 0$, because $\delta_t = 2^{m-1}$ for $t \geq m-1$. Here is a direct proof of this equality, drawn from [82]: Clearly, for all $t > m-1$, we can construct a code of size $\delta_t = 2^{m-1}$. It happens that for any given length t this size

is maximum. Otherwise, there must be two *different* words u and v whose prefixes of length k coincide. In order to avoid the forbidden pattern, the $k+1$-th symbols must also be equal, and so on. But then both words are equal, and we have reached a contradiction.

Corollary 6.2. *Let D be given and let r, r_1 and r_2 be defined as above. Then* $\frac{\log_2 \delta_t(D)}{t} - \frac{1}{t}\max(r_1+r_2, r+1) \leq \mathrm{cap}(D) \leq \frac{\log_2 \delta_t(D)}{t}$.

Proof. If $r_1 + r_2 \geq r+1$ this follows from Theorem 6.2 and from simple calculations. If $r_1 + r_2 < r+1$ simply use the fact that the capacity is always less than one in Theorem 6.2, and

$$\frac{\log_2 \delta_t(D)}{t} - (r_1+r_2) \leq (t+(r+1)-(r_1+r_2))\mathrm{cap}(D) \leq t\,\mathrm{cap}(D) + (r+1) - (r_1+r_2).$$

These bounds can be used to design an approximation algorithm that computes the capacity to any desired accuracy by evaluating δ_t for sufficiently large values of t. In contrast to previously known algorithms this algorithm has guaranteed computational cost: once the desired accuracy is given, the corresponding computational cost can easily be computed. As an illustration, consider the case of a set D for which $r_1 = r_2 = 2$ and $r = 4$. Then, by Corollary 6.2,

$$\frac{\log_2 \delta_t(D)}{t} - \frac{5}{t} \leq \mathrm{cap}(D) \leq \frac{\log_2 \delta_t(D)}{t} \tag{6.6}$$

and we can use $\log_2 \delta_t(D)/t$ as an estimate for $\mathrm{cap}(D)$ and choose a value of t for which (6.6) provides satisfactory accuracy bounds.

The easiest way of computing δ_t is to apply Equation (6.3), by evaluating the maximum-normed product of length $t-m+1$ of matrices taken in the set Σ. Moision et al. mention in [83] an improvement of this brute force method, similar to the ones proposed in Chapter 2: The main idea is to compute successively some sets of matrices $\bar{\Sigma}_l$, $l = 1, 2\ldots$, with $\bar{\Sigma}_1 = \Sigma$. These are sets of products of length l, obtained by computing iteratively all products of a matrix in $\bar{\Sigma}_{l-1}$ with a matrix in Σ, and then removing from the set $\bar{\Sigma}_l$ a matrix A if it is dominated by another matrix B in this set, that is, if each entry of A is less or equal than the corresponding entry of B. For more information about this algorithm, we refer the reader to [83]. We propose here an improvement of this method: given the set $\bar{\Sigma}_l$, one can directly compute a set $\bar{\Sigma}_{2l}$ by computing the set $\bar{\Sigma}_l^2$ and then removing from this set all matrices that are dominated. This small modification of the algorithm has dramatically improved the computational time for all the examples on which we have used it.

We may specialize the general bounds of Theorem 6.2 to sets of particular interest.

Corollary 6.3. *Let D be given and let r, r_1 and r_2 be defined as above. Then*

1. *If* $\mathrm{cap}(D) = 0$ *the size of any code avoiding D is bounded above by the constant* $2^{r_1+r_2}$.
2. *If the patterns in D contain no zero, then*

$$t\,\mathrm{cap}(D) \le \log_2 \delta_t(D) \le (t+1)\,\mathrm{cap}(D).$$

3. If none of the patterns in D starts nor ends with a zero, then

$$t\,\mathrm{cap}(D) \le \log_2 \delta_t(D) \le (t+r+1)\,\mathrm{cap}(D).$$

6.4 Positive Capacity Can Be Decided in Polynomial Time

As previously seen by a direct argument, the capacity of the set $\{0^{m-1}+\}$ is equal to zero. In this section we provide a systematic way of deciding when the capacity of a set is equal to zero. We first provide a simple positivity criterion that can be verified in finite time and then exploit this criterion for producing a positivity checking algorithm that runs in polynomial time. In the sequel we shall use the notation $-D$ to denote the set of elements that are the opposites to the elements of D, for example if $D = \{-+0, 0--\}$ then $-D = \{+-0, 0++\}$.

Theorem 6.3. *Let D be a set of forbidden patterns of lengths at most m. Then* $\mathrm{cap}(D) > 0$ *if and only if there exists a word on the alphabet $\{+,-,0\}$ that does not contain any word of $D \cup -D$ as subword and that has a prefix 0^m and a suffix $+0^{m-1}$.*

Proof. Let us first suppose $0^m \notin D$. The capacity is positive iff $\rho(\Sigma(D)) > 1$. We know (see Chapter 3) that for binary matrices this is equivalent to the fact that there is a product in Σ^* that has a diagonal entry larger than one. In turn, by construction of the set $\Sigma(D)$, this is equivalent to the existence of two words with the same $m-1$ first characters, and the same $m-1$ last characters, whose difference avoids the forbidden patterns. Now, this latter fact is possible iff there is a nontrivial sequence on $\{0,+,-\}$ of the shape $0^{m-1}d0^{m-1}$ that avoids $D \cup -D$.

Now in order to handle the case $0^m \in D$, which implies $\mathrm{cap}(D) = 0$, we add a zero at the beginning and by doing this, we do not change anything to the admissibility of this word, except that we remove the possibility $0^m \in D$.

Corollary 6.4. *If every word in D contains at least two nonzero symbols, then*

$$cap(D) > 0.$$

Proof. For any such set the word $d = 0^m + 0^{m-1}$ is admissible, and by Theorem 6.3 the capacity is positive.

Corollary 6.5. *If D consists of one forbidden pattern p of length m, then its capacity is zero if and only if p has at least $m-1$ consecutive zeros.*

Proof. If a pattern p is 0^m or $+0^{m-1}$, then obviously there are no admissible strings, and by Theorem 6.3 the capacity is zero. The same holds for -0^{m-1}, since this is the negation of $+0^{m-1}$ and for $0^{m-1}\pm$ because of the symmetry. In all the other cases the admissible string exists and so $\mathrm{cap(D)} > 0$. Indeed, if p has a unique nonzero character, then the word $d = 0^m + +0^{m-1}$ is admissible, if it has at least two nonzero characters, then the proof follows from Corollary 6.4.

We now prove the polynomial-time solvability of the problem of determining whether the capacity of a set D is positive. The proof is constructive and is based on the so-called *Aho-Corasick* automaton that checks whether a given text contains as a subsequence a pattern taken from a given set [1]. Let P be a set of patterns, that do not have to be of the same length. The transition graph of the Aho-Corasick automaton for the set P is defined as follows (see Figure 6.3 for an example). First, construct the *retrieval tree*, or trie, of the set P. The trie of P is the directed tree of which each vertex has a label representing a prefix of a pattern in P, and all prefixes are represented, including the patterns themselves. The label of the root of the tree is the empty string. Edges have a label too, which is a symbol of the used alphabet. There is an edge labeled with the symbol a from a vertex s to a vertex t if t is the concatenation sa.

In order to have an automaton, we complete the trie by adding edges so that for each vertex s, and each symbol a, there is an edge labeled a leaving s. This edge points to the vertex of the trie of which the label is the longest suffix of the concatenation sa. Note that this vertex can be the root (that is, the empty string) if no vertex in the trie is a suffix of sa. Finally, the accepting states of the automaton are the vertices whose labels are patterns of P. This automaton accepts words that contain a pattern in P and halts whenever this pattern is a suffix of the entered text.

If $0^k \in D$ or $+0^k \in D$, then, by Theorem 6.3, $\mathrm{cap}(D) = 0$. If this is not the case, we construct the graph of the automaton of Aho-Corasick for the set $P = D \cup (-D) \cup \{+0^{m-1}\}$. We then remove any vertex whose label is a complete pattern in P (i.e., a state reached when a suffix of the text entered is in the set P) except the vertex labeled $\{+0^{m-1}\}$. The size of the constructed graph is polynomial in the size and the number of the forbidden patterns. Moreover, since we have removed vertices corresponding to forbidden patterns, any path in the remaining graph is an admissible word. Let us now denote q_{0^m} the state reached after entering the word 0^m. This state is well defined since 0^m does not contain any forbidden pattern, and hence no state reached after entering any prefix of the string 0^m was removed from the primary automaton. We also denote $q_{+0^{m-1}}$ the state corresponding to the suffix $+0^{m-1}$ for the entered text (i.e. the accepting state corresponding to the pattern $+0^{m-1}$ in the Aho-Corasick automaton). Figure 6.3 presents the graph for $D = \{0+0\}$ that is obtained from the Aho-Corasick automaton of the set $P = \{0+0, 0-0, +00\}$.

We have the following criterion for zero-capacity:

Theorem 6.4. *The capacity of a set D is positive if and only if there is a path from q_{0^m} to $q_{+0^{m-1}}$ in the graph constructed above.*

Proof. If $\mathrm{cap(D)} > 0$, by Theorem 6.3, there exists a word d, beginning with m zeros, and ending with $+0^{m-1}$, that avoids $D \cup -D$. Hence, entering this word in the automaton, the finite state will be (well defined and will be) the vertex labeled $+0^{m-1}$, because the vertices removed from the original automaton of Aho-Corasick do not make any problem, since we do not reach the vertices labeled with forbidden patterns.

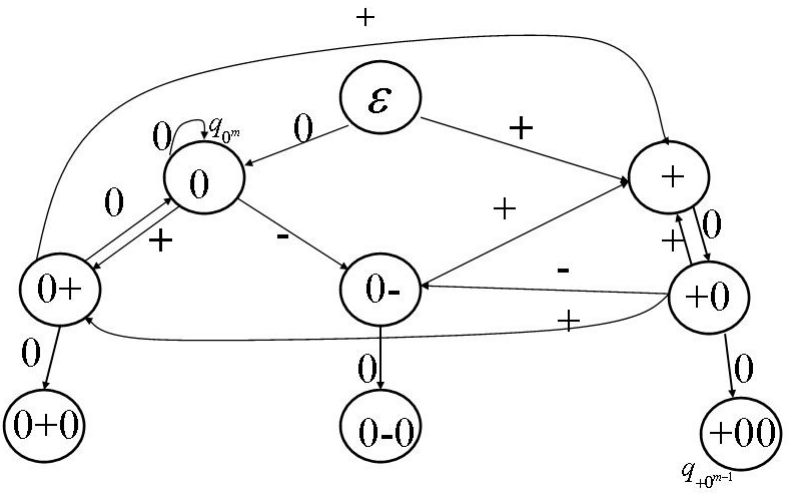

Fig. 6.3 The graph for $D = \{0+0\}$. We have constructed the Aho-Corasick automaton for $P = \{0+0, 0-0, +00\}$, and then removed the states $0+0$ and $0-0$ that are forbidden. The empty word is represented by ε. The path $0 \to 0- \to + \to +0 \to +00$ provides the admissible word $000-+00$

On the other hand, a path in the constructed graph represents an acceptable word, since it does not pass through any removed vertex, and hence no suffix of any prefix of this word will be in the forbidden set.

Moreover, a shortest path will give the shortest acceptable word, since the length of the path is equal to the length of the represented word.

Corollary 6.6. *The problem of determining whether or not the capacity of a given set of forbidden patterns is positive can be solved in polynomial time.*

Proof. Aho shows in [1] that the automaton is constructible in polynomial time. The determination of the state q_{0^m} and the computation of the shortest path are obviously polynomially feasible.

Corollary 6.7. *If for a set D of forbidden patterns there are admissible words, then the length of a shortest admissible word does not exceed $2M + 2m$, where m is the maximal length of all patterns in D and M is the sum of the lengths of each forbidden pattern.*

Proof. The number of vertices of the graph does not exceed $2M + m + 1$. Indeed, for each pattern of length l in $D \cup -D$ we add to the automaton at most l states, since there are no more than l prefixes of this pattern. We still add the pattern $\{+0^{m-1}\}$ (maximum m new states), and the root. If there is a path connecting two given vertices, this path can be chosen so that its length (in terms of number of vertices) will not exceed the total number of vertices (if it does not pass through the same vertex twice). Every edge of this path adds one bit to the admissible string. The initial length of the string is m (we start from 0^m), therefore the total length of the admissible word is at most $2M + 2m$.

Proposition 6.2. *If the capacity is positive, then* $\mathrm{cap}(D) > 1/(2M+m)$*, where* m *is the maximal length of all patterns in* D *and* M *is the sum of the lengths of each forbidden pattern.*

Proof. If $\mathrm{cap(D)} > 0$, then there is an admissible string of length $t \leq 2M+2m$ (Corollary 6.7). Consider a code as given by Equation (6.5). Its size is 2^l and the length of its words is at most

$$T_l \;=\; l(2M+2m-m) = l\big(2M+m\big).$$

Therefore

$$\begin{aligned}\mathrm{cap(D)} \;&=\; \lim_{\mathrm{l}\to\infty} \frac{\log_2 \delta_{\mathrm{T_l}}}{\mathrm{T_l}} \\ &\geq\; \lim_{l\to\infty} \frac{\log_2 2^l}{l\big(2M+m\big)} \;=\; \frac{1}{2M+m}.\end{aligned}$$

6.5 Positive Capacity Is NP-Hard for Extended Sets

We now consider the situation where forbidden patterns are allowed to contain the $\pm$ symbol. The symbol $\pm$ is to be understood in the following sense: whenever it occurs in a forbidden pattern, both the occurrences of $+$ and of $-$ are forbidden at that particular location. So, for example, avoiding the forbidden set $\{0\pm+\pm\}$ is equivalent to avoiding the set $\{0++{+},0++{-},0-+{+},0-+{-}\}$. All results obtained for forbidden patterns over $\{-,0,+\}$ have therefore their natural counterparts in the situation where the forbidden patterns are defined over the alphabet $\{-,0,+,\pm\}$. In particular, the results of Section 6.3 do transfer *verbatim* and the bounds derived in Theorem 6.2 are valid exactly as stated there. However, the symbol $\pm$ allows us to compress the number of forbidden patterns so that the new instance is exponentially smaller. Thus, the polynomial time algorithm described above for normal sets could well not be polynomial in the size of the compressed instance. We now prove that unless $P = NP$, there is no polynomial time algorithm to decide zero capacity when the symbol $\pm$ is allowed.

Theorem 6.5. *The problem of determining if the capacity of a set of forbidden patterns over* $\{0,+,-,\pm\}$ *is equal to zero is NP-hard.*

Proof. The proof proceeds by reduction from the Not-All-Equal 3SAT problem that is known to be NP-complete (see [44]). In the Not-All-Equal 3SAT problem, we are given m binary variables $x_1,\ldots,x_m$ and t clauses that each contain three literals (a literal can be a variable or its negation), and we search a truth assignment for the variables such that each clause has at least one true literal and one false literal.

Suppose that we are given a set of clauses. We construct a set of forbidden patterns D such that $\mathrm{cap}(D) > 0$ if and only if the instance of Not-All-Equal 3SAT has a solution. The first part of D is given by:

$$\{(0\pm 0),(0\pm\pm 0),\ldots,(0\pm^{m-1}0)\}. \tag{6.7}$$

Words over $\{-,0,+\}$ that avoid these patterns are exactly those words for which any two consecutive zeros are either adjacent or have at least m symbols on $\{+,-\}$ between them. We use these m symbols as a way of encoding possible truth assignments for the variables (the first one is "+" if $x_1 = 1$, etc...).

We then add to D two patterns for every clause: they will force a sequence of m nonzero symbols to encode a satisfying assignement for the instance of Not-All-Equal 3SAT. These patterns are of length m and are entirely composed of symbols $\pm$, except for the positions corresponding to the three variables of the clause, which we set to $+$ if the clause contains the variable itself, or to $-$ if the clause contains the negation of the variable. We also add the opposite of this pattern; this last pattern is not necessary for the proof but preserves the symmetry and simplifies the construction.

For example, if the instance of Not-All-Equal 3SAT consists of the two clauses $(x_1,\bar{x}_3,x_4)$ and $(\bar{x}_2,x_4,x_5)$, the corresponding set D will be $D = \{(0\pm 0),(0\pm\pm 0),(0\pm\pm\pm 0),(0\pm\pm\pm\pm 0),(+\pm-+\pm),(-\pm+-\pm),(\pm-\pm++),(\pm+\pm--)\}$.

Such a set D has always a length polynomial in the number of clauses and the number of variables.

We now prove that there is a solution to the instance of Not-All-Equal 3SAT if and only if $\mathrm{cap}(D) > 0$. First, suppose that there exists a satisfying truth assignment for x and denote it by $(\omega_1,\ldots,\omega_m) \in \{0,1\}^m$. Associated to any $k \geq 1$ we construct a code of length $k(m+1)$ containing 2^k words as follows:

$$C_{k(m+1)} = \{0\omega 0\omega 0\omega 0\cdots 0\omega 0\omega, 0\omega 0\omega 0\omega 0\cdots 0\omega 0\bar{\omega},$$

$$0\omega 0\omega 0\omega 0\cdots 0\bar{\omega}0\omega,\ldots,0\bar{\omega}0\bar{\omega}0\bar{\omega}0\cdots 0\bar{\omega}0\bar{\omega}\},$$

where $\omega = \omega_1\cdots\omega_m$.

Any difference between two words in this code is a word of the form

$$0z_1 0z_2 0\cdots 0z_k,$$

where for every $1 \leq i \leq k$, z_i is either a sequence of m 0's or a word of length m over $\{-,+\}$. Because ω satisfies the instance of Not-All-Equal 3SAT, these words avoid the set D constructed above. Moreover, the cardinality of $C_{k(m+1)}$ is 2^k and hence

$$\mathrm{cap}(D) \geq \lim_{k\to\infty} \log_2 2^{\frac{k}{k(m+1)}} = \frac{1}{m+1} > 0. \tag{6.8}$$

For the converse implication, assume now that $\mathrm{cap}(D) > 0$. The capacity is positive, and so one can find two words whose differences contain a 0 and a $+$. But then since this difference must avoid the first part of the forbidden pattern, for a code C large enough, there must exist two words in the code whose difference contains a word over $\{-,+\}$ of length m. But this sequence avoids also the second part of D, and thus it represents an acceptable solution to our instance of Not-All-Equal 3SAT.

Note that a similar proof can be given if we replace the symbol "$\pm$" in the statement of the theorem by a symbol that represents either $+$, $-$, or 0.

6.6 Extremal Norms and Computing the Capacity

As we have seen in previous chapters, the existence of an extremal norm can simplify many problems related to the joint spectral radius: it allows for instance to apply the geometrical algorithm exposed in Section 2.3. Recall that an extremal norm is a norm $||\cdot||$ such that

$$\max_{A\in\Sigma} ||A|| = \rho(\Sigma).$$

It turns out that in the case of capacity computation, the matrices do in fact always possess an extremal norm:

Theorem 6.6. *For any set D of forbidden patterns the set $\Sigma(D)$ possesses an extremal norm.*

Proof. Corollary 6.2 implies that $\Sigma(D)$ is not defective. To see this, replace $\mathrm{cap}(D)$ by $\log_2 \rho$ in Corollary 6.2 and recall that δ_t is, by definition of the set Σ, the maximal norm of products of length $t-(m-1)$ of matrices taken in Σ. We have seen in Section 2.1 that the nondefectiveness implies the existence of an extremal norm.

The existence of an extremal norm for a set of matrices makes it possible to apply the geometric algorithm described in Section 2.3 for computing the capacity with a given relative accuracy.

The complexity of this algorithm is exponential with respect to m, as the one proposed in Section 6.3 that approximates the capacity by successive estimations of δ_t. The advantages of one algorithm over the other appear in numerical computation of the capacity. Moreover, in many cases the approximation of invariant bodies by polytopes can lead to the exact value of the joint spectral radius, as mentioned in Section 2.3. Let us illustrate this method by computing the exact values of the capacity for several codes. In Examples 6.3 and 6.4 we find the values of capacities that were approximated in [82]. Example 6.5 deals with a code with $m = 4$.

Example 6.3. *$cap(\{0++\}) = \log_2 \rho(A_0) = \log_2\left(\frac{\sqrt{5}+1}{2}\right) = 0.69424191\ldots$. The eigenvector is $v = (2, \sqrt{5}-1, 2, \sqrt{5}-1)^T$. The algorithm terminates after five steps, the polytope $P = P_5$ has 32 vertices.*

Example 6.4. *$cap(\{0+-\}) = \log_2 \rho(A_0) = \log_2\left(\frac{\sqrt{5}+1}{2}\right)$. The algorithm terminates after four steps, $v = (2, \sqrt{5}-1, \sqrt{5}-1, 2)^T$, $P = P_4$, the polytope has 40 vertices.*

Example 6.5. *$cap(\{+++-\}) = \log_2\left(\frac{\sqrt{3+2\sqrt{5}}+1}{2}\right) = \log_2 \sqrt{\rho(A_0A_1)} = 0.90\ldots$. The algorithm terminates after eleven steps, the polytope $P = P_{11}$ has 528 vertices.*

6.7 Conclusion

One way to compute the capacity of a set of forbidden patterns is to compute the joint spectral radius of a set of matrices. In practice, this leads to a number of difficulties: first, the size of the matrices is exponential in the size of the set of forbidden patterns. Second, their number can also be exponential in the size of the instance. Third, the joint spectral radius is in general NP-hard to compute.

We have shown here that, in spite of these discouraging results, the simpler problem of checking the positivity of the capacity of a set defined on $\{+,-,0\}$ is polynomially decidable. However the same problem becomes NP-hard when defined over the alphabet $\{+,-,0,\pm\}$, so that we see a threshold between polynomial time and exponential time feasibility. We have also provided bounds that allow faster computation of the capacity. Finally we have proved the existence of extremal norms for the sets of matrices arising in the capacity computation, which is the only "good news" that we see concerning the possible feasibility of the capacity computation. To the best of our knowledge the problem remains open for the moment:

Open question 9. *Is the capacity computation/approximation NP-hard?*

For instance, one has to keep in mind that the approach that consists in computing a joint spectral radius cannot lead to a polynomial algorithm because of the exponential size of the sets of matrices. Nevertheless, it is conjectured in [11] that the sets of matrices with binary entries, and, in particular, those constructed in order to compute a capacity do always possess the finiteness property:

Open question 10. *Do matrices that arise in the context of capacity computation satisfy the finiteness property?*

Numerical results in [82], [57], and in this chapter seem to support this conjecture, and moreover the length of the period seems to be very short: it seems to be of the order of the size of the forbidden patterns, which would be surprising, because this length would be logarithmic in the size of the matrices.

We end this chapter by mentionning another question that has not been solved yet. We have seen that if the capacity is positive, one is able to exhibit an admissible word of the shape $0^m d 0^{m-1}$. This word has moreover a size which is polynomial in the size of D since it is represented by a path in the auxiliary graph constructed from the Aho-Corasick automaton. Now if we allow the use of "$\pm$" characters, since the problem can be translated in a classical instance D' with characters in $\{0,+,-\}$, a positive capacity also implies the existence of a certificate of the shape $0^m d 0^{m-1}$. But what about the length of this word? Since this length is only polynomial in the new instance D', we cannot conclude that there exists a certificate whose size is polynomial in the former instance. If this was the case, we would have that the problem with "$\pm$" characters would be in NP. This motivates our last open question:

Open question 11. *Is the problem of determining if the capacity of a set of forbidden patterns D over $\{0,+,-,\pm\}$ is equal to zero in NP? Is there, for any set $D \in \{0,+,-,\pm\}^*$, an admissible word of the shape $0^m d 0^{m-1}$ whose length is polynomial in the size of D?*