UCLouvain

École polytechnique de Louvain

Synchronizing Automata, Primitive Matrix Sets, and the Černý Conjecture

Authors: **Umer Azfar, Ludovic Charlier**
Supervisors: **Costanza Catalano, Raphaël Jungers**
Readers: **François Gonze, Raphaël Jungers, Jean-Pierre Tignol**
Academic year 2018–2019
Master [120] in Mathematical Engineering

Contents

Acknowledgements

We would like to thank first of all Costanza Catalano, for her numerous explanations/clarifications and her helpful suggestions, but above all for her patient reading of our write-ups and the ensuing (much-needed) comments and criticism she provided us throughout this year. We are also indebted to Raphaël Jungers for not only introducing us to this exciting topic in the first place, but also for his guidance. Together, our two supervisors have made this thesis a great learning opportunity for the both of us.

We are also grateful to our readers Jean-Pierre Tignol and François Gonze for taking the time to read our thesis, and we would like to apologize in advance for any eventual lack of clarity/mistakes they may encounter in this manuscript.

Introduction

> *"No one has yet discovered any warlike purpose to be served by the theory of numbers or relativity, and it seems unlikely that anyone will do so for many years.*
>
> G.H. Hardy – *A Mathematician's Apology* (1941)

In this master's thesis, we study the Černý conjecture and some related problems. We would do well to first consider here what value there is in this pursuit, aside from the purely personal value one derives from the satisfaction of one's intellectual curiosity.

In the case of most master theses in applied mathematics, one starts with a concrete, practical problem, the resolution of which requires the application of non-trivial mathematics, and finally the value of the work likes quite simply in the solution provided. In our case however, the situation is quite different. We consider the Černý conjecture to be inherently worthy of study *for its own sake*, as prosecuting this research is likely to provide important insights into the foundational notion of *synchronization*. Applications will then follow as they always do[1], and some of them are detailed in the following chapter. Indeed in the course of our research we shall develop new algorithms for solving synchronization problems (amongst others) that seem to improve on at least one of the best known algorithms in the case studied. Algorithms of the sort are of practical importance in robotics and communications systems.

Although the theory of finite state automata goes back to the dawn of the computing era, and Černý's elegant conjecture on synchronizing automata was first proposed in 1964, the conjecture has remained open to this day, despite considerable efforts on the part of the mathematical community. What is most surprising about this situation is that the conjecture is extremely simple, easily

[1]The Hardy quote in the epigraph predates the Manhattan Project – which would have been impossible were it not for the prior development of relativity theory – by only a few years. The question of whether this particular practical application was of any value is best left aside.

understandable by the average layman, yet mathematicians so far have failed to crack the problem. It is, in that way reminiscent of some the most enticing problems in mathematics, like the Goldbach Conjecture and the Collatz Conjecture.

By contrast, the theory of primitive sets of matrices is quite recent, indeed most of the results of this theory that we shall be exploring are not more than a few years old. In light of some of these results, it seems clear that the theory of primitive matrix sets is quite closely related to the theory of synchronizing automata. We shall therefore study problems quite similar to Černý's in this framework to see if this approach yields fruitful results.

Approaching the problem from this angle allows us to utilize the tools of automata theory, linear algebra, graph theory and combinatorics to explore a challenging research problem in applied mathematics/theoretical computer science.

The general structure of our thesis is as follows.

- Chapter 1 gives a survey of the literature concerning the theory of synchronizing automata and primitive sets of matrices. It also defines some of the notions/problems we shall explore in greater detail in the following sections, in particular the k-RT (k-Rendezvous-Time) of a primitive matrix set.
- Chapter 2 introduces the notations we shall employ in our presentation, and some simple lemmas that shall be used extensively throughout this manuscript.
- Chapter 3 consists of a theoretical analysis of the k-RT of NZ primitive sets. Several functions providing upper bounds on the k-RT are derived and compared in this section. Surprisingly, we show that the k-RT of an NZ primitive set is bounded by a linear function of the matrix size n for large n.
- Chapter 4 presents the results of our analysis of an algorithm that was recently proposed for the random generation of slowly synchronizing automata/NZ primitive sets with large exponents. We present arguments that the automata/matrix sets generated by this algorithm will all belong to a certain restricted class of automata/matrix sets with high probability.
- Chapter 5 introduces a heuristic algorithm that we developed for approximating the k-RT and the exponent of NZ primitive sets, notably those produced by the algorithm studied in the preceding chapter. We compare the performance of our new heuristic to a generalization of the well-known Eppstein heuristic.

Chapter 1

Literature review

1.1 Synchronizing automata and the Černý conjecture

The theory of automata grew naturally out of the theory of computation during the early 20th century when automata were defined as mathematical models of computing machines in order to facilitate a more abstract analysis of their properties. They were initially defined in more than one way. Some of the pioneering seminal works in this domain were the classic papers by Pitts and McCulloch [27], and Kleene [23] on nerve-nets, wherein they study the neurophysiological basis of mental computation and related mathematical problems[1]. Building on their work, Rabin and Scott introduced their definition of finite automata in [30] as a more realistic model of computing machines than Turing machines. The latter were introduced by Turing in his classic paper [45].

Since then, the theory of automata has found diverse applications not only in the study of neurophysiological processes and computability theory, but also among various other fields of mathematics, science and engineering such as formal linguistics [12], robotics [29] and logic [10].

For the scope of this thesis, we shall restrict our attention mainly to complete, deterministic, finite state automata which we introduce here below.

Definition 1. *A (complete, deterministic, finite state) automaton is a triplet* $\mathcal{A} = \langle Q, \Sigma, \delta \rangle$ *where* $Q = \{q_1, \ldots, q_n\}$ *is a finite set of states,* $\Sigma = \{a_1, \ldots, a_m\}$ *is a finite set of input symbols (the letters of the automaton) and* $\delta : Q \times \Sigma \to Q$ *is*

[1] Also, in the case of Pitts and McCulloch, distinctly philosophical problems.

the transition function. Let $i_1, i_2, ..., i_l \in \{1, ..., m\}$*, then* $w = a_{i_1}a_{i_2}...a_{i_l}$ *is called a word (of length* l*) and we define* $\delta(q, w) = \delta(\delta(q, a_{i_1}a_{i_2}...a_{i_{l-1}}), a_{i_l})$*.*

Remark 1. *The automaton* $\mathcal{A}$ *can be equivalently represented by the set of binary*[2]*, row-stochastic*[3] *matrices* $\{A_1, \ldots, A_m\}$ *where, for all* $i = 1, \ldots, m$ *and* $l, k = 1, \ldots, n$*,* $(A_i)_{lk} = 1$ *if* $\delta(q_l, a_i) = q_k$*,* $(A_i)_{lk} = 0$ *otherwise. This matrix representation will be used to show the connection between synchronizing automata and primitive matrix sets (see section 1.4).*

Synchronizing automata were explicitly defined for the first time in 1964, in Černý's famous paper [46] (although they were called *directable* automata at the time). For an automaton representing a certain system, its synchronizability captures the idea of the possibility of driving the system to a given state, without knowing its initial state. This problem arises in many applied domains: for example the automatic orientation of mechanical parts for manufacturing and assembly ([29]), the synchronization of molecular automata in biocomputing applications (see [6, 49]) or resilient data compression (see [38]). Some of the notions and results were re-invented independently a few times, for example in [26, 43]. For an overview of the history of synchronizing automata, see [49]. The formal definition of *synchronizing* automata is as follows:

Definition 2. *An automaton is synchronizing if and only if it admits a word* w*, called a synchronizing word (or a reset word), and a state* q *such that* $\delta(q', w) = q$ *for any state* $q' \in Q$*. The reset threshold of an automaton* $\mathcal{A}$*, denoted by* $\mathrm{rt}(\mathcal{A})$*, is the length of the shortest reset word of the automaton.*

We illustrate the notion of synchronization by showing its applicability to a recent development in computing: biocomputing. In recent times, the possibility has emerged of solving difficult computational problems that resist conventional methods by exploiting molecular computers. The latter are notable because of their excellent energy efficiency, high information density and parallelism, and have been put to use to solve NP-Hard problems with large number of variables, see for example [9] wherein the authors present their method of solving a 3-SAT problem in 20 variables. In a recent paper, [6], Benenson et al present a method of using DNA molecules as input data/fuel to molecular computers to solve computational problems. The molecular computer that they developed can be modeled by a unidirectional read-only Turing machine which in turn can be represented by a finite state automaton, as explained below.

[2]A matrix is binary if each entry is either zero or 1.

[3]A matrix is row stochastic if the entries in each row sum to 1.

A finite automaton can be seen as a unidirectional read-only Turing machine whose input is a finite string of symbols. The Turing machine starts with a starting input symbol and an initial state. Initially, we first read the leftmost input symbol, in a default initial state. Each transition moves one symbol to the right, possibly changing its internal state. The Turing machine needs transition rules: given a current state and symbol, the transition rule specifies the next state. After the last input symbol is read, the final state is the output of the system, some of these states being said to be *accepting*. We say that the automaton representing such a machine accepts an input if, starting from the initial state and following the transitions dictated by the input symbols it ends in an accepting final state. These automata can then be "programmed" by choosing the transition rules and accepting states adequately.

The Figure 1.1 shows an example of a automaton with an initial state, and one accepted state. For example, we can easily verify that the input sequence *abba* is accepted, since, starting from the initial state 1, if we apply the input sequence, we end up at an accepted state (in this case, 1).

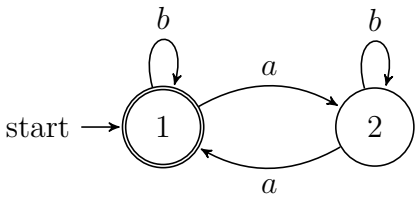


Figure 1.1: Finite automaton with an initial state, and one accepted state. The transition rules are represented by the labeled arrows. A double circle represents an accepting state.

In their experiment, Benenson et al. produced a solution containing 3×10^{12} identical molecular automata per micro litre which work in parallel on different inputs. Once a computation has been completed, and these automata are in their various output states, if we want to perform a new computation, we need to *synchronize* them, that is to say bring them all to the initial state, by adding to the solution sufficiently many copies of a DNA molecule whose nucleotide sequence encodes a reset word.

Although the problem of determining whether an automaton is synchronizing can be solved in polynomial time (see e.g. [49]), computing its reset threshold is an NP-hard problem [13]. However, this does not exclude the possibility of finding a theoretical upper bound on the reset threshold of an arbitrary synchronizing

automaton. In his paper [46], Černý presented a family of n-state automata for $n > 2$ with a reset threshold of $(n-1)^2$, and stated his famous conjecture:

Conjecture 1 (The Černý Conjecture [46])**.** *The reset threshold of any synchronizing automaton on n states is at most $(n-1)^2$.*

Clearly, if the conjecture is true then this bound can not be further improved given the existence of Černý's family of automata. The conjecture has remained open to this date, though it has been shown to be true for certain classes of automata (see [21, 41, 48]). So far the best known asymptotic upper bound on the reset threshold of an arbitrary synchronizing automaton is Szykula's cubic bound of $\big(85059n^3 + 90024n^2 + 196504n - 10648\big)/511104$, where n is the number of states of the automaton. This bound slightly improves previous cubic bounds [32, 14].

Informally speaking, the reset threshold of an n-state synchronizing automaton gives the length of the shortest word that maps all n states to the same output state. It seems natural to investigate the more general problem of bounding the k-rendezvous time (k-RT) of a synchronizing automaton $\mathcal{A}$ (written $rt_k(\mathcal{A})$) which is defined as the length of the shortest word that maps k distinct states of the automaton to the same output state. So far the only known bound on the k-RT (outside of the trivial[4] results for $k = 1$ and $k = 2$) is the quadratic bound on the 3-RT found by Jungers and Gonze in [17].

Recent years have seen a growth of interest in families of automata with reset thresholds that are quadratic in the number of states of the automaton, called *extremal* or *slowly synchronizing* automata. It is hoped that such families may help improve our understanding of the behaviour of possible counter-examples to the Černý conjecture. However such families seem to be hard to find, see [4, 11, 47, 20] for some examples.

1.2 Algorithmic aspects of synchronization

Although computing the reset threshold of a synchronizing automaton is an NP-hard problem, an algorithm was developed by Kisielewicz et al. in [22] that computes the

[4]Clearly the case $k = 1$ is trivial since in a complete automaton we can always find some state that is mapped to some other state, hence 1-RT = 1. The case $k = 2$ is also trivial since any automaton which never maps two states to the same state may only have letters that act as permutations on the set of states, and any such automaton can not be synchronizing, since the composition of permutations is a permutation. Hence for any synchronizing automaton, 2-RT = 1.

reset threshold in polynomial time for quite a few families of slowly synchronizing automata. Of course the worst-case behaviour is exponential, but a number of heuristics implemented as part of the algorithm make the algorithm work quite well in general.

One of the heuristics used is a reduction of the search space using upper bounds provided by the Eppstein heuristic on the words used to make up the reset word. The Eppstein heuristic was presented in [13] as a polynomial-time algorithm for computing short reset words of synchronizing automata. It was shown to produce reset words that are at most cubic in the number of states of the automaton, in cubic execution time (see [13]). It was also shown (see [3]) that the reset word produced by the Eppstein heuristic for an n-state automaton $\mathcal{A}$ is of length at most $(n-1) \cdot \mathrm{rt}(\mathcal{A})$. We shall make use of the Eppstein algorithm in later sections, as it seems to be the most widely used heuristic for finding reset sequences of synchronizing automata. Indeed, quite a few of the other algorithms are actually based on the Eppstein heuristic.

There exist other algorithms for finding (short) reset words of a synchronizing automaton. Some examples include the following.

- Trahtman's `Cycles` algorithm, which works similarly to Eppstein's algorithm, see [44].
- Roman's `SynchroP` algorithms which is a (computationally much slower) attempt at improving the Eppstein heuristic, see [36].
- The `FastSynchro` algorithm proposed by Kudłacic et al as an improvement of Roman's `SynchroP` algorithm, as the name suggests it is faster than the original, see [25].
- Kowalski and Szykula's `CutOff-IBFS` algorithm that builds synchronizing words in reverse, starting from the end and finishing with the first letter, see [24].
- Roman's evolutionary algorithm based on the simple genetic algorithm model, see [35].

An algorithm was also proposed in [44] for generating automata with large reset thresholds. A new randomized algorithm for generating automata with large reset thresholds was also recently proposed in [11]. We shall be analyzing this algorithm in greater detail in Chapter 4.

1.3 Primitive matrix sets and their exponents

The theory of primitive matrix sets was developed quite recently (see [34, 1, 2, 7] for example) as a natural generalization of the Perron-Frobenius theory of nonnegative matrices[5] which was developed by Perron for positive matrices[6] in [31] and generalized by Frobenius to nonnegative matrices in [15]. The Perron-Frobenius theory has enjoyed a very widespread and diverse applicability to fields like economics [8], probability theory (specifically the theory of Markov chains [28]) and population modeling (see [28]) among others.

A fundamental result in the theory of nonnegative matrices relates the notions of primitivity, irreducibility and aperiodicity.

Definition 3. *A matrix A is irreducible iff for every pair of indices i and j, there is a natural number k such that $(A^k)_{ij} > 0$.*

Definition 4. *A matrix A is primitive iff it is nonnegative and there is some m such that $A^m > 0$. The smallest m satisfying this condition is called the exponent of A, denoted by $\exp(A)$.*

Definition 5. *Let A be a nonnegative matrix of dimension $n \times n$. The A-period of an index i, denoted by $\tau_A(i)$, is the greatest common denominator of all natural numbers k such that $(A^k)_{ii} > 0$. A is aperiodic iff for all indices $i \in \{1, \ldots, n\}$, $\tau_A(i) = 1$.*

Theorem 1 (See Theorem 1.4 in [40], p. 21)**.** *A nonnegative matrix is primitive iff it is irreducible and aperiodic.*

One of the many interesting results that came out of the Perron-Frobenius theory is known as the Wielandt bound. This result gives a (tight) quadratic bound on the exponent of a primitive matrix:

Theorem 2 (Wielandt's bound [37])**.** *If A is a primitive $n \times n$ matrix, then $A^q > 0$, where $q = n^2 - 2n + 2$. Further, there is such a matrix A for which A^{q-1} is not positive. In other words, for any primitive matrix A, we have that $\exp(A) \leqslant n^2 - 2n + 2$.*

The bound on the exponent of a primitive matrix raises a more general question: what is the length of the shortest product of matrices of a given set that gives a positive matrix? This is the problem of bounding the exponent of a primitive matrix set (the Wielandt bound solves this problem for the special case of sets

[5] A nonnegative matrix is a matrix such that every element of the matrix is nonnegative.
[6] A positive matrix is a matrix such that every element of the matrix is positive.

with only one element).

Though the concept of primitive matrix sets had already made appearances in [19, 33, 18, 39], it was first explicitly defined by Protasov and Voynov in [34] in the following way:

Definition 6. *A set of nonnegative matrices $\mathcal{M} = \{M_1, \ldots, M_m\}$ is called primitive if there exist some indices $i_1, \ldots, i_r \in \{1, \ldots, m\}$ such that the product $M_{i_1} \cdots M_{i_r}$ is entrywise positive. A product of this kind is called a positive product and the length of the shortest positive product of a primitive set $\mathcal{M}$ is called its exponent and it is denoted by $\exp(\mathcal{M})$. The set of all finite products of matrices of $\mathcal{M}$ is written $\mathcal{M}^*$, and the set of all products of at most d matrices of $\mathcal{M}$ is written $\mathcal{M}^d$.*

Protasov and Voynov derived a necessary and sufficient condition for the primitivity of a matrix set belonging to the class of irreducible, NZ matrix sets. Before stating their theorem we need to define the following notions:

Definition 7. *An NZ matrix is a nonnegative matrix with no zero rows or columns, an NZ matrix set is a set of NZ matrices.*

Definition 8. *A matrix set $\mathcal{M}$ is irreducible if and only if for every pair of indices i and j, there is a product $M \in \mathcal{M}^*$ such that $M_{ij} > 0$.*

Definition 9. *Let $\Omega = \{\Omega_1, \ldots, \Omega_q\}$ be a partition of $\{1, \ldots, n\}$ and let S_q be the symmetric group[7] of order q. We say that an $n \times n$ matrix A acts as permutation $\sigma \in S_q$ on Ω iff, for all $k \in \{1, \ldots, q\}, i \in \Omega_k$ and any index $j \in \{1, \ldots, n\}$ we have that $j \notin \Omega_{\sigma(k)} \implies A_{ij} = 0$. We say that A acts as a permutation on Ω iff there exists a permutation $\sigma \in S_q$ such that A acts as permutation σ over Ω.*

In other words, an $n \times n$ matrix A acts as a permutation on a partition of the set $\{1, \ldots, n\}$ iff, when transformed by simultaneous row and column permutations into a block form corresponding to the partition, each block row and column contains exactly one nonzero block.

For example, consider the following matrix:

$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \end{array}$$

[7]The symmetric group of order q is the group of all permutations on q elements under the composition operation.

It can be shown that it acts as a permutation on the partition $P = \{\{1,3\},\{2,4\}\}$ by noting that simultaneously permuting the second and third rows and columns (to obtain a block form corresponding to the partition P) gives[8]:

$$\begin{bmatrix}1&0&0&0\\0&0&1&0\\0&1&0&0\\0&0&0&1\end{bmatrix}\begin{bmatrix}0&0&1&0\\0&1&0&0\\1&0&0&0\\0&0&0&1\end{bmatrix}\begin{bmatrix}1&0&0&0\\0&0&1&0\\0&1&0&0\\0&0&0&1\end{bmatrix}^T = \begin{array}{c} \\ 1\\3\\2\\4\end{array}\begin{array}{c}\begin{array}{cccc}1&3&2&4\end{array}\\\begin{pmatrix}0&1&&\\1&0&&\\&&1&0\\&&0&1\end{pmatrix}\end{array}$$

One could also obtain the same result by simultaneously permuting columns and rows to obtain another block form corresponding to the partition P:

$$\begin{bmatrix}0&1&0&0\\0&0&0&1\\1&0&0&0\\0&0&1&0\end{bmatrix}\begin{bmatrix}0&0&1&0\\0&1&0&0\\1&0&0&0\\0&0&0&1\end{bmatrix}\begin{bmatrix}0&1&0&0\\0&0&0&1\\1&0&0&0\\0&0&1&0\end{bmatrix}^T = \begin{array}{c} \\ 2\\4\\1\\3\end{array}\begin{array}{c}\begin{array}{cccc}2&4&1&3\end{array}\\\begin{pmatrix}1&0&&\\0&1&&\\&&0&1\\&&1&0\end{pmatrix}\end{array}$$

Finally, the statement of the Protasov-Voynov characterisation theorem is as follows.

Theorem 3 (Protasov and Voynov's theorem [34])**.** *An irreducible, NZ matrix set $\mathcal{M}$ of $n \times n$ matrices is not primitive iff there exists a partition of $\{1,\ldots,n\}$ on which every matrix from $\mathcal{M}$ acts as a permutation.*

Comparing this result to Theorem 1, one might expect to find some relationship between the notion of aperiodicity and the notion of matrices acting as permutations over a partition of the index set. The results of Chapter 4 shed some light on this relationship.

Protasov and Voynov's proof of Theorem 3 was based on a geometric approach, and they conjectured that the theorem could be proved using purely combinatorial reasoning. Blondel, Jungers and Olshevsky answered this question by giving a purely combinatorial proof in [7]. Another – arguably simpler – proof was provided by Al'pin and Al'pina in [1]. They also generalized the theorem to sets of matrices with no zero rows (or columns) in [2].

[8]The blank entries are all zeros.

It was shown in [7] that recognizing primitive matrix sets is an NP hard problem for matrix sets with more than 2 matrices. However, for NZ matrix sets, Protasov and Voynov presented a polynomial time algorithm solving this problem in [34]. Blondel et al also showed in [7] that the exponent of a matrix set could be exponentially large. However, for NZ primitive matrix sets we shall see that it is possible to obtain a cubic bound on the exponent with respect to the matrix size.

1.4 Connecting synchronizing automata to primitive matrix sets

In recent years, various connections have been discovered between the theory of synchronizing automata and the theory of primitive matrix sets as the two problems turn out to be related. For example, primitive matrix sets have been used in algorithms for finding slowly synchronizing automata, see [11, 4].

The following definition was suggested in [7], relating matrix sets and automata (See Remark 1 for the representation of automata as matrix sets):

Definition 10. *Let $\mathcal{M}$ be a $\mathcal{NZ}$ and finite set of binary $n \times n$ matrices. The automaton associated to the set $\mathcal{M}$ is the automaton $Aut(\mathcal{M})$ such that $A \in Aut(\mathcal{M})$ if and only if A is a binary and row-stochastic matrix and there exists $M \in \mathcal{M}$ such that $A \leqslant M$ (entrywise). We denote with $Aut(\mathcal{M}^T)$ the automaton associated to the set $\mathcal{M}^T = \{M_1^T, \dots, M_m^T\}$.*

The following theorem was also proved, showing a strong connection between the exponent of a primitive matrix set and the reset thresholds of two associated automata:

Theorem 4 ([7], Theorems 16-17)**.** *Let $\mathcal{M} = \{M_1, \dots, M_m\}$ be a primitive set of binary NZ matrices. Then $Aut(\mathcal{M})$ and $Aut(\mathcal{M}^T)$ are synchronizing and it holds that:*

$$rt\big(Aut(\mathcal{M})\big) \leqslant exp(\mathcal{M}) \leqslant rt\big(Aut(\mathcal{M})\big) + rt\big(Aut(\mathcal{M}^T)\big) + n - 1. \tag{1.1}$$

The restriction to *binary* matrix sets in Theorem 4 is without loss of generality, as the primitivity of a matrix set does not depend on the magnitude of the positive entries of the matrices of the set. In view of this fact, in this manuscript we will restrict ourselves to the set of binary matrices by using the boolean product as opposed to the usual matrix product, so setting for any binary matrices A and B that $(AB)_{ij} = 1$ if and only if $\sum_s A_{is}B_{sj} > 0$.

We note in particular that a primitive matrix set with an exponent greater than $2(n-1)^2+n-1$ would disprove the Černý conjecture. However it was also shown (in [7]) as a corollary of the preceding theorem that a general bound on the reset threshold of synchronizing automata automatically gives a bound on the exponent of any NZ primitive matrix set:

Corollary 1 (See [7], Theorem 17)**.** *For any NZ primitive set of matrices M of dimension n there is a product of length smaller than $2f(n)+n-1$ with positive entries, where $f(n)$ is any upper bound on the minimal length of a synchronizing word for n-state automata.*

In view of Szykula's bound on the reset threshold of a synchronizing automaton (see [42]) one can easily obtain, using the preceding corollary, the following cubic bound on the exponent of an NZ primitive matrix set of matris size n, written $\exp_{NZ}(n)$.

$$\exp_{NZ}(n) \leqslant (15617n^3+7500n^2+56250n-78125)/46875 \tag{1.2}$$

In this thesis we shall also extend the notion of the k-RT from automata to matix sets via the following definition.

Definition 11. *Let $\mathcal{M}$ be a primitive set of $n\times n$ NZ matrices and $2\leqslant k\leqslant n$. We define the* k-rendezvous time *(k-RT) of $\mathcal{M}$ to be the length of the shortest product of matrices from $\mathcal{M}$ having a column or a row with k positive entries and we denote it by $rt_k(\mathcal{M})$. We indicate with $rt_k(n)$ the maximal value of $rt_k(\mathcal{M})$ among all the primitive sets $\mathcal{M}$ of $n\times n$ NZ matrices.*

We shall obtain some upper bounds on the k-RT of NZ primitive matrix sets in Chapter 3. Indeed, we shall show in particular that for large n, $rt_k(n)$ is bounded by a linear function of n.

Chapter 2

Preliminaries

We begin by defining some notions and proving some simple results that will be used extensively throughout this manuscript.

2.1 Associated graph of a matrix set

Definition 12. *Let $\mathcal{M}$ be a finite set of binary $n \times n$ matrices. We define the associated graph of $\mathcal{M}$, as the directed graph $\mathcal{G}(\mathcal{M}) = (V_{\mathcal{M}}, E_{\mathcal{M}})$, where the set of vertices is $V_{\mathcal{M}} = \{1, \ldots, n\}$. The edge $(i, j) \in E_{\mathcal{M}} \iff \exists M \in \mathcal{M} : M_{ij} > 0$, and we label the edge with all the matrices in $\mathcal{M}$ that satisfy this property.*

Definition 13. *A directed graph $G = (V, E)$ is strongly connected if and only if for all $i, j \in V$, there is a path from i to j in G.*

Lemma 1. *Let $\mathcal{M}$ be a finite set of binary $n \times n$ matrices. The associated graph of $\mathcal{M}$ is strongly connected if and only if $\mathcal{M}$ is irreducible.*

Proof. Trivial by Definition 8. □

Definition 14. *We define the* support *of a vector x, denoted by* $\mathrm{supp}(x)$*, as the following set :* $\mathrm{supp}(x) = \{i \mid x_i > 0\}$*. We define the* weight *of a vector as the cardinality of its support.*

Lemma 2. *Let $\mathcal{M}$ be a finite set of binary $n \times n$ matrices. Let $C \in \mathcal{M}$ and i, j two indices such that $C_{ij} > 0$. Then for any $A \in \mathcal{M} : \mathrm{supp}(A_{*i}) \subseteq \mathrm{supp}((AC)_{*j})$.*

Proof. The j-th column of AC is equal to $(AC)_{*j} = C_{1j}A_{*1} + \cdots + C_{nj}A_{*n}$. The result follows as $C_{ij} > 0$. □

Lemma 3. *Let $\mathcal{M}$ be a finite set of binary $n \times n$ matrices and let $M_{i_1}, \ldots, M_{i_s} \in \mathcal{M}$. Then $(M_{i_1} \ldots M_{i_s})_{ij} > 0$ iff there is a path labeled by $M_{i_1}, \ldots, M_{i_s}$ from i to j in $\mathcal{G}(\mathcal{M})$.*

Proof. We prove the claim by induction. By definition of $\mathcal{G}(\mathcal{M})$, the lemma is trivially true for $s = 1$.

Suppose now that the proposition is true for some $s \in \mathbb{N}_0$. Suppose also that $M_{i_{s+1}}$ is a matrix such that $(M_{i_1} \dots M_{i_s} M_{i_{s+1}})_{ij} > 0$. Then since all the matrices are nonnegative, this implies that there exists $k \in \{1, ..., n\}$ such that $(M_{i_1} \dots M_{i_s})_{ik} > 0$ and $(M_{i_{s+1}})_{kj} > 0$. Therefore there exists a path in $\mathcal{G}(\mathcal{M})$ from i to k and from k to j. It follows that there is a path from i to j in $\mathcal{G}(\mathcal{M})$.

Suppose now that there exists a path in $\mathcal{G}(\mathcal{M})$ from i to j labeled by $M_{i_1}, \dots, M_{i_s}, M_{i_{s+1}}$. Then there exists a vertex $k \in \{1, ..., n\}$ such that there is a path from i to k labeled by $M_{i_1}, \dots, M_{i_s}$ and there exists an edge from k to j labeled by $M_{i_{s+1}}$. This implies that $(M_{i_1} \dots M_{i_s})_{ik} > 0$ and $(M_{i_{s+1}})_{kj} > 0$, it follows by nonnegativity of $\mathcal{M}$ that $(M_{i_1} \dots M_{i_s} M_{i_{s+1}})_{ij} > 0$. □

Proposition 1. *Let $\mathcal{M}$ be an irreducible and finite set of binary $n \times n$ matrices, $A \in \mathcal{M}$, and let A_{*i} be an arbitrary column of A, then for any $j = 1, \dots, n$ there exists $B \in \mathcal{M}^{n-1}$ such that* $\operatorname{supp}(A_{*i}) \subseteq \operatorname{supp}((AB)_{*j})$.

Proof. Since $\mathcal{M}$ is irreducible, $\mathcal{G}(\mathcal{M})$ is strongly connected and there is a path of length at most $n - 1$ between any two nodes in $\mathcal{G}(\mathcal{M})$. Therefore there exists a path $a_1 = i \to a_2 \to \dots \to a_m = j$ in $\mathcal{G}(\mathcal{M})$, where $m \leqslant n$. Picking one matrix from the label of each edge in this path, we find a sequence of matrices $C^1, \dots, C^{m-1} \in M$. By Lemma 3, we know that $(C^1 \dots C^{m-1})_{a_1, a_m} > 0$. Applying Lemma 2, we conclude that:

$$\operatorname{supp}(A_{*i}) \subseteq \operatorname{supp}((AC^1 \dots C^{m-1})_{*j})$$

Therefore the required matrix B is given by $C^1 \dots C^{m-1}$. □

2.2 Permutation matrices

We prove some results about permutation matrices here that will be used in Chapter 4 to study characteristics of the automata generated by a randomized algorithm for generating slowly synchronizing automata (see [11]).

Definition 15. *We say that a matrix P is a permutation matrix (of size $n \times n$) iff P is a binary matrix with exactly one nonzero entry in every row and every column. Moreover, let δ_i^j be such that*

$$\delta_i^j = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{otherwise.} \end{cases}$$

and e_i be a column vector such that $(e_i)_j = \delta_i^j$. We say that a permutation matrix P is the permutation matri associated to a permutation π of $\{1, \ldots, n\}$ iff $P = [e_{\pi(1)}, \ldots, e_{\pi(n)}]$.

Remark 2. *For the remainder of this thesis we let $\mathbb{P}_n$ denote the set of $n \times n$ permutation matrices.*

Lemma 4. *Let A be a binary $n \times n$ matrix, and P be a permutation matrix. The graph $\mathcal{G}(\{PAP^T\}) = (V, E_{PAP^T})$ is such that $(\pi(i), \pi(j)) \in E_{PAP^T} \iff (i, j) \in E_A$*

Proof. Let δ_i^j be as defined above. Let $i' = \pi(i), j' = \pi(j)$, we calculate:

$$\begin{aligned}(PAP^T)_{i'j'} &= \sum_a \sum_b P_{i'a} A_{ab} (P^T)_{bj'} = \sum_a \sum_b P_{i'a} A_{ab} P_{j'b} \\ &= \sum_a \sum_b \delta^{i'}_{\pi(a)} A_{ab} \delta^{j'}_{\pi(b)} = A_{ij}\end{aligned}$$

and this implies that:

$$(\pi(i), \pi(j)) \in E_{PAP^T} \iff (i, j) \in E_A$$

□

The following lemma fully characterizes the associated graph of a permutation matrix.

Lemma 5. *Every strongly connected component of the graph $\mathcal{G}(\{P\})$ is a cycle iff P is a permutation matrix.*

Proof. Let P be a permutation matrix, and $\mathcal{G}(\{P\}) = (V, E)$ its associated graph. For any $i_1 \in V$, there is exactly one edge (i_1, i_2) out of it, and the same holds true for i_2, etc. So we can create exactly one path $i_1 \to i_2 \to \ldots$, but since the graph is finite this path must end somewhere, either on some node that has no out-edges (not possible), or the sequence eventually returns to some previously visited vertex, but then the first revisited vertex must be i_1, otherwise some node would have two in-edges. Therefore we have just proved that the strongly connected component containing i_1 must be a cycle.

Conversely, let each strongly connected component in $\mathcal{G}(\{P\}) = (V, E)$ be a cycle, then there can be no edges between two distinct strongly connected components as then they would not be cycles. Hence there is exactly one edge in and out of each vertex, and so P has exactly one nonzero entry per row and per column, and is therefore a permutation matrix. □

In what follows, we shall often refer to the cycle lengths of a permutation matrix as a convenient shorthand for the lengths of the cycles in the associated graph of the permutation matrix. In Chapter 4, the *max cycle length of a permutation matrix*, shall play an important role, it is defined here below:

Definition 16. *Let P be the permutation matrix. The* max cycle length of P, *written $\mathscr{L}_*(P)$, is the length of the longest cycle in the associated graph of P.*

2.3 Pair digraph of a matrix set

Definition 17. *Let $\mathcal{M}$ be a finite set of binary $n \times n$ matrices. The pair digraph of the matrix set $\mathcal{M}$ is a digraph $\mathcal{PD}(\mathcal{M}) = (V_\mathcal{M}, E_\mathcal{M})$ with $V_\mathcal{M} = \{(i,j) \mid 1 \leqslant i \leqslant j \leqslant n\}$ and $((i,j),(i',j')) \in E_\mathcal{M} \iff \exists M \in \mathcal{M} : M_{ii'} > 0, M_{jj'} > 0$ or $M_{ji'} > 0, M_{ij'} > 0$. A vertex of the form (s,s) is called a singleton.*

Lemma 6. *Let $\mathcal{M}$ be a finite set of binary $n \times n$ matrices. A path of length d in the pair digraph $\mathcal{PD}(M)$ connecting the non-singleton (i,j) to a singleton (s,s) defines a sequence of matrices $A_1, \ldots, A_d$ in $\mathcal{M}$, such that for any $A \in \mathcal{M}$:* $\mathrm{supp}(A_{*i}) \bigcup \mathrm{supp}(A_{*j}) \subseteq \mathrm{supp}((AA_1 \ldots A_d)_{*s})$.

Proof. The path from (i,j) to (s,s) defines a sequence of matrices $A_1, \ldots, A_d$ in $\mathcal{M}$, such that $(A_1 \ldots A_d)_{i,s} > 0$ and $(A_1 \ldots A_d)_{j,s} > 0$. The thesis follows by applying Lemma 2 with $C = A_1 \ldots A_d$. □

2.4 Synchronizability and primitivity

Theorem 5. *Let $\mathcal{M}$ be a finite set of binary $n \times n$ matrices with non zero rows. Then there exists $M \in \mathcal{M}^*$ and $j \in \{1, ..., n\}$ such that for all $i \in \{1, ..., n\}$, $M_{ij} > 0$ if and only if in its pair digraph $\mathcal{PD}(\mathcal{M}) = (V_\mathcal{M}, E_\mathcal{M})$, for all $(i,j) \in V_\mathcal{M}$, there exists a path from (i,j) to some singleton.*

Proof. If there exists $M \in \mathcal{M}^*$ and $j \in \{1, ..., n\}$ such that for all $i \in \{1, ..., n\}$, $M_{ij} > 0$, then by definition of the pair digraph, for all $i, k \in \{1, ..., n\}$ there exists a path from (i,k) to (j,j) in the pair digraph.

Let us suppose now that for all $i, j \in \{1, ..., n\}$, there exists a path from (i,j) to some singleton in the pair digraph. Let us take an arbitrary matrix $B \in \mathcal{M}$. Let $j_1 \in \{1, ..., n\}$. If $|\mathrm{supp}(B_{*j_1})| = n$, we are done. Otherwise, there exists $i \in \{1, ..., n\}$ such that $i \notin \mathrm{supp}(B_{*j})$. Since B has non zero rows, there exists $j_2 \in \{1, ..., n\}$ such that $B_{i,j_2} > 0$. Since there exists $s \in \{1, ..., n\}$ such that there is a path in $\mathcal{PD}(\mathcal{M})$ from (j_1, j_2) to (s,s), by Lemma 6, there exists a matrix

$D \in \mathcal{M}^*$ such that $|\text{supp}(B_{*j_1})| < |\text{supp}(BD)_{*s}|$. If we repeat this process, it is clear that we will have the desired matrix M in at most $n-1$ steps. □

Remark 3. *It follows from the preceding theorem that given an automaton $\mathcal{A}$, we can easily check synchronizability of $\mathcal{A}$ by checking whether the matrix set representing $\mathcal{A}$ (see Remark 1) satisfies the conditions of the theorem.*

It is interesting that the same necessary and sufficient condition suffices for primitivity of irreducible NZ matrix sets. This is proved below.

Lemma 7. *An irreducible, NZ and finite matrix set $\mathcal{M}$ of $n \times n$ matrices is primitive if and only if in its pair digraph $\mathcal{PD}(\mathcal{M}) = (V_\mathcal{M}, E_\mathcal{M})$, for all $(i,j) \in V_\mathcal{M}$, there exists a path from (i,j) to some singleton.*

Proof. Suppose $\mathcal{M}$ is primitive. Then the existence of a positive product makes $\mathcal{PD}(\mathcal{M})$ strongly connected. This trivially implies that for all $(i,j) \in V_\mathcal{M}$, there exists a path from (i,j) to some singleton.

Let us now suppose that for all $(i,j) \in V_\mathcal{M}$, there exists a path from (i,j) to some singleton. Let $A \in \mathcal{M}$. Since $\mathcal{M}$ is $\mathcal{NZ}$, for each $i = 1, \ldots, n$ there exists a column $j_i \in \{1, \ldots, n\}$ such that $A_{ij_i} > 0$. Let $s_1 = j_1$. For $i = 1, ..., n-1$, let s_{i+1} be one of the vertices such that there exists a path from (s_i, j_{i+1}) to (s_{i+1}, s_{i+1}). Then, according to Lemma 6, by post-multiplying A with matrices in the labels of the edges of these paths we can produce a matrix A' that merges the columns indexed by $j_1, \ldots, j_n$ into an all-ones column in AA'.

We now prove by induction the existence of a positive matrix in $\mathcal{M}$. WLOG we suppose that the first k columns of some matrix A are positive (recall that Proposition 1 proves the existence of a matrix with a positive first column given a matrix with any positive column). If $k = n$ then we are done, otherwise, since A has a positive entry in each column, let i_{k+1} be the row-index of a positive entry in column $k+1$. Letting $B \in \mathcal{M}$ such that the i_{k+1}-th column of B is positive (again, Proposition 1 ensures its existence), we conclude by noting that the $k+1$ first columns of BA are positive. □

Chapter 3

Bounding the k-RT of NZ primitive matrix sets

In this chapter, we consider a problem very similar to the problem of finding an upper bound on the reset thresholds of synchronizing automata: we find an upper bound on the k-rendezvous time of a primitive set of NZ matrices.

In view of the connection between synchronizing automata and primitive sets, in Chapter 1, we extended the k-RT problem to primitive sets by introducing the k-RT of a primitive set $\mathcal{M}$, which is the length of the shortest product having a row or a column with k positive entries. The following proposition shows how the k-RT of a primitive set $\mathcal{M}$ of NZ matrices (denoted by $rt_k(\mathcal{M})$) is linked to the length of the shortest word for which there exists a set of k states mapped by it onto a single state in the automata $Aut(\mathcal{M})$ and $Aut(\mathcal{M}^T)$ (denoted respectively by $rt_k(Aut(\mathcal{M}))$ and $rt_k(Aut(\mathcal{M}^T))$).

Proposition 2. *Let $\mathcal{M}$ be a primitive set of $n \times n$ binary NZ matrices and let $Aut(\mathcal{M})$ and $Aut(\mathcal{M}^T)$ be the automata defined in Definition 10. Then for every $2 \leqslant k \leqslant n$, it holds that $rt_k(\mathcal{M}) = \min\{rt_k(Aut(\mathcal{M})), rt_k(Aut(\mathcal{M}^T))\}$.*

Proof. We remind that the *support* of a nonnegative vector v is the set of the indices of its positive entries and its *weight* is the cardinality of its support.
By Definition 10, each matrix of $Aut(\mathcal{M})$ and $Aut(\mathcal{M}^T)$ is entrywise smaller than a matrix of $\mathcal{M}$. It follows that $rt_k(\mathcal{M}) \leqslant \min\{rt_k(Aut(\mathcal{M})), rt_k(Aut(\mathcal{M}^T))\}$.
Let now $M = M_{i_1} \cdots M_{i_u}$ be the product that attains the k-RT, that is a product of length $rt_k(\mathcal{M})$ having a column or a row with k positive entries. Suppose that M has a column with k positive entries: we show that $rt_k(\mathcal{M}) \geqslant rt_k(Aut(\mathcal{M}))$. Let j be the index of this column and S be its support. We claim that for every $r \in [u]$ we can safely set to zero some entries of M_{i_r} in order to make its rows be stochastic while making sure that the final product still has the j-th column

with support S. In other words, we claim that for every $r \in [u]$ we can select a binary row-stochastic matrix $A_r \leqslant M_{i_r}$ (entrywise) such that the j-th column of the product $A_1 \cdots A_u$ has support S. If this is true, since by hypothesis the matrices $A_1, \ldots, A_u$ belong to $Aut(\mathcal{M})$, it holds that $rt_k\big(Aut(\mathcal{M})\big) \leqslant rt_k(\mathcal{M})$. We now prove the claim: let D_r be the digraph on n vertices and edge set E_r such that $p \to q \in E_r$ if and only if $(M_{i_r})_{pq} > 0$. The fact that the j-th column of $M_{i_1} \cdots M_{i_u}$ has support S means that for every $s \in S$ there exists a sequence of vertices $v_1^s, \ldots, v_{u+1}^s \in [n]$ such that:

$$v_1^s = s \ , \tag{3.1}$$

$$v_{u+1}^s = j \ , \tag{3.2}$$

$$v_r^s \to v_{r+1}^s \in E_r \quad \forall r = 1, \ldots, u \ . \tag{3.3}$$

We can impose an additional property on these sequences: if at step t two sequences share the same vertex, then they have to coincide for all the steps $t' > t$. More formally, if for some $t \in [u]$ we have that $v_t^s = v_t^{s'}$ for $s \neq s'$, then we set $v_{t'}^{s'} = v_{t'}^s$ for all $t' > t$ as the new sequence $v_1^{s'}, \ldots, v_t^{s'}, v_{t+1}^s, \ldots, v_{u+1}^s$ for vertex s' fulfills all the requirements (3.1), (3.2) and (3.3). For every $r \in [u]$, we now remove from E_r all the edges that are not of type (3.3). Furthermore, for every $r \in [u]$ and vertex $w \notin \{v_r^s\}_{s \in S}$, we remove from E_r all the outgoing edges of w but one. We call this new edge set $\tilde{E}_r$ and let $\tilde{D}_r$ be the subgraph of D_r with edge set $\tilde{E}_r$. Then, for every $r \in [u]$, we set A_r to be the adjacency matrix of $\tilde{D}_r$. Since M_{i_r} is NZ, if A_r has some zero-rows we can always add a one in each of them while preserving the property $A_r \leqslant M_{i_r}$. We do so in order to make A_r row stochastic. By construction, for all $r \in [u]$, A_r has exactly one positive entry in each row and it is entrywise smaller than M_{i_r}, so $A_r \in Aut(\mathcal{M})$. Finally, the j-th column of $A_1 \cdots A_u$ has support S by construction.

The case when M has a row with k positive entries can be proved via a similar reasoning by observing that the product $M^T = M_{i_u}^T \cdots M_{i_1}^T$ has a column with k positive entries, and so for every $r \in [u]$ we can select a binary matrix $B_r \leqslant M_{i_r}^T$ (entrywise) such that $B_r \in Aut(\mathcal{M}^T)$ and $B_1 \cdots B_u$ has a column with k positive entries. This implies that $rt_k\big(Aut(\mathcal{M}^T)\big) \leqslant rt_k(\mathcal{M})$. □

In view of the above proposition, by upper bounding the k-RT of all primitive sets of $\mathcal{NZ}$ matrices, then for any of these sets $\mathcal{M}$, we also bound the minimum between the k-RT of the automata associated to $\mathcal{M}$, and the k-RT of the automata associated to $\mathcal{M}^T$. In this chapter, we therefore attempt to find a bound on the k-RT of primitive sets of NZ matrices. It is hoped that this may shed light on the behaviour of the k-RT of automata associated to these sets, and thereby also on the Černý's conjecture.

In the first section, we find a recurrence relation for a function that upper bounds the k-rendezvous time of NZ primitive matrix sets. In the second section, we solve this recurrence relation and show that for any k, the k-rendezvous time of an NZ primitive matrix set with matrices of dimension n is bounded by a linear function in n for large n. In the third section, we show that we cannot improve this bound by improving the value of one parameter that will be defined later. The fourth section shows some numerical results. The two next sections make some small modifications to the original argument in order to derive an improved upper bound.
The results of sections 3.1 to 3.4 were submitted (and accepted) in a conference paper to Developments in Language Theory 2019, see [5].

3.1 Derivation of a recurrence relation bounding the k-RT

In this section, we prove a recurrence relation for a function $B_k(n)$ that upper bounds the k-RT.

Our goal is then to find, for any $n \geqslant 2$ and $2 \leqslant k \leqslant n$, a function $B_k(n)$ such that $rt_k(n) \leqslant B_k(n)$.

Definition 18. *Let n, k integers such that $n \geqslant 2$ and $2 \leqslant k \leqslant n-1$. We denote by $\mathcal{S}_k^n$ the set of all the $n \times n$ NZ matrices having every row and column of weight at most k and at least one column of weight exactly k. For any $A \in \mathcal{S}_n^k$, let $\mathcal{C}_A$ be the set of the indices of the columns of A having weight equal to k. We define $a_k^n(A) = \min_{c \in \mathcal{C}_A} |\{i : supp(A_{*i}) \nsubseteq supp(A_{*c})\}|$ and $a_k^n = \min_{A \in \mathcal{S}_n^k} a_k^n(A)$.*

In other words, $a_k^n(A)$ is the minimum over all the indices $c \in \mathcal{C}_A$ of the number of columns of A whose support is not contained in the support of the c-th column of A. Since the matrices are NZ, it clearly holds that for any $A \in \mathcal{S}_n^k$, $1 \leqslant a_k^n \leqslant a_k^n(A)$. The following theorem shows that for every $n \geqslant 2$, we can recursively define a function $B_k(n) \geqslant rt_k(n)$ on k by using the term a_k^n.

Theorem 6. *Let $n \geqslant 2$ integer. The following recursive function $B_k(n)$ is such that for all $2 \leqslant k \leqslant n$, $rt_k(n) \leqslant B_k(n)$.*

$$\begin{cases} B_2(n) = 1 \\ B_{k+1}(n) = B_k(n) + n(1 + n - a_k^n)/2 \quad \textit{for } 2 \leqslant k \leqslant n-1. \end{cases} \tag{3.4}$$

Proof. We prove the theorem by induction, for the base case, let $k = 2$. Any primitive set of NZ matrices must have a matrix with a row or a column with two positive entries, as otherwise it would be made of just permutation matrices and hence it would not be primitive. This trivially implies that $rt_2(n) = 1 \leqslant B_2(n)$.

Suppose now that $rt_k(n) \leqslant B_k(n)$, we show that $rt_{k+1}(n) \leqslant B_{k+1}(n)$. We remind that we denote with $\mathcal{M}^d$ the set of all the products of matrices from $\mathcal{M}$ having length smaller than or equal to d. If in $\mathcal{M}^{rt_k(\mathcal{M})+n-1}$ there exists a product having a column or a row with $k+1$ positive entries then $rt_{k+1}(\mathcal{M}) \leqslant rt_k(\mathcal{M}) + n - 1 \leqslant B_{k+1}(n)$.

Suppose now that this is not the case. This means that in $\mathcal{M}^{rt_k(\mathcal{M})+n-1}$ every matrix has all the rows and columns of weight at most k. Let $A \in \mathcal{M}^{rt_k(\mathcal{M})}$ be a matrix having a row or a column of weight k, and suppose it is a column. The case when A has a row of weight k will be studied later. By Proposition 1 applied on the matrix A, for every $i \in [n]$ there exists a matrix $W_i \in \mathcal{M}^{rt_k(\mathcal{M})+n-1}$ having the i-th column of weight k (and all the other columns and rows of weight $\leqslant k$). Every W_i has at least a_k^n (see Definition 18) columns whose support is not contained in the support of the i-th column of W_i: let $c_i^1, c_i^2, \ldots, c_i^{a_k^n}$ be the indices of these columns. Notice that any product B of matrices from $\mathcal{M}$ of length l such that $B_{is} > 0$ and $B_{c_i^j s} > 0$ for some $s \in [n]$ and $j \in [a_k^n]$ would imply that $W_i B$ has the s-th column of weight at least $k+1$ and so $rt_{k+1}(\mathcal{M}) \leqslant rt_k(\mathcal{M}) + n - 1 + l$. We now want to minimize this length l over all $i, s \in [n]$ and $j \in [a_k^n]$: we will prove that there exists $i, s \in [n]$ and $j \in [a_k^n]$ such that $l \leqslant n(n - 1 - a_k^n)/2 + 1$. To do this, we consider the pair digraph $\mathcal{PD}(\mathcal{M}) = (\mathcal{V}, \mathcal{E})$ (see Definition 17) and the vertices:

$$\begin{array}{ccccc} (1,c_1^1), & (1,c_1^2), & \ldots & , (1,c_1^{a_k}) \\ (2,c_2^1), & (2,c_2^2), & \ldots & , (2,c_2^{a_k}) \\ \vdots & \vdots & & \vdots \\ (n,c_n^1), & (n,c_n^2), & \ldots & , (n,c_n^{a_k}) \end{array} \tag{3.5}$$

By Lemma 7, for each vertex in Eq.(3.5) there exists a path in $\mathcal{PD}(\mathcal{M})$ connecting it to a singleton. By lemma 6, a path of length l from (i, c_i^j) to a singleton (s, s) would result in a product B_j of matrices from $\mathcal{M}$ of length l such that $W_i B_j$ has the s-th column of weight at least $k+1$. We hence want to estimate the minimal length among the paths connecting the vertices in Eq.(3.5) to a singleton. Notice that Eq.(3.5) contains at least $\lceil n a_k^n / 2 \rceil$ different elements, since each element occurs at most twice. It is clear that the shortest path from a vertex in the list (3.5) to a singleton does not contain any other element from that list. The vertex set $\mathcal{V}$ of $\mathcal{PD}(\mathcal{M})$ has cardinality $n(n+1)/2$ and it contains n vertices of type (s, s). It fol-

lows that the length of the shortest path connecting some vertex from the list (3.5) to some singleton is at most of $n(n+1)/2 - n - \lceil na_k^n/2 \rceil + 1 \leqslant n(n-1-a_k^n)/2 + 1$. In view of what said before, we have that there exists a product B of matrices from $\mathcal{M}$ of length $\leqslant n(n-1-a_k^n)/2+1$ and $i \in [n]$ such that W_iB_j has a column of weight at least $k+1$. Since W_iB_j belongs to $\mathcal{M}^{rt_k(\mathcal{M})+n-1+n(n-1-a_k^n)/2+1}$, it follows that $rt_{k+1}(\mathcal{M}) \leqslant rt_k(\mathcal{M}) + n(n+1-a_k^n)/2 \leqslant B_{k+1}(n)$.

Suppose now $A \in \mathcal{M}^{rt_k(\mathcal{M})}$ has a row of weight k. We can use the same argument as above on the matrix set $\mathcal{M}^T$ made of the transpose of all the matrices in $\mathcal{M}$. □

Notice that the above argument stays true if we replace a_k^n by a function $b(n,k)$ such that for all $n \geqslant 2$ and $2 \leqslant k \leqslant n-1$, $1 \leqslant b(n,k) \leqslant a_k^n$. It follows that Eq.(3.4) still holds true if we replace a_k^n by $b(n,k)$.

3.2 Solution of the recurrence relation

We now find an analytic expression for a lower bound on a_k^n and we then solve the recurrence (3.4) in Theorem 6 by using this lower bound. We then show that this is the best estimate on a_k^n we can hope for.

Lemma 8. *Let n, k integers such that $n \geqslant 2$ and $2 \leqslant k \leqslant n-1$, and let a_k^n as in Definition 18. It holds that $a_k^n \geqslant n - k(k-1) - 1$.*

Proof. Let now $A \in S_n^k$ (see Definition 18) and let a be one of its columns of weight k. Let $S = supp(a)$; by assumption, the rows of A have at most k positive entries, so there can be at most $(k-1)k$ columns of A different from a whose support is contained in S. Therefore, since A is NZ, there must exist at least $n - k(k-1) - 1$ columns of A whose support is not contained in $supp(a)$ and so $a_k^n \geqslant n - k(k-1) - 1$. □

Lemma 9. *Let n, k integers such that $n \geqslant 2$ and $2 \leqslant k \leqslant n-1$, and let a_k^n as in Definition 18. It holds that $a_k^n \geqslant \lceil (n-k)/k \rceil$.*

Proof. Let again $A \in S_n^k$ and let a be one of its columns of weight k. Let $S = [n] \backslash supp(a)$; S has cardinality $n-k$ and since A is NZ, for every $s \in S$ there exists $s' \in [n]$ such that $A_{ss'} > 0$. By assumption each column of A has weight of at most k, so there must exist at least $\lceil (n-k)/k \rceil$ columns of A different from a whose support is not contained in $supp(a)$. It follows that $a_k^n \geqslant \lceil (n-k)/k \rceil$. □

Lemma 10. *Let n, k integers such that $n \geqslant 2$ and $2 \leqslant k \leqslant n-1$, and let a_k^n as in Definition 18. It holds that $a_k^n \geqslant \max\{n - k(k-1) - 1, \lceil (n-k)/k \rceil, 1\}$.*

Proof. We have that $a_k^n \geqslant 1$ since $k \leqslant n-1$ and the matrices are NZ. The conclusion follows from Lemma 8 and Lemma 9. □

Since $\lceil (n-k)/k \rceil \geqslant (n-k)/k$, $n - k(k-1) - 1 \geqslant (n-k)/k$ for $k \leqslant \lfloor \sqrt{n} \rfloor$ and $(n-k)/k \geqslant 1$ for $k \leqslant \lfloor n/2 \rfloor$, the recursion (3.4) with a_k^n replaced by $\max\{n - k(k-1) - 1, (n-k)/k, 1\}$ now reads as:

$$\tilde{B}_{k+1}(n) = \begin{cases} 1 & \text{if } k = 1 \\ \tilde{B}_k(n) + n(1 + k(k-1)/2) & \text{if } 2 \leqslant k \leqslant \lfloor \sqrt{n} \rfloor \\ \tilde{B}_k(n) + n(1 + n(k-1)/2k) & \text{if } \lfloor \sqrt{n} \rfloor + 1 \leqslant k \leqslant \lfloor n/2 \rfloor \\ \tilde{B}_k(n) + n^2/2 & \text{if } \lfloor n/2 \rfloor + 1 \leqslant k \leqslant n-1 \end{cases} . \quad (3.6)$$

The following proposition shows the solution of the recursion (3.6):

Theorem 7. *Equation (3.6) is fulfilled by the following function:*

$$\tilde{B}_k(n) = \begin{cases} \dfrac{n(k^3 - 3k^2 + 8k - 12)}{6} + 1 & \text{if } 2 \leqslant k \leqslant \lfloor \sqrt{n} \rfloor \\ \tilde{B}_{\lfloor \sqrt{n} \rfloor}(n) + \dfrac{n(n+2)(k - \lfloor \sqrt{n} \rfloor)}{2} - \dfrac{n^2}{2} \displaystyle\sum_{i=\lfloor \sqrt{n} \rfloor}^{k-1} \dfrac{1}{i} & \text{if } \lfloor \sqrt{n} \rfloor + 1 \leqslant k \leqslant \lfloor \frac{n}{2} \rfloor \\ \tilde{B}_{\lfloor \frac{n}{2} \rfloor}(n) + \dfrac{(k - \lfloor \frac{n}{2} \rfloor) n^2}{2} & \text{if } \lfloor \frac{n}{2} \rfloor + 1 \leqslant k \leqslant n \end{cases} . \quad (3.7)$$

Therefore, for any constant k such that $k \leqslant \sqrt{n}$, the k-rendezvous time $rt_k(n)$ is at most linear in n.

Proof. If $2 \leqslant k \leqslant \lfloor \sqrt{n} \rfloor$, let $C_k(n) = \tilde{B}_k(n)/n$. By Eq.(3.6), it holds that:

$$C_{k+1}(n) - C_k(n) = 1 + k(k-1)/2.$$

By setting $C_k(n) = \alpha k^3 + \beta k^2 + \gamma k + \delta$, it follows that:

$$3\alpha k^2 + (3\alpha + 2\beta)k + \alpha + \beta + \gamma = k^2/2 - k/2 + 1.$$

Since this must be true for all k, by equating the coefficients we have that:

$$C_k(n) = k^3/6 - k^2/2 + 4k/3 + \delta.$$

Imposing the initial condition $\tilde{B}_2(n) = 1$ gives finally the desired result

$$\tilde{B}_k(n) = n(k^3 - 3k^2 + 8k - 12)/6 + 1.$$

If $\lfloor\sqrt{n}\rfloor + 1 \leqslant k \leqslant \lfloor n/2\rfloor$, let again $C_k(n) = \tilde{B}_k(n)/n$. By Eq.(3.6), it holds that:

$$C_{k+1}(n) - C_k(n) = 1 + n(k-1)/2k$$

and so

$$C_k(n) = C_{\lfloor\sqrt{n}\rfloor}(n) + (k-2)(1+n/2) - (n/2)\sum_{i=\lfloor\sqrt{n}\rfloor}^{k-1} i^{-1}.$$

Since $C_{\lfloor\sqrt{n}\rfloor}(n) = \tilde{B}_{\lfloor\sqrt{n}\rfloor}(n)/n$, it follows that

$$\tilde{B}_k(n) = \tilde{B}_{\lfloor\sqrt{n}\rfloor}(n) + (k - \lfloor\sqrt{n}\rfloor)n(n+2)/2 - (n^2/2)\sum_{i=\lfloor\sqrt{n}\rfloor}^{k-1} i^{-1}.$$

If $\lfloor n/2\rfloor + 1 \leqslant k \leqslant n-1$, by Eq.(3.6) it is easy to see that:

$$\tilde{B}_k(n) = \tilde{B}_{\lfloor n/2\rfloor}(n) + (k - \lfloor n/2\rfloor)n^2/2,$$

which concludes the proof. □

3.3 Limitation of this technique

We now show that $a_k^n = \max\{n - k(k-1) - 1, \lceil (n-k)/k\rceil, 1\}$, and so we cannot improve the upper bound $\tilde{B}_k(n)$ on $rt_k(n)$ by improving our estimate of a_k^n.

Lemma 11. *Let n, k integers such that $n \geqslant 2$ and $2 \leqslant k \leqslant n-1$. It holds that:*

$$1 \leqslant a_k^n \leqslant u(n,k) := \begin{cases} n - k(k-1) - 1 & \text{if } n - k(k-1) - 1 \geqslant \lceil (n-k)/k\rceil \\ \lceil (n-k)/k\rceil & \text{otherwise} \end{cases}.$$

Proof. We need to show that for every $n \geqslant 2$ and $2 \leqslant k \leqslant n-1$, there exists a matrix $A \in \mathcal{S}_n^k$ such that $a_k^n(A) = u(n,k)$ (see Definition 18).

We define the matrix $C_i^{m_1 \times m_2}$ as the $m_1 \times m_2$ matrix having all the entries of the i-th column equal to 1 and all the other entries equal to 0, and the matrix $R_i^{m_1 \times m_2}$ as the $m_1 \times m_2$ matrix having all the entries of the i-th row equal to 1

and all the other entries equal to 0. We indicate with $\mathbf{0}^{m_1\times m_2}$ the $m_1 \times m_2$ matrix having all its entries equal to zero and with $\mathbf{I}^{m\times m}$ the $m \times m$ identity matrix. Let $v_k^n = \lceil (n-k)/k \rceil + 1$ and $q = n \mod k$.

Suppose that $n-k(k-1)-1 \geqslant \lceil (n-k)/k \rceil$ and set $\alpha = n-k(k-1)-1-\lceil (n-k)/k \rceil$.Then the following matrix $\hat{A}$ is such that $a_k^n(\hat{A}) = n-k(k-1)-1 = u(n,k)$:

$$\hat{A} = \left[\begin{array}{c|ccccc} C_1^{k\times v_k^n} & R_1^{k\times(k-1)} & R_2^{k\times(k-1)} & \cdots & R_k^{k\times(k-1)} & \\ C_2^{k\times v_k^n} & & & & & \\ \vdots & & \mathbf{0}^{\ (n-k)\times[k(k-1)]} & & & D \\ C_{v_k^n-1}^{k\times v_k^n} & & & & & \\ C_{v_k^n}^{q\times v_k^n} & & & & & \end{array}\right],$$

$$D = \left[\begin{array}{c} \mathbf{0}^{k\times\alpha} \\ \mathbf{I}^{\alpha\times\alpha} \\ \mathbf{0}^{(n-k-\alpha)\times\alpha} \end{array}\right].$$

Indeed by construction, the first column of $\hat{A}$ has exactly k positive entries. The columns of $\hat{A}$ whose support is not contained in $\hat{A}_{*1}$ are the columns $\hat{A}_{*i}$ for $i = 2,\ldots,v_k^n$ and all the columns of D. In total we have $\lceil (n-k)/k \rceil + \alpha = n-k(k-1)-1$ columns, so it holds that $a_k^n(\hat{A}) = n-k(k-1)-1$.

Suppose that $n-k(k-1)-1 \leqslant \lceil (n-k)/k \rceil$. Then the following matrix $\tilde{A}$ is such that $a_k^n(\tilde{A}) = \lceil (n-k)/k \rceil = u(n,k)$:

$$\tilde{A} = \left[\begin{array}{c|cccccc} C_1^{k\times v_k^n} & R_1^{k\times(k-1)} & R_2^{k\times(k-1)} & \cdots & R_{k-1}^{k\times(k-1)} & R_k^{k\times(n-v_k^n-(k-1)^2)} \\ \hline C_2^{k\times v_k^n} & & & & & \\ \vdots & & & \mathbf{0}^{(n-k)\times(n-v_k^n)} & & \\ C_{v_k^n-1}^{k\times v_k^n} & & & & & \\ C_{v_k^n}^{q\times v_k^n} & & & & & \end{array}\right]$$

Indeed by construction, the first column of $\tilde{A}$ has exactly k positive entries and the columns of $\tilde{A}$ whose support is not contained in $\tilde{A}_{*1}$ are the columns $\tilde{A}_{*i}$ for $i = 2,\ldots,v_k^n$. Therefore it holds that $a_k^n(\tilde{A}) = v_k^n - 1 = \lceil (n-k)/k \rceil$. □

3.4 Numerical results

We report here some numerical results that compare the theoretical bound $\tilde{B}_k(n)$ on $rt_k(n)$ of Eq.(3.7) with either the exact k-RT or with an heuristic approximation of the k-RT when the computation of the exact value is not computationally feasible. In Fig. 3.1 we compare our bound with the real k-RT of the primitive sets $\mathcal{M}_{CPR}$ and $\mathcal{M}_K$ reported here below:

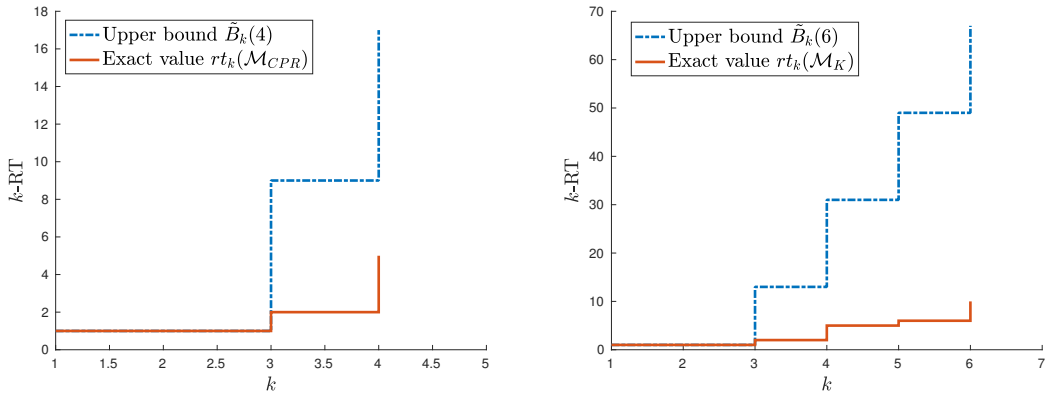
$$\mathcal{M}_{CPR} = \left\{ \begin{pmatrix} 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \end{pmatrix} \right\},$$

$$\mathcal{M}_K = \left\{ \begin{pmatrix} 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}, \begin{pmatrix} 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \right\}.$$

The sets $\mathcal{M}_K$ and $\mathcal{M}_{CPR}$ are primitive sets of matrices that are based on the Kari automaton [20] and the Černý-Piricka-Rozenaurova automaton [47] respectively. We can see that for small values of k, the upper bound is fairly close to the actual value of $rt_k(\mathcal{M})$.

When n is large, computing the k-RT for every $2 \leqslant k \leqslant n$ becomes hard, so we compare our upper bound on the k-RT with a method for approximating it. The *Eppstein heuristic* is a greedy algorithm developed by Eppstein in [13] for approximating the reset threshold of a synchronizing automaton. Given a primitive set $\mathcal{M}$ of binary NZ matrices, we can apply a slightly modified Eppstein heuristic to obtain, for any k, an upper bound on $rt_k(\mathcal{M})$.

In Fig. 3.2 we compare our upper bound with the results of the Eppstein heuristic on the k-RT of the primitive sets with quadratic exponent presented by Catalano and Jungers in [11], Section 4; here we denote these sets by $\mathcal{M}_{C_n}$ where n is the matrix dimension. Finally, Fig. 3.3 compares the evolution of our bound with the results of the Eppstein heuristic on the k-RT of the family $\mathcal{M}_{C_n}$ for fixed $k = 4$ and as n varies. It can be noticed that the bound $\tilde{B}_k(n)$ does not increase

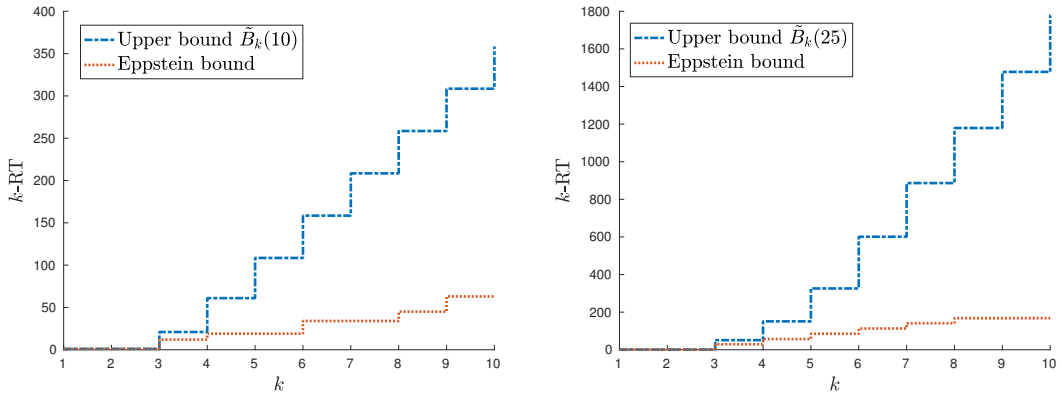


(a) $n = 4$, $\mathcal{M} = \mathcal{M}_{CPR}$ (b) $n = 6$, $\mathcal{M} = \mathcal{M}_K$

Figure 3.1: Comparison between the bound $\tilde{B}_k(n)$, valid for all primitive NZ sets, and $rt_k(\mathcal{M})$ for $\mathcal{M} = \mathcal{M}_{CPR}$ (left) and $\mathcal{M} = \mathcal{M}_K$ (right).

very rapidly as compared to the Eppstein approximation.

In the previous sections, we used a_k^n to find a first upper bound on the k-RT based on Equation (3.7), and we showed that we cannot improve this bound by improving our estimate a_k^n. However, there are other ways to improve the bound, we explore some in the next two sections.



(a) $n = 10$, $\mathcal{M} = \mathcal{M}_{C_{10}}$ (b) $n = 25$, $\mathcal{M} = \mathcal{M}_{C_{25}}$

Figure 3.2: Comparison between $\tilde{B}_k(n)$ and the Eppstein approx. of $rt_k(\mathcal{M})$, for $\mathcal{M} = \mathcal{M}_{C_{10}}$ (left) and $\mathcal{M} = \mathcal{M}_{C_{25}}$ (right). We recall that $\tilde{B}_k(n)$ is a generic bound valid for all primitive NZ sets, while the Eppstein bound is computed on each particular set.

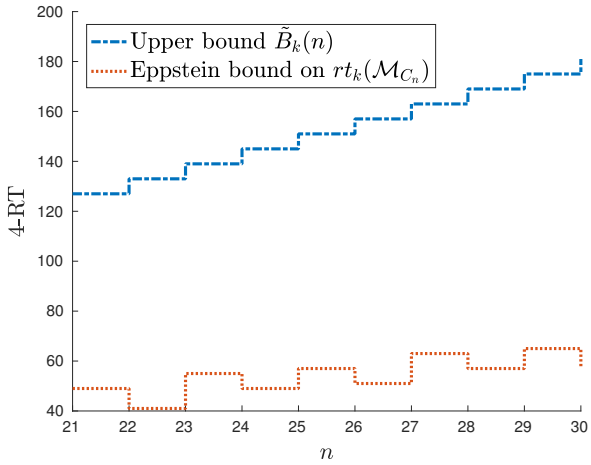


Figure 3.3: Comparison between $\tilde{B}_k(n)$ and the Eppstein approx. of $rt_k(\mathcal{M}_{C_n})$ for $k = 4$. We recall that $\tilde{B}_k(n)$ is a generic bound valid for all primitive NZ sets, while the Eppstein bound is computed on each particular set.

3.5 A first attempt at improving the result

In the previous sections, we obtained an upper bound on $rt_k(n)$ given by the equation (3.7). In this section, we try to improve this bound. In light of the fact that our preceding bound on the k-RT works well for small k, the idea here is to

use products of matrices with columns of weight k for small k to obtain a matrix with large column weight.

In what follows, we will always use the assumption that A has a column of weight k. This is W.L.O.G, since if A has a row of weight k, we can use the same arguments on the matrix set $\mathcal{M}^T$ made of the transpose of all the matrices in $\mathcal{M}$.

Let us now suppose that $A \in S_n^k \bigcap \mathcal{M}^{B_k(n)}$ for some $k \in \{1, \ldots, n-1\}$, and let $p \in \{1, \ldots, n-k\}$. In what follows, we find an upper bound on the length of a product $D \in \mathcal{M}^*$ such that AD has two columns such that the union of their support is of cardinality at least $k+p$. We shall use this bound to obtain a new upper bound on the $(k+p)$-RT.

Let us denote by a_l the vector representing the l-th column of the matrix A. Let's fix $l \in \{1, ..., n\}$ such that $a_l \in \mathcal{C}_A$. We note $S = \text{supp}\{a_l\} = \{r_1^S, r_2^S, ..., r_k^S\}$. In what follows, let us define the set $\tilde{S} = \{r_1^{\tilde{S}}, r_2^{\tilde{S}}, ..., r_{n-k}^{\tilde{S}}\} = \{1, ..., n\} \backslash S$. Since A is NZ, for all $i \in \{1, ..., n-k\}$ there exist $c_i^{\tilde{S}} \in \{1, \ldots, n\}$ such that $A_{r_i^{\tilde{S}}, c_i^{\tilde{S}}} > 0$. We therefore define $\tilde{T} = \{c_1^{\tilde{S}}, ..., c_{n-k}^{\tilde{S}}\}$[1], the set of the indices of the columns of A that have positive entries in the rows indexed by $\tilde{S}$.

Let g be a column of an arbitrary NZ matrix G of size $n \times n$. We refer to $|\text{supp}\{g\} \cap \tilde{T}|$ as the $\tilde{T}$-weight of g. For all positive integers p such that $1 \leqslant p \leqslant |\tilde{T}|$ we denote by C_G^p the set of all columns of G of $\tilde{T}$-weight at least p. Finally, letting $\mathcal{S}_p$ be the set of all $n \times n$, $\mathcal{NZ}$ matrices G such that $C_G^p \neq \varnothing$, we define d_p^* as the smallest integer satisfying $\mathcal{M}^{d_p^*} \bigcap \mathcal{S}_p \neq \varnothing$.

Remark 4. *It should be understood that $S, \tilde{S}, \tilde{T}, \mathcal{S}_p$ and d_p^* actually depend on the choice of A, k and l. But because A, k and l are fixed, we omit writing this dependence explicitly in order to lighten the notations.*

We show that any matrix $D \in \mathcal{S}_p$ satisfies the condition that AD has two columns such that the union of their support is of cardinality at least $k+p$.

Lemma 12. *Let $D \in \mathcal{S}_p, t \in C_D^p$ such that for all $j \in \{1, ..., p\}$, we have $D_{c_j^{\tilde{S}}, t} > 0$. Then there exists $u \in \{1, ..., n\}$ such that for all $i \in \{1, ..., k\}$, $j \in \{1, ..., p\}$ we have $(AD)_{r_i^S, u} > 0$ and $(AD)_{r_j^{\tilde{S}}, t} > 0$.*

[1]We note that these elements are not, in general, distinct, and so $\tilde{T}$ may have cardinality less than $n-k$.

Proof. Since D is NZ, there exists $u \in \{1, ..., n\}$ such that $D_{l,u} > 0$. Then, for all $i \in \{1, ..., k\}$, we have :

$$(AD)_{r_i^S,u} = \sum_{k=1}^{n} A_{r_i^S,k} D_{k,u}.$$

Since A and D are nonnegative, $(AD)_{r_i^S,u} > 0$ iff there exists $k \in \{1, ..., n\}$ such that $A_{r_i^S,k} > 0$ and $D_{k,u} > 0$. This condition is fulfilled by $k = l$.

Similarly, for all $j = 1, ..., p$, we have $(AD)_{r_j^{\tilde{S}},t} > 0$ since $A_{r_j^{\tilde{S}},k} > 0$ and $D_{k,t} > 0$ for $k = c_j^{\tilde{S}}$. □

It follows that we can bound the $(k+p)$-RT using the following lemma.

Lemma 13. *Letting d_p^* be as defined above, we have an upper bound on $rt_{k+p}(n)$ given by $B_k(n) + d_p^* + w$, with $w = \frac{n(n-1)}{2}$.*

Proof. By hypothesis, $A \in \mathcal{M}^{B_k(n)}$. This implies that there exists $D \in \mathcal{S}_p$ such that $AD \in \mathcal{M}^{B_k(n)+d_p^*}$. By Lemma 12, there exists $u \in \{1, ..., n\}$ such that for all $i \in \{1, ..., k\}$, $j \in \{1, ..., p\}$, we have $(AD)_{r_i^S,u} > 0$ and $(AD)_{r_j^{\tilde{S}},t} > 0$. By construction of the pair digraph of the matrix set M, the shortest path between the node (u, t) and a singleton is of length at most w. Using Lemma 6, we can conclude the proof. □

Trivially, the bound we derived above stays true if we replace the value of d_p^* by one of its upper bounds. All that remains to be done now is to obtain an upper bound on d_p^*, so we can use Lemma 13 to bound the $(k+p)$-RT. For this purpose, we define the following set Σ_p of matrices in $\mathcal{S}_p$ of maximal $\tilde{T}$-weight p.

$$\Sigma_p = \left\{ G \in \mathcal{S}_p \;\middle|\; \text{for every column } g \text{ of } G, |\text{supp}\{g\} \cap \tilde{T}| \leqslant p \right\}$$

For every $G = [g_1, \ldots, g_n] \in \Sigma_p$ we define $\tilde{a}_p(G) = \min_{g \in C_G^p} |\{i : \text{supp}\{g_i\} \cap \tilde{T} \nsubseteq \text{supp}\{g\} \cap \tilde{T}\}|$. We then define $\tilde{a}_p := \min_{G \in \Sigma_p} \tilde{a}_p(G)$. This allows us to bound d_p^* as a function of $\tilde{a}_p$ in the same way that we bounded the k-RT as function of a_k^n in the previous section. We obtain the following recurrence.

Lemma 14. *Let $1 \leqslant p \leqslant |\tilde{T}|$. Then $d_p^* \leqslant B_k^p(n)$, where :*

$$\begin{cases} B_k^1(n) = 1 \\ B_k^{p+1}(n) = B_k^p(n) + \frac{n}{2}(1 + n - \tilde{a}_p) \quad \textit{for } 1 \leqslant p \leqslant |\tilde{T}| - 1. \end{cases} \tag{3.8}$$

Proof. Similar to Theorem 6. □

Lemma 15. *Let p and k be as defined above. We have $\tilde{a}_p \geqslant \frac{a_k^n - p}{p}$.*

Proof. Using the same reasoning as Lemma 9, we have that $\tilde{a}_p \geqslant \frac{|\tilde{T}|-p}{p}$. Observing that $|\tilde{T}| = a_k^n(A) \geqslant a_k^n$, we can conclude the proof. □

Using the preceding lemma we can finally calculate an explicit bound on d_p^* as a function of n, k and p. The recursion (3.8) with $\tilde{a}_p$ replaced by the preceding lower bound:

$$\begin{cases} \frac{n-k(k-1)-1-p}{p} & \text{if } k \leqslant \lfloor\sqrt{n}\rfloor \\ \frac{\frac{n-k}{k}-p}{p} & \text{if } \lfloor\sqrt{n}\rfloor < k \leqslant \lfloor\frac{n}{2}\rfloor \end{cases}.$$

gives, if $k \leqslant \lfloor\sqrt{n}\rfloor$, the following expression:

$$\tilde{B}_k^{p+1}(n) = \begin{cases} 1 & \text{if } p = 0 \\ \tilde{B}_k^p(n) + \frac{n}{2}\left(1 + n - \frac{n-k(k-1)-1-p}{p}\right) & \text{if } 1 \leqslant p \leqslant |\tilde{T}| - 1. \end{cases} \tag{3.9}$$

and if $\lfloor\sqrt{n}\rfloor < k \leqslant \lfloor\frac{n}{2}\rfloor$, it gives :

$$\tilde{B}_k^{p+1}(n) = \begin{cases} 1 & \text{if } p = 0 \\ \tilde{B}_k^p(n) + \frac{n}{2}\left(n(1 - \frac{1}{kp}) + 2 + \frac{1}{p}\right) & \text{if } 1 \leqslant p \leqslant |\tilde{T}| - 1. \end{cases} \tag{3.10}$$

Solving these recurrence relations, we find an explicit bound on d_p^* given by the following functions.

Theorem 8. *Equation* (3.9) *is fulfilled by the following function:*

$$\tilde{B}_k^p(n) = \frac{n^2}{2}\left(p - 1 - \sum_{q=1}^{p-1}\frac{1}{q}\right) + \frac{n}{2}\left(2(p-1) + \sum_{q=1}^{p-1}\frac{1}{q}(k^2 - k + 1)\right) + 1.$$

And the recurrence (3.10) *is solved with the following function:*

$$\tilde{B}_k^p(n) = n^2\left(\frac{(p-1)}{2} - \frac{1}{2k}\sum_{q=1}^{p-1}\frac{1}{q}\right) + n\left(p - 1 + \frac{1}{2}\sum_{q=1}^{p-1}\frac{1}{q}\right) + 1$$

Proof. For the recurrence (3.9), let $\tilde{C}_k^p(n) = \tilde{B}_k^p(n)/n$. We have:

$$\tilde{C}_k^{p+1}(n) - \tilde{C}_k^p(n) = \frac{n}{2} - \frac{n}{2p} + \frac{k^2}{2p} - \frac{k}{2p} + \frac{1}{2p} + 1.$$

From the above equation, we can deduce:

$$\tilde{C}_k^{p+s}(n) - \tilde{C}_k^p(n) = \frac{sn}{2} + \frac{1}{2}\sum_{q=0}^{s-1} \frac{1}{p+q}(k^2 - k - n + 1) + s.$$

And then, we have:

$$\tilde{B}_k^p(n) = \frac{n^2}{2}\left(p - 1 - \sum_{q=1}^{p-1} \frac{1}{q}\right) + \frac{n}{2}\left(2(p-1) + \sum_{q=1}^{p-1} \frac{1}{q}(k^2 - k + 1)\right) + \tilde{B}_k^1(n)$$

Then the desired result follows from the initial condition $\tilde{B}_k^1(n) = 1$.

For the recurrence (3.10), let again $\tilde{C}_k^p(n) = \tilde{B}_k^p(n)/n$. Then it is clear that the following equation holds:

$$\tilde{C}_k^{p+1}(n) - \tilde{C}_k^p(n) = \frac{n}{2} - \frac{n}{2kp} + 1 + \frac{1}{2p}.$$

From this equation, we can obtain:

$$\tilde{C}_k^{p+s}(n) - \tilde{C}_k^p(n) = \frac{sn}{2} - \frac{n}{2k}\sum_{q=0}^{s-1} \frac{1}{p+q} + s + \sum_{q=0}^{s-1} \frac{1}{2(p+q)}.$$

And it easily follows that:

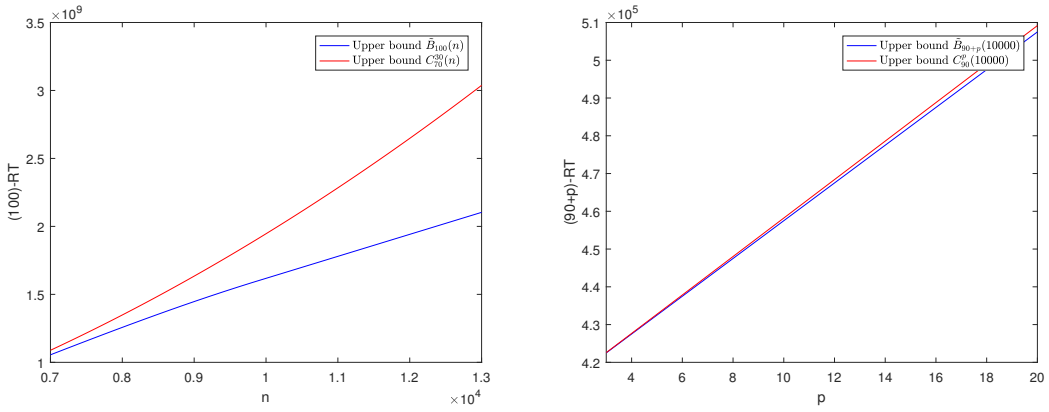
$$\tilde{B}_k^p(n) = n^2\left(\frac{(p-1)}{2} - \frac{1}{2k}\sum_{q=1}^{p-1} \frac{1}{q}\right) + n\left(p - 1 + \frac{1}{2}\sum_{q=1}^{p-1} \frac{1}{q}\right) + \tilde{B}_k^1(n)$$

Now the desired result follows from the initial condition $\tilde{B}_k^1(n) = 1$. □

Finally by combining the results of Lemma 13, Lemma 14 and Theorem 8, we obtain that $rt_{k+p}(n) \leqslant \tilde{B}_k(n) + \tilde{B}_k^p(n) + w = C_n^p(n)$, with $w = \frac{n(n-1)}{2}$.

In the Figure 3.4 and Figure 3.5 we plot the values of $\tilde{B}_{k+p}(n)$ and $C_k^p(n)$, two upper bounds on rt_{k+p}.

In Figure 3.4, on the left, we use the values of $k = 70$ and $p = 30$, and we plot those upper bound for some values of n. On the right, we use the values of



(a) $k = 70$, $p = 30$ (b) $n = 10000$, $k = 90$

Figure 3.4: Comparison between $\tilde{B}_k(n)$ and $C_k^p(n)$.

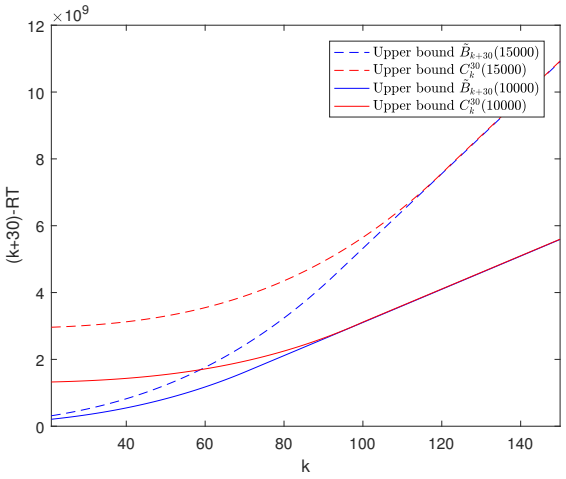


Figure 3.5: Comparison between $\tilde{B}_k(n)$ and $C_k^p(n)$, with $n \in \{10000, 15000\}$ and $p = 30$.

$n = 10000$ and $k = 90$, and we graph the upper bounds for some values of p.

On the Figure 3.5, we use the values of $n \in \{10000, 15000\}$ and $p = 30$, and we plot those upper bounds for some values of k.

As we can see on those figures, this technique is not very efficient. Indeed, for each values of $p, k, n \in \mathcal{N}_0$ such that $1 \leqslant p \leqslant k \leqslant n$ and $p + k \leqslant n$, the upper

bound $C_k^p(n)$ seems to be bigger than $\tilde{B}_{k+p}(n)$.

However, in the previous argument, we did not take into account the fact that if $\tilde{T}$ has a cardinality less than $n-k$, and if D has a $\tilde{T}$-weight of p, it is possible that the matrix AD has two columns such that the union of their support is of cardinality strictly more than $k+p$. For example, it should be noticed that if D has a column of $\tilde{T}$-weight of $|\tilde{T}|$, then AD has two columns such that the union of their supports has cardinality n. The previous argument could therefore possibly be improved by leveraging this fact.

3.6 An improvement of the result

In this section, we present another method for improving the bound we found in Theorem 7. We first begin this section by an intuitive explanation of what we will do in the rest of this section. This explanation is not rigorous, its only goal is the help the understanding.

Intuitively, in order to obtain the bound given by Equation (3.7), we supposed that for each k, we had at least a_k^n columns that we could "merge" with a fixed column of weight k to obtain a column of weight at least $k+1$ (using Lemma 6).

But in fact, we did not take into account the weight of the column we merged with the one of weight k. Indeed, maybe we can merge the column of weight k with another column such that the resulting column is of weight $k+p > k+1$.

In what follows, we improve the previous argument by considering that if $k < n$, there always exists some integer p such that if we merge the column of weight k with any of the other columns we get a resulting column of weight at most $k+p$, while there exists a column such that if we merge it with the one of weight k we obtain a column of weight $k+p$. Then, in this case, we consider the two following techniques:

1. Merging the column of weight k with the "fixed" column such that the resulting column is of weight $k+p$
2. Merging the column of weight k with any columns that improve the weight of the column (which means such that the resulting column is of weight at least $k+1$)

Then, starting from a matrix $A \in \mathcal{M}^l$ that has a column of weight k, suppose that we can find a matrix $M_1 \in \mathcal{M}^{m_1}$ and $M_2 \in \mathcal{M}^{m_2}$, for "small" $m_1, m_2 \in \mathbb{N}$, such

that M_1 and M_2 are the matrices we obtain by applying the first technique and the second technique respectively (which means that the matrix AM_1 has a column of weight $k+p$, and AM_2 has a column of weight at least $k+1$). Then we have a bound on the shortest product of matrices from $\mathcal{M}$ that gives a matrix with a column of weight $k+p$ (resp. at least $k+1$) given by $l+m_1$ (resp. $l+m_2$). We can then reiterate the argument with the new weight of the column we have obtained.

Then, for every $k \leqslant k_f \leqslant n$, suppose that starting from a product $A \in \mathcal{M}^l$ that has a column of weight k, we want to end with a product $B \in \mathcal{M}^*$ with a column of weight at least k_f. Then any combination of techniques 1 or 2 that produces at the end a column of weight at least k_f gives a matrix $M \in \mathcal{M}^*$ such that the length of the matrix AM is an upper bound on the value of the length of the smallest product of matrices from $\mathcal{M}$ that gives a column of weight k_f. We can then choose the combination that gives the smallest upper bound. However, since at each stage the true value of p is unknown, we have to consider the worst case scenario. This is further formalized below.

Let $1 \leqslant k \leqslant n-1$, and $A \in S_n^k$ and fix $l \in \{1,...,n\}$ such that $a_l \in \mathcal{C}_A$. We note $S_l = \text{supp}\{a_l\}$. Let $\tilde{S}_l = \{1,...,n\} \backslash S_l$. We call the $\tilde{S}_l$-weight of a column c of A the number y such that $y = |\text{supp}\{c\} \cap \tilde{S}_l|$.

Let $1 \leqslant p \leqslant \min(k, n-k)$ and $S_{n,p}^k$ be the set of all matrices $A \in S_n^k$ such that for all $a_m \in \mathcal{C}_A$, all other columns of A have an $\tilde{S}_m$-weight of at most p, and there exists at least one column $a_l \in \mathcal{C}_A$ such that there exists a column of A of $\tilde{S}_l$-weight p. Trivially, we have:

$$\bigcup_{p=1}^{\min(k,n-k)} S_{n,p}^k = S_n^k. \tag{3.11}$$

For every $A \in S_{n,p}^k$, let C_A^p be the set of all columns $a_l \in \mathcal{C}_A$ such that there exists a column of A of $\tilde{S}_l$-weight p.

For every $A \in S_{n,p}^k$, we define $a_{k,p}^n(A) := \min_{a \in C_A^p} |\{i : \text{supp}(A_{*i}) \nsubseteq \text{supp}(A_{*a})\}|$ and $a_{k,p}^n := \min_{A \in S_{n,p}^k} a_{k,p}^n(A)$.

Remark 5. *It is clear that $a_{k,p}^n \geqslant a_k^n$, so every lower bound on a_k^n still holds for $a_{k,p}^n$.*

Lemma 16. *Let $a_{k,p}^n$ be as defined above. Then it follows that $a_{k,p}^n \geqslant \frac{n-k}{p}$.*

Proof. Similar to Lemma 9. □

Let's fix k_f such that $k < k_f \leqslant n$. Let us also define :

$$\mathcal{O}_k^{k_f}(n) = \max_{A \in S_n^k} \min\{x : \exists D \in \mathcal{M}^x \text{ such that } AD \text{ has a column of weight at least } k_f\}.$$

Let $B_k(n)$ be as defined in Theorem 6. It is clear that :

$$rt_{k_f}(n) \leqslant B_k(n) + \mathcal{O}_k^{k_f}(n) \tag{3.12}$$

In what follows, we will find a downward recurrence relation in order to compute an upper bound on the value of $\mathcal{O}_k^{k_f}(n)$. Let $1 \leqslant p \leqslant \min(k, n-k)$ and further define:

$$\tilde{\mathcal{O}}_{k,p}^{k_f}(n) = \max_{A \in S_{n,p}^k} \min\{x : \exists D \in \mathcal{M}^x \text{ such that } AD \text{ has a column of weight at least } k_f\}.$$

Then, it follows from (3.11) that:

$$\mathcal{O}_k^{k_f}(n) = \max_{p \in \{1,\ldots,\min(k,n-k)\}} \tilde{\mathcal{O}}_{k,p}^{k_f}(n) \tag{3.13}$$

Lemma 17. *Let $1 \leqslant p \leqslant \min(k, n-k)$ and $k < k_f \leqslant n$. Then we have:*

$$\tilde{\mathcal{O}}_{k,p}^{k_f}(n) \leqslant \mathcal{O}_{k+p}^{k_f}(n) + \frac{n}{2}(n-1).$$

Proof. We need to prove that for any matrix $A \in S_{k,p}^n$, there exists a matrix $B \in \mathcal{M}^{\frac{n}{2}(n-1)}$ such that $AB \in S_{k+p}^n$. Let $a_m \in C_A^p$, such that there exists a column a_l of $\tilde{S}_l$- weight p. By construction of the pair digraph of the matrix set, we can find an upper bound on the length of the shortest path between the node (m, l) and a singleton. The shortest path from any node to any other node is of length at most $n(n+1)/2 - 1$. Removing from this path all the extra singletons, and using Lemma 6 we get the desired result. □

Lemma 18. *Let $1 \leqslant p \leqslant \min(k, n-k)$ and $k < k_f \leqslant n$. Then we have that:*

$$\tilde{\mathcal{O}}_{k,p}^{k_f}(n) \leqslant \mathcal{O}_{k+1}^{k_f}(n) + \frac{n}{2}(1 + n - a_{k,p}^n).$$

Proof. We need to prove that for any matrix $A \in S_{k,p}^n$, there exists a matrix $B \in \mathcal{M}^{\frac{n}{2}(1+n-a_{k,p}^n)}$ such that $AB \in S_{k+1}^n$. Since we have the assumption that $A \in S_{k,p}^n$, we can adapt the proof of Theorem 6 by replacing a_k^n by $a_{k,p}^n$. □

Theorem 9. *Let $1 \leqslant p \leqslant \min(k, n-k)$ and $k < k_f \leqslant n$. Let $\tilde{a}_{k,p}^n = \max\{n - k(k-1) - 1, \frac{n-k}{p}, 1\}$. Then we have :*

$$\tilde{\mathcal{O}}_{k,p}^{k_f}(n) \leqslant \min\{\mathcal{O}_{k+p}^{k_f}(n) + \frac{n}{2}(n-1); \mathcal{O}_{k+1}^{k_f}(n) + \frac{n}{2}(1 + n - \tilde{a}_{k,p}^n)\}$$

Proof. Trivial in view of Remark 5, Lemma 16, 17 and 18. □

Using the relation (3.13) and Theorem 9, we have the following inequality:

$$\mathcal{O}_k^{k_f}(n) \leqslant \max_{p\in[\min(k,n-k)]} \min\{\mathcal{O}_{k+p}^{k_f}(n) + \frac{n}{2}(n-1); \mathcal{O}_{k+1}^{k_f}(n) + \frac{n}{2}(1+n-\tilde{a}_{k,p}^n)\}$$

Finally we obtain the following bound:

Theorem 10. *Let $n \geqslant 1$ be an integer and $\tilde{a}_{k,p}^n = \max\{n-k(k-1)-1, \frac{n-k}{p}, 1\}$. Then we can define a function $U_k^{k_f}(n)$ such that for all $1 \leqslant k, k_f \leqslant n$, $U_k^{k_f}(n) \geqslant \mathcal{O}_k^{k_f}(n)$ by the following relation :*

$$\begin{cases} U_k^{k_f}(n) = 0 & \text{if } k \geqslant k_f \\ U_k^{k_f}(n) = \max\limits_{p\in[\min(k,n-k)]} \min\{U_{k+p}^{k_f}(n) + \frac{n}{2}(n-1); U_{k+1}^{k_f}(n) + \frac{n}{2}(1+n-\tilde{a}_{k,p}^n)\} & \text{if } k < k_f \end{cases} . \tag{3.14}$$

For a fixed value of k_f and n, the value of $U_k^{k_f}(n)$ can be computed for all value of k using dynamic programming.

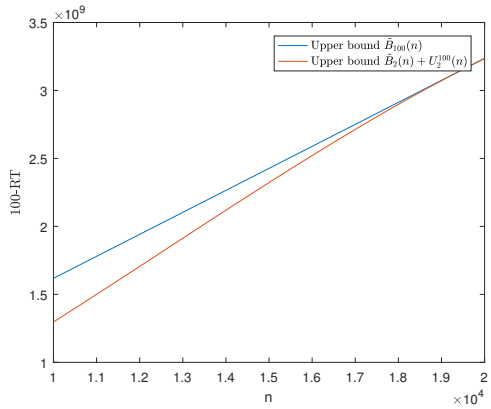
Theorem 11. *Let $1 \leqslant k_f \leqslant n$ be integers. Let $\tilde{B}_k(n)$ be as in equation (3.7), and $U_k^{k_f}(n)$ be as in equation (3.14). Then, for all $1 \leqslant k \leqslant k_f$, we have an upper bound on $rt_{k_f}(n)$ given by :*

$$rt_{k_f}(n) \leqslant \tilde{B}_k(n) + U_k^{k_f}(n) \tag{3.15}$$

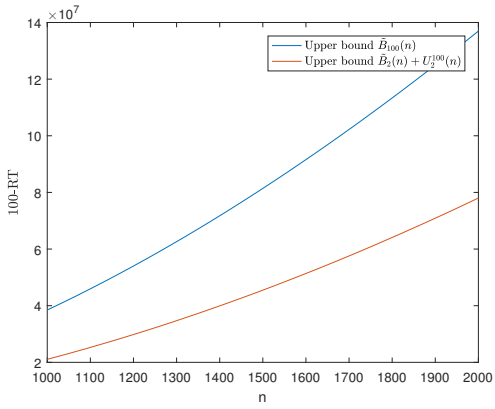
We know from equation (3.7) that $\tilde{B}_k(n)$ grows as a linear function of n if $k \leqslant \lfloor\sqrt{n}\rfloor$, and increases as a quadratic function of n otherwise. Figure 3.6 shows the difference between the upper bound for the $rt_{k_f}(n)$ we get using equation (3.15) with $k = k_f$ (which is the same upper bound on the k_f-RT we had from equation (3.7)), and the one we get with $k = 2$, for multiple values of n. In the plot on the left, we choose $k_f = 100$ and the values of n such that $k_f \in \{1, ..., \lfloor\sqrt{n}\rfloor\}$ for all value of n. In the plot on the right, we choose $k_f = 100$ and the values of n such that $k_f \in \{\lfloor\sqrt{n}\rfloor + 1, ..., \lfloor n/2\rfloor\}$ for all value of n.

Figure 3.7 also compares the value of $\tilde{B}_k(n) + U_k^{k_f}(n)$ in the case where $k = k_f$, and in the case where $k = 2$, for multiple values of n. But this time, in the plot on the left, we choose k_f and n such that $k_f \in \{\lfloor n/2\rfloor + 1, ..., n\}$, and in the plot on the right, we choose $k_f = n$.

Figure 3.8 shows the difference between $\tilde{B}_k(n) + U_k^{k_f}(n)$ in the case where $k = k_f$, and in the case where $k = 2$, for multiple values of k, and with $n = 1000$.

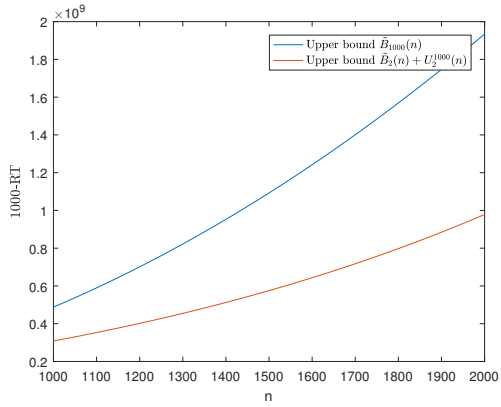


(a) $k_f = 100$

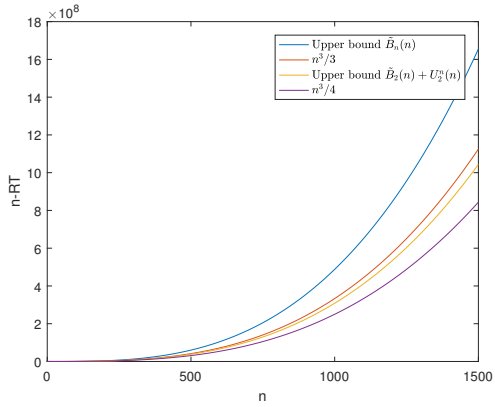


(b) $k_f = 100$

Figure 3.6: Comparison of $\tilde{B}_k(n) + U_k^{k_f}(n)$ in the case where $k = k_f$, and in the case where $k = 2$.



(a) $k_f = 1000$



(b) $k_f = n$

Figure 3.7: Comparison of $\tilde{B}_k(n) + U_k^{k_f}(n)$ in the case where $k = k_f$, and in the case where $k = 2$.

This technique is interesting, as it happens to give a better upper bound on $rt_{k_f}(n)$ than the one we found in equation (3.7) for all values of $2 \leqslant k < k_f$. Moreover, for all $2 \leqslant k_f \leqslant n$, the best upper bound on $rt_{k_f}(n)$ was always observed with $k = 2$. However, obtaining a simple analytic expression for $U_k^{k_f}$ seems to be difficult. It could be simpler to find an upper bound on $U_k^{k_f}$ close enough to the true value of the function, in order to improve the bound (3.7). This is left for future work.

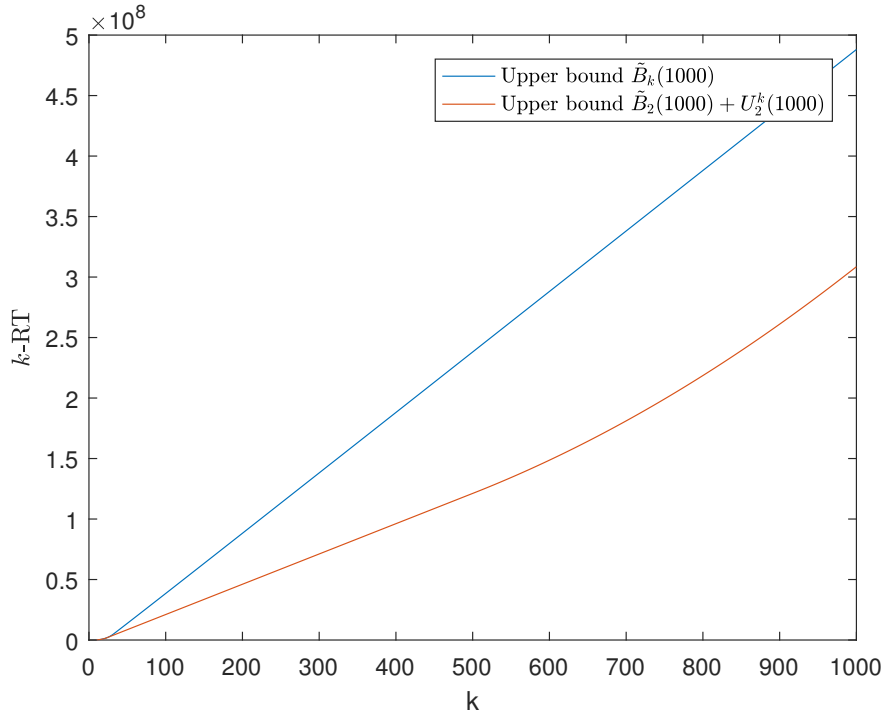


Figure 3.8: Comparison of $\tilde{B}_k(n) + U_k^{k_f}(n)$ in the case where $k = k_f$, and in the case where $k = 2$, for multiple values of k, and with $n = 1000$.

Chapter 4

Analysis of a randomized algorithm for generating extremal automata

In this section we study an algorithm, henceforth referred to as $\mathcal{C}$, proposed by Catalano and Jungers in [11] for randomly generating slowly synchronizing automata with high probability. The value of their contribution lies in the fact that the algorithm always provides n-state, m-letter *minimally synchronizing automata*, for the chosen values of n and m. The definition of minimally synchronizing automata is as follows.

Definition 19. *A synchronizing automaton is* minimally synchronizing *if the elimination of any letter from the set of letters results in an automaton that is not synchronizing.*

We argue that this high probability of finding slowly synchronizing automata is due to the fact that the algorithm produces automata from a very limited class of automata with high probability. Indeed we provide theoretical upper bounds on the probability that the automata generated by this algorithm will have large cycle lengths, and show that these can be expected to decrease as a function of m and n. We also provide results of numerical simulations of our implementation of $\mathcal{C}$ that further confirm this hypothesis.

4.1 The algorithm $\mathcal{C}$

4.1.1 Summary of the algorithm

The idea behind $\mathcal{C}$ is to construct minimally synchronizing automata by first constructing *minimally primitive perturbed permutation sets*, the definition of which

is given below.

Definition 20. *A* perturbed permutation set *is a set of permutation matrices where one of the matrices has had one of its zero entries changed to a 1.*

Definition 21. *A set of NZ matrices $\mathcal{M}$ is* minimally primitive *iff it is primitive, and every proper subset of $\mathcal{M}$ is not.*

To do this, the algorithm constructs a perturbed permutation set $\mathcal{M} = \{M_1, \ldots, M_m\}$, while using the Protasov-Voynov theorem (see Theorem 3) to ensure that every subset of $m-1$ matrices from $\mathcal{M}$ is not primitive.

More specifically, the algorithm takes as input a *seed vector* $\mathbf{p} = [q_1, \ldots, q_m]$ containing m prime numbers and initializes the matrix set $\mathcal{M}$ to contain m all-ones matrices[1] of dimension $n \times n$, where $n = q_1 \ldots q_j$. Then for any $j \in \{1, \ldots, m\}$, letting $\mathcal{M}_j = \mathcal{M} \backslash \{M_j\}$, the algorithm imposes that every matrix from $\mathcal{M}_j$ acts as a permutation[2] over a q_j-partition Ω^j of $\{1, \ldots, n\}$ by setting certain entries to zero. The definition of a *q-partition* is given below.

Definition 22. *A q-partition of a set S is a partition of S into q subsets of equal cardinality.*

We also define the notion of a matrix with a *q-permutation structure*.

Definition 23. *An $n \times n$ matrix A has a q-permutation structure iff it acts as a permutation over some q-partition of $\{1, \ldots, n\}$. We say that A has a block permutation structure iff it has a q-permutation structure for some $q \in \mathbb{N}$.*

However, since we ultimately want the algorithm to produce a perturbed permutation set, we have to ensure each time we impose a new q-permutation structure on a matrix that the matrix still *dominates a permutation*. This notion is defined below, as is the notion of *compatibility*.

Definition 24. *A matrix M dominates a permutation iff there exists a permutation matrix P such that $A \geqslant P$ (entrywise).*

Remark 6. *Let M be a $n \times n$ matrix, and $I, J \subseteq \{1, \ldots, n\}$. We introduce the notation $M_{I,J}$ for the submatrix of M consisting of the rows indexed by I and the columns indexed by J.*

[1] An all-ones matrix is a matrix such that all its entries are equal to 1.

[2] The method for choosing the permutation depends on the parameter `met` which takes either the value 2 or 3. (To make comparison with the original article easier for the reader we use the same numbering of methods used in the original article, and we consider only the methods 2 and 3, this explains the absence of a `met`=1 option.). The methods 2 and 3 are explained in the next subsection.

Definition 25. *For all $q \in \mathbb{N}_0$, we denote by S_q the symmetric group of order q.*

Definition 26 ([11], Definition 10)**.** *Given an $n \times n$ matrix M and a q-partition of $\{1, \ldots, n\}$, $\Omega = \{\Omega_1, \ldots, \Omega_q\}$, we say that a permutation $\sigma \in S_q$ is compatible with M and Ω if for all $i = 1, \ldots, q$, there exists a permutation matrix Q_i such that $M_{\Omega_i, \Omega_{\sigma(i)}} \geqslant Q_i$. We say that a q-partition Ω is compatible with M if there exists a permutation $\sigma \in S_q$ such that σ is compatible with M and Ω.*

Clearly if we want the algorithm to produce perturbed permutation matrix sets, we need to make sure at each step, that for each matrix M_i, the algorithm only imposes a block permutation structure over a partition that is compatible with M_i. In practice, the algorithm randomly chooses partitions (uniformly from the set of all partitions), rejecting the incompatible ones until a compatible one is found, over which a block permutation structure is then imposed.

Once the matrix set $\mathcal{M}$ satisfying these constraints has been constructed, each matrix M_i in $\mathcal{M}$ is replaced by a permutation matrix dominated by M_i. The choice of the extracted permutations is once again determined by the choice of the parameter `met`. $\mathcal{M}$ is now a set of permutation matrices, all that remains to be done now is to select a matrix in $\mathcal{M}$ uniformly at random, and to change one of the zero entries to a 1 while taking care to preserve the block permutation structures of the matrix.

Finally, the perturbed permutation matrix set $\mathcal{M}$ produced by $\mathcal{C}$ is checked for primitivity[3], if it is not primitive, it is discarded, otherwise we construct the associated automaton $Aut(\mathcal{M})$ (See Definition 10). Now, by Proposition 9 in [11], if the automaton $\overline{Aut}(\mathcal{M})$ (this is $Aut(\mathcal{M})$ with the permutation matrix to which the 1 was added removed) is synchronizing, then it is minimally synchronizing, otherwise $Aut(\mathcal{M})$ is. So by checking synchronizability of $\overline{Aut}(\mathcal{M})$ using the pair digraph criterion (see Lemma 5 and Remark 3), we obtain a minimally synchronizing automaton.

4.1.2 Two methods of finding a permutation matrix dominated by a given matrix

Given a matrix M, the algorithm $\mathcal{C}$ needs to be able to extract a permutation matrix P such that M dominates P. However, there are often multiple permutation matrices P satisfying this condition. It shall be shown in the following section that the choice of P has a significant impact on the probability distribution of the cycle lengths of the permutation matrices in the perturbed permutation set

[3]For this, the Protasov Voynov algorithm is used, see [34].

produced by the algorithm. This choice is determined by the value of the parameter `met` passed as an argument to the algorithm $\mathcal{C}$. It can be shown that if `met` = 2, the permutation matrix P is chosen randomly following a uniform probability distribution over the set of permutation matrices dominated by M. In the case where `met` = 3, the choice is deterministic.

The pseudocode for the `ExtractPerm` function (see [11]) used by $\mathcal{C}$ to extract a dominated permutation matrix P from a given matrix M is provided below.

Algorithm 1: Pseudo-code for permutation extraction algorithm

Input: An $n \times n$ matrix M.
Output: A permutation matrix P such that M dominates P.

$B \longleftarrow 0$;
P initialized to a zero matrix of dimensions $n \times n$;
$i \longleftarrow 0$;

while $B = 0$ *and* $i \leqslant n$ **do**

  $i \longleftarrow i + 1$;
  $j \longleftarrow$ non-blacklisted row or col of min weight in M ;

  if *supp(j)* $= \varnothing$ **then**
    $B \longleftarrow 1$;
  end
  else
    Pick a positive entry in supp(j) ; `/* See Remark 7 */`
    Set corresponding entry in P to 1 ;
    Blacklist row and col corresponding to this entry ;
  end

end

if $B = 1$ **then**
  M does not dominate a permutation
end

return P;

Remark 7. *(See pseudocode above). The choice of this entry is dictated by* `met`*. If* `met` *= 2, then this entry is chosen randomly following a uniform distribution over supp(j). However if* `met` *= 3, the first entry in lexicographical order is chosen.*

4.1.3 Pseudocode for algorithm $\mathcal{C}$

The pseudocode for the part of algorithm $\mathcal{C}$ relevant to our analysis is given below (see [11] p.9 for further details). To obtain the numerical results of this chapter we implemented this algorithm in `Matlab`.

Algorithm 2: Pseudo-code for algorithm $\mathcal{C}$

Input:

- A seed vector of prime numbers $\mathbf{p} = [q_1, \ldots, q_m]$,
- a value of parameter `met` $= 2$ or 3,
- a natural number T, the max number of attempts to find a compatible partition.

Output: A perturbed permutation matrix set $\mathcal{M} = \{M_1, \ldots, M_m\}$ that, if primitive, is minimally primitive.

Initialize $M_1, \ldots, M_m$ to all-ones matrices ;

Continued on next page...

---

for $j = 1, \ldots, m$ **do**

$t \longleftarrow 0$; $B \longleftarrow 0$;

while $t < T$ *and* $B = 0$ **do**

$t \longleftarrow t + 1$; $B \longleftarrow 1$; Choose a q_j-partition Ω^j ;

for $k = 1, \ldots, m$ *and* $k \neq j$ *and* $B = 1$ **do**

if Ω^j *is compatible with* M_k **then**

$A_k \longleftarrow M_k$ with a compatible q_j-permutation structure imposed over Ω^j, the choice of permutation being decided by `met` ;
$B \longleftarrow 1$;

end
else
$B \longleftarrow 0$;
end

end

end

if $t = T$ *and* $B = 0$ **then**
No compatible partition found after T tries ;
end
else
for $k = 1, \ldots, m$ *and* $k \neq j$ **do**
$M_k \longleftarrow A_k$;
end
end

end

for $k = 1 \ldots, m$ **do**
$M_k \longleftarrow$ A permutation matrix dominated by M_k, chosen according to `met` ;
end

Select a matrix in $\mathcal{M}$ and change a 0 entry to a 1 while respecting the block-permutation structures ;
return $\mathcal{M}$;

4.2 Bounding the max cycle lengths as a function of the prime numbers used

We are interested in analyzing how the distribution of cycle lengths in the automata generated by $\mathcal{C}$ is affected by the choice of $\mathbf{p} = [q_1, \ldots, q_m]$. We shall see that for both possible choices of the parameter `met` we can expect the cycle lengths to be smaller relative to the state space dimension n for larger values of $q^* = \max\{q_1, \ldots, q_m\}$.

Let π be a permutation in the symmetric group of order q, S_q. We shall find bounds on the max cycle lengths of the permutation matrices generated by the algorithm as a function of the choice of the first prime number in the seed vector $\mathbf{p}$, written $q = q_1$. We remind that according to $\mathcal{C}$, at this step a q-permutation structure is to be imposed over a q-partition Ω.

For the purposes of this section, we define[4] the following function of the random variable π, and the q-partition Ω:

$$L_n(\pi, \Omega) = \max_{P \in \mathcal{P}_n(\pi, \Omega)} \mathscr{L}_*(P)$$

where $\mathcal{P}_n(\pi, \Omega) = \{P \in \mathbb{P}_n \mid P \text{ acts as permutation } \pi \text{ over } \Omega\}$ (we remind that the notation $\mathbb{P}_n$ was defined in Remark 2). Clearly, $L_n(\pi, \Omega)$ is an upper bound on the cycle lengths of any permutation matrix acting as permutation π over Ω, and therefore provides us with an upper bound on the cycle lengths of the matrices in the matrix set produced by $\mathcal{C}$.

We shall study the behaviour of L_n separately for the two different cases `met` $=$ 2 and `met` $= 3$.

4.2.1 The case `met = 3`

If `met` $= 3$, the choice of the first block permutation structure is deterministic, indeed the identity permutation is always chosen, so $\pi \sim Uniform(\{\mathrm{id}_{S_q}\})$, where id_{S_q} is the identity in S_q. Imposing an identity block permutation structure over a q-partition implies that $\mathbb{P}[L_n(\pi, \Omega) = n/q] = 1$.

[4] See Definition 16 for the definition of the function $\mathscr{L}_*$.

4.2.2 The case `met = 2`

Supposing `met` $= 2$, since the first block permutation structure is chosen uniformly at random, we can find an upper bound on the probability of a particular matrix in the matrix set generated by $\mathcal{C}$ being cyclic[5] using the following theorem.

Theorem 12. *Let Ω be a q-partition of $\{1, \ldots, n\}$ and let $\pi \sim$ Uniform(S_q). Then it follows that: $\mathbb{P}[L_n(\pi, \Omega) = n] = 1/q$.*

Proof. $L_n(\pi, \Omega) = n$ iff π is a cyclic permutation. There are $(q-1)!$ cyclic permutations of the set Q, hence $\mathbb{P}[L_n(\pi, \Omega) = n] = (q-1)!/q! = 1/q$. □

We note that the probability of obtaining a cyclic letter after imposing a permutation structure over a q-partition must be less than $1/q$, so for large q, this situation is highly improbable. We can generalize this result to smaller cycle lengths as well. Letting the notation be as before, we have the following result.

Theorem 13. *Let Ω be a q-partition of $\{1, \ldots, n\}$ and let $\pi \sim$ Uniform(S_q).: $\mathbb{P}[L_n(\pi, \Omega) = (q-1)n/q] = 1/(q-1)$.*

Proof. $L_n(\pi, \Omega) = (q-1)n/q$ iff π has a cycle of length $q-1$. There are $q(q-2)!$ such permutations of the set Q (since there are q choices for the element that must be mapped to itself and $(q-2)!$ cyclic permutations on the $q-1$ remaining elements), hence $\mathbb{P}[L_n(\pi, \Omega) = n(q-1)/q] = q(q-2)!/q! = 1/(q-1)$. □

The preceding results suggest that the probability of obtaining letters with large cycle lengths relative to the dimension of the matrices decreases as the prime numbers used get larger. Of course this is unsurprising given that for larger q there are more possibilities for the cycle lengths of permutations on q elements, so we should expect all the probabilities to decrease as the probability distribution gets spread out over $\{1, \ldots, q\}$. What is more relevant is the expectation of $L_n(\pi, \Omega)$. We note however, that for all prime numbers q, $\mathbb{P}[L_n(\pi, \Omega) = n(q-1)/q] \geqslant \mathbb{P}[L_n(\pi, \Omega) = n]$ since $\mathbb{P}[L_n(\pi, \Omega) = n(q-1)/q]/\mathbb{P}[L_n(\pi, \Omega) = n] = q/(q-1) > 1$.

Generalizing these results gives the probability distribution of $L_n(\pi, \Omega)$. The generalization is not developed here as the problem is equivalent to finding the probability distribution of the max cycle length of a permutation chosen uniformly at random from the symmetric group of order q, and a recurrence relation for the number of permutations on n objects with greatest cycle length k was derived by Golomb and Gaal in [16]. The recurrence is not reproduced here but the relevant

[5] A permutation matrix of dimension $q \times q$ is cyclic iff its max cycle length is q. A permutation is cyclic iff the associated permutation matrix is.

interesting results from Golomb and Gaal's paper are given below.

We define first the *normalized max cycle length of a permutation matrix.*

Definition 27. *The normalized max cycle length, denoted by* $\overline{\mathscr{L}_*}$*, of a permutation matrix* P *of dimensions* $q \times q$ *is the max cycle length of* P *divided by* q*. More formally,* $\overline{\mathscr{L}_*}(P) = \mathscr{L}_*(P)/q$*.*

Naturally, we are interested in the expected value of the normalized max cycle length of P for $P \sim Uniform(\mathbb{P}_q)$. Unsurprisingly, it decreases as a function of n.

Theorem 14 (See [16] p.7)**.** *Let* $\pi_q \sim Uniform(S_q)$*. The expectation of the normalized max cycle length of the permutation matrix* P_{π_q} *associated to* π_q *is a decreasing sequence in* q *and tends to* λ *for large* q*, where* $\lambda \approx 0.62432965....$ *More formally:*

$$\lim_{q\to\infty} \mathbb{E}\big[\overline{\mathscr{L}_*}(P_{\pi_q})\big] = \lambda \approx 0.62432965...$$

To determine the median normalized max cycle length for large q we cite the following theorem from the same paper.

Theorem 15 (See [16] p.9)**.** *Let* $\pi_q \sim Uniform(S_q)$*. Letting* P_{π_q} *be the permutation matrix associated to* π_q*, we have that, for large* q*,* $\mathbb{P}\big[\overline{\mathscr{L}_*}(P_{\pi_q}) > a\big] = \ln(1/a)$ *for* $a \in \big[1/2, 1\big]$*.*

It follows that, asymptotically for large q, the probability that the normalized max cycle length is larger than $1/2$ is given by $\ln(2) \approx 0.69315$ and also that there is a one-in-two chance of a permutation on q elements having a normalized max cycle length larger than $1/\sqrt{\mathrm{e}} \approx 0.60653$ (or in other words, the median value is $1/\sqrt{\mathrm{e}}$).

To conclude, Theorem 14 gives us reason to expect that the normalized max cycle length of the permutation matrices in the matrix set produced by $\mathcal{C}$ should decrease as the prime numbers used in the seed vector $\mathbf{p}$ get larger. The least restrictive case would then seem to be the case where $\mathbf{p} = [q_1, \ldots, q_m] = [2, \ldots, 2]$. This case is analyzed in greater detail in Section 4.5. It is important to note however, that all the preceding results depend crucially on the choice of the probability distribution according to which permutations are chosen, it may well be possible to find a distribution such that the expectation of the normalized max cycle length does not decrease as the prime numbers grow larger, this was not attempted.

4.3 Bounding the max cycle lengths as a function of the number of block permutation structures imposed

We would like to establish bounds in this section on the max cycle length of a permutation matrix that acts as permutation σ_j over the q_j-partition Ω_j for all $j = 1, \ldots, m-1$. These bounds will then be used to study the perturbed permutation matrix sets produced by $\mathcal{C}$ in Section 4.5. We begin by proving the following lemma, which is a generalization of the well-known bipartite graph theorem[6].

Lemma 19. *Let M be a permutation matrix, $\mathcal{G}(\{M\}) = (V, E)$ its associated graph, and $\Omega = \{\Omega_1, \ldots, \Omega_q\}$ a partition of V. If the matrix M acts as a cyclic permutation over the partition Ω, then the length of the cycle, q, divides the length of any cycle in $\mathcal{G}(\{M\})$.*

Proof. Let $v_0 \to v_1 \to \cdots \to v_l = v_0$ be a cycle (of length l) in $\mathcal{G}(\{M\})$. WLOG, we may relabel the sets in Ω such that $v_0 \in \Omega_0$ and $\forall i = 0, \ldots, q-1$ all the edges from Ω_i go to $\Omega_{i+1 (\mathrm{mod}\ q)}$. Since $v_0 \in \Omega_0$, then $v_1 \in \Omega_{1 (\mathrm{mod}\ q)}$, and $v_2 \in \Omega_{2 (\mathrm{mod}\ q)}$, and more generally:

$$v_r \in \Omega_{r (\mathrm{mod}\ q)}, \quad \forall r = 0, \ldots, l \tag{4.1}$$

But $v_0 = v_l$ and by (4.1), $v_0 \in \Omega_0$ and $v_l \in \Omega_{l (\mathrm{mod}\ q)}$. Since Ω is a partition, it follows that $\Omega_0 = \Omega_{l (\mathrm{mod}\ q)} \implies l = 0 \pmod{q} \implies q \mid l$, that is to say that q divides l.

□

We can also study the sequence of nodes in the cycles of a permutation matrix with a more general block permutation structure. Recalling that any permutation's graph is a union of disjoint cycles (see Lemma 5), we develop a notation for partitions over which a matrix acts as a permutation. The fact that the following renumbering of sets of a partition is always possible follows trivially from the definition of a matrix acting as a permutation over a partition.

Definition 28. *Let $\sigma \in S_q$ be a permutation with k cycles[7] of lengths $c_1, \ldots, c_k$. Also let n be an integer such that $n \geqslant q$, and let $\Omega = \{\Omega_1, \ldots, \Omega_q\}$ be a partition of*

[6] The bipartite graph theorem states that a graph is bipartite if and only if every cycle in the graph is of even length.

[7] That is to say such that the graph associated to the matrix associated to σ is made of k disjoint cycles.

$\{1,\dots,n\}$. *Finally let* $v \in \{1,\dots,n\}$. *We say that* Ω' *is* Ω *numbered according to* v *and* σ *iff* $v \in \Omega_{1,0}$ *and for all* $r = 1,\dots,k$*: all the edges from* $\Omega_{r,s}$ *go to* $\Omega_{r,s+1(mod\ c_r)}$ *for all* $s = 0,\dots,c_r - 1$ *and* $\Omega' = \{\Omega_{r,s} \mid r \in \{1,\dots,k\}, s \in \{0,\dots,c_r-1\}\} = \Omega$.

Generalizing the result of Lemma 19, we get the following lemma on cycles in a permutation matrix satisfying $m-1$ block permutation structures.

Lemma 20. *Let* M *be a permutation matrix,* $\mathcal{G}(\{M\}) = (V,E)$ *its associated graph, and* $\Omega^j = \{\Omega^j_1,\dots,\Omega^j_{q_j}\}$ *be partitions of* V *for* $j = 1,\dots,m-1$*. Also let* $v_0 \to v_1 \to \cdots \to v_l = v_0$ *be a cycle in* $\mathcal{G}(\{M\})$*. For every* $j = 1,\dots,m-1$*, if* M *acts as permutation* σ_j *over* Ω^j*, then assuming*[8] Ω^j *to be numbered according to* v_1 *and* σ_j*, we have the following constraint:*

$$v_s \in \Omega^j_{1,s(mod\ c^j_1)} \text{ for all } s = 0,\dots,c^j_1 - 1. \tag{4.2}$$

Proof. Follows trivially from Definition 28. □

This immediately gives a bound on the maximal cycle length of the permutation matrices in the perturbed permutation matrix set produced by $\mathcal{C}$, as stated in the following corollary.

Corollary 2. *Let* M *be a permutation matrix,* $\mathcal{G}(\{M\}) = (V,E)$ *its associated graph, and* $\Omega^j = \{\Omega^j_1,\dots,\Omega^j_{q_j}\}$ *be partitions of* V *for* $j = 1,\dots,m-1$*. Also let* $v_0 \to v_1 \to \cdots \to v_l = v_0$ *be a cycle in* $\mathcal{G}(\{M\})$*. For every* $j = 1,\dots,m-1$*, if* M *acts as permutation* σ_j *over* Ω^j*, then assuming* Ω^j *to be numbered according to* v_0 *and* σ_j*, we have that:*

$$l \leqslant \left(\min_{s \in \{1,\dots,\mathrm{lcm}(c^1_1,\dots,c^{m-1}_1)\}} \left| \bigcap_{j=1,\dots,m-1} \Omega^j_{1,s(mod\ c^j_1)} \right| \right) \mathrm{lcm}(c^1_1,\dots,c^{m-1}_1)$$

Proof. Let $c = \mathrm{lcm}(c^1_1,\dots,c^{m-1}_1)$ so $c^j_1 \mid c$ for all $j = 1,\dots,m-1$. By equation (4.2) in Lemma 20, we have that $v_{kc} \in \bigcap_{j=1,\dots,m-1} \Omega^j_{1,0}$ for all $k \in \mathbb{N}_0$. Since a cycle can only visit each element once, it follows that $l \leqslant \left|\bigcap_{j=1,\dots,m-1} \Omega^j_{1,0}\right| c$.

Now renaming the partition Ω according to v_s, we can reapply the same reasoning to deduce that:

$$l \leqslant \left| \bigcap_{j=1,\dots,m-1} \Omega^j_{1,s(\mathrm{mod}\ c^j_1)} \right| c \text{ for all } s = 1,\dots,c.$$

The conclusion follows. □

[8] WLOG of course.

Remark 8. *It also follows from this reasoning that any cycle in the associated graph of M must have a length of at least c.*

Since the expected value of the parameter c for a random sequence of permutations seems to be difficult to calculate, we obtain also the following bound, which is easier to apply in the general case.

Corollary 3. *Let M be a permutation matrix, $\mathcal{G}(\{M\}) = (V, E)$ its associated graph, and $\Omega^j = \{\Omega^j_1, \ldots, \Omega^j_{q_j}\}$ be partitions of V for $j = 1, \ldots, m-1$. Also let $v_0 \to v_1 \to \cdots \to v_l = v_0$ be a cycle in $\mathcal{G}(\{M\})$. For every $j = 1, \ldots, m-1$, if M acts as permutation σ_j over Ω^j, then assuming Ω^j to be numbered according to v_0 and σ_j, we have that:*

$$l \leqslant \left| \bigcap_{j=1}^{m-1} \bigcup_{s=0}^{c_1^j - 1} \Omega^j_{1,s} \right|$$

Proof. Let $j \in \{1, \ldots, m\}$. By Definition 28 and Lemma 20, $v_0 \in \Omega^j_{1,0}$ and if $v_i \in \Omega^j_{1,i}$ then $v_{i+1} \in \Omega^j_{1,i+1(\bmod c_1^j)}$. Therefore:

$$v_i \in \bigcup_{s=0}^{c_1^j - 1} \Omega^j_{1,s} \text{ for all } i = 0, \ldots, l-1 \implies \{v_0, \ldots, v_{l-1}\} \subseteq \bigcup_{s=0}^{c_1^j - 1} \Omega^j_{1,s}.$$

Since j was arbitrary, it follows that $\{v_0, \ldots, v_{l-1}\} \subseteq \bigcap_{j=1}^{m-1} \bigcup_{s=0}^{c_1^j - 1} \Omega^j_{1,s}$. And the conclusion follows by noting that the $v_0, \ldots, v_{l-1}$ are distinct since they form a cycle, and so the set $\{v_0, \ldots, v_{l-1}\}$ has cardinality l. □

These corollaries seems to suggest an intuitive reason for why the cycle lengths of the permutation matrices diminish as a function of m. Indeed the intersections of the various partitions can be expected to contain fewer and fewer elements as m increases, so if the rate of decrease of the cardinality of these intersections exceeds the rate of increase of n as a function of m, the maximal cycle length can be expected to diminish as well. We shall study the issue further in section 4.5.

4.4 A result on compatible block permutation structures

In this section we investigate in greater detail the notion of compatibility. We shall do so by studying how a matrix acts over its intersection partitions.

Definition 29. *Let a square binary matrix M of dimension $n = q_1 \dots q_m$ act as permutation $\sigma_j \in S_{q_j}$ over q_j-partition $\Omega^j = \{\Omega_1^j, \dots, \Omega_{q_j}^j\}$ of $\{1, \dots, n\}$ for $j = 1, \dots, m-1$. For all $k = 1, \dots, m-1$, letting $n_k = q_1 \dots q_k$ we define the k-th intersection partition as :*

$$\begin{aligned}\Gamma^k &= \left\{\Gamma_1^k, \dots, \Gamma_{n_k}^k\right\} \\ &= \left\{\Omega_{i_1}^1 \bigcap \dots \bigcap \Omega_{i_k}^k \;\middle|\; i_j \in \{1, \dots, q_j\} \text{ for all } j \in \{1, \dots, k\}\right\}.\end{aligned}$$

Where $\Gamma_i^k(\Omega^1, \dots, \Omega^k) = \Omega^1_{(i-1) (\mathrm{mod}\ q_1)+1} \bigcap \left(\bigcap_{j=2}^k \Omega^j_{\lfloor \frac{i-1}{q_1 \dots q_{j-1}} \rfloor (\mathrm{mod}\ q_j)+1} \right)$. Clearly Γ^{m-1} forms a partition of $\{1, \dots, n\}$.

Lemma 21. *Let a square binary matrix M of dimension $n = q_1 \dots q_m$ act as permutation $\sigma_j \in S_{q_j}$ over q_j-partition $\Omega^j = \{\Omega_1^j, \dots, \Omega_{q_j}^j\}$ of $\{1, \dots, n\}$ for $j = 1, \dots, m-1$. Then M acts as a permutation over the $(m-1)$-th intersection partition of $\{1, \dots, n\}$, $\Gamma^{m-1} = \left\{\Gamma_1^{m-1}, \dots, \Gamma_{n_k}^{m-1}\right\}$:*

Proof. We prove this lemma by induction, for $m = 1$, this is trivial.

We now suppose the theorem true for all $m = 1, \dots, m'$ and show that it holds also for $m = m' + 1$ when we impose a new permutation $\sigma_{m'+1}$ over a $q_{m'+1}$-partition $\Omega^{m'+1}$. Letting $n_{m'} = q_1 \dots q_{m'}$, we have by the inductive hypothesis that M acts as some permutation π over $\{\Gamma_1^{m'}, \dots, \Gamma_{n_{m'}}^{m'}\}$. To this end, we consider the submatrices $M_{\Gamma_i^{m'}, \Gamma_{\pi(i)}^{m'}}$ for $i = 1, \dots, n_{m'}$ as all the other blocks are zero.

Let $i \in \{1, \dots, n_{m'}\}$. Since M acts as a permutation π on $\Gamma^{m'}$, if M acts also as a permutation $\sigma^{m'+1}$ on $\Omega^{m'+1}$, then if $M_{i,j} > 0$ there exist $k_1 \in \{1, ..., n_{m'}\}, k_2 \in \{1, ..., q_{m'+1}\}$ such that $i \in \Gamma_{k_1}^{m'}$ and $\Omega_{k_2}^{m'+1}$, $j \in \Gamma_{\pi(k_1)}^{m'}$ and $j \in \Omega^{m'+1}_{\sigma^{m'+1}(k_2)}$. The conclusion follows. □

We shall see that the submatrices we just considered turn out to hold the key to the notion of compatibility. We have the following result.

Theorem 16. *Let a square binary matrix M of dimension $n = q_1 \dots q_m$ act as permutation $\sigma_j \in S_{q_j}$ over q_j-partition $\Omega^j = \{\Omega_1^j, \dots, \Omega_{q_j}^j\}$ of $\{1, \dots, n\}$ for $j = 1, \dots, k$, $k < m-1$. Let $q_1 \dots q_k = n_k$ and let π be the permutation such that M acts as π over Γ^k, the k-th intersection partition of $\{1, \dots, n\}$. Then a q_{k+1}-partition Ω^{k+1} is compatible with M iff there exists a permutation σ_{k+1} in $S_{q_{k+1}}$ and a matrix M' dominated by M such that for all $i = 1 \dots, n_k$, and all $k_1, k_2 \in \{1, \dots, q_{k+1}\}$ such that $\Gamma_i^k \bigcap \Omega_{k_1}^{k+1}, \Gamma_{\pi(i)}^k \bigcap \Omega_{k_2}^{k+1}$ are not empty the following holds:*

$$\sigma_{k+1}(k_1) \neq k_2 \implies M'_{\Gamma_i^k \bigcap \Omega_{k_1}^{k+1}, \Gamma_{\pi(i)}^k \bigcap \Omega_{k_2}^{k+1}} = 0, \tag{4.3}$$

and $M'_{\Gamma_i^k, \Gamma_{\pi(i)}^k}$ *dominates a permutation.*

Proof. The forward direction follows trivially from Lemma 21 and Definition 26.

For the reverse direction it suffices to note that since M acts as π over Γ^k and $M' \leqslant M$, for all $i_1, i_2 \in \{1, \ldots, n_k\}$, $\pi(i_1) \neq i_2 \implies M'_{\Gamma_{i_1}^k, \Gamma_{i_2}^k} = 0$. Condition (4.3) then ensures that M' acts as σ_{k+1} over Ω^{k+1}. Now clearly if each of the submatrices $M'_{\Gamma_i^k, \Gamma_{\pi(i)}^k}$ dominates a permutation, then M' does too. The conclusion follows. □

The preceding theorem is applied in the next section to study the compatibility of 2-partitions.

4.5 Analysis of a particular case

We consider the behaviour of $\mathcal{C}$ in the case where $\mathbf{p} = [2, \ldots, 2]$ and $\texttt{met} = 2$. We shall see that in this case there are two ways to arrive at the same result.

Let P be a matrix belonging to the perturbed permutation set produced by $\mathcal{C}$. By Corollary 2, we know that:

$$\mathbb{E}[\mathscr{L}_*(P)] \leqslant 2\mathbb{E}\left[\left|\Gamma_1^{m-1}\right|\right] \tag{4.4}$$

where Γ_1^{m-1} is an arbitrary element of Γ^{m-1}, the $(m-1)$-th intersection partition of $\{1, \ldots, 2^m\}$.

The other way to come to the same conclusion is to note that following the reasoning of Lemma 21, we can prove that P must act as an *involution*[9] over Γ^{m-1}. So the expected max length of a cycle in P is bounded by twice the expectation of the cardinality of an aribtrary element of Γ^{m-1}. The proof that P must act as an involution is given below.

Theorem 17. *Let $\mathcal{M}$ be the perturbed permutation matrix set produced by $\mathcal{C}$ with inputs $\mathbf{p} = [q_1, \ldots, q_m] = [2, \ldots, 2]$ and $\texttt{met} = 2$ or 3. Also let Γ^{m-1} be the $(m-1)$-th intersection partition of $\{1, \ldots, 2^m\}$. Then any $P \in \mathcal{M}$ acts on Γ^{m-1} as an involution.*

Proof. The theorem is proved by induction on the Γ^j for $j = 1, \ldots, m-1$. The case $j = 1$ is trivial since any permutation on 2 elements is an involution.

[9] An involution is a permutation that is its own inverse.

We now suppose that P acts as an involution π on Γ^{j-1} and show that if we impose a 2-permutation σ over a 2-partition Ω^j, then P will act as an involution over Γ^j. Let $n_{m-1} = q_1 \dots q_{m-1}$ and let $v_0 \to v_1 \to \cdots \to v_l = v_0$ be a cycle in $\mathcal{G}(\{P\})$. then, numbering Ω^j according to σ and v_0, we have that if $v_0 \in \Gamma_i^{j-1} \bigcap \Omega_{1,0}^j$ for some $i \in \{1, \dots, n_{m-1}\}$, then $v_1 \in \Gamma_{\pi(i)}^{j-1} \bigcap \Omega_{1,1(\text{mod } c_1)}^j \implies v_2 \in \Gamma_i^{j-1} \bigcap \Omega_{1,0}^j$. It follows that P acts as an involution over Γ^j. □

We note γ^j the cardinality of an arbitrary element of Γ^j for $j = 1, \dots, m-1$. Trivially, $\mathbb{E}[\gamma^1] = n/2$. Now if the partitions were chosen without regard to compatibility, we would have that $\gamma^{j+1} \sim Hypergeometric(n, \gamma^j, n/2)$ for $j = 1, \dots, m-1$. We easily obtain then, by noting that the mean of a random variable distributed hypergeometrically with population n, number of success states γ and draw size $n/2$ is $\gamma/2$, that: $\mathbb{E}[\gamma^{j+1}] = \sum_{i=0}^{n/2} \mathbb{E}[\gamma^{j+1} \mid \gamma^j = i]\mathbb{P}[\gamma^j = i] = (1/2)\sum_{i=0}^{n/2} i\mathbb{P}[\gamma^j = i] = \mathbb{E}[\gamma^j]/2$. It follows that $\mathbb{E}\left[\left|\Gamma_1^{m-1}\right|\right] = 2$, and so equation (4.4) gives $\mathbb{E}[\mathscr{L}_*(P)] \leqslant 4$, regardless of the value of m.

However, if we take into account the fact that the partitions chosen must be compatible, the analysis becomes a lot more complicated. We have the following theorem on compatibility of 2-partitions.

Theorem 18. *Let M be an $n \times n$ binary matrix acting as an involution $\pi \in S_{n_j}$ over some partition $\Gamma = \{\Gamma_1, \dots, \Gamma_{n_j}\}$ of $\{1, \dots, n\}$ such that for all $i = 1, \dots, n_j$, $M_{\Gamma_i, \Gamma_{\pi(i)}}$ is either empty if Γ_i or $\Gamma_{\pi(i)}$ is, or else is an all-ones matrix. Then a 2-partition $\Omega = \{\Omega_1, \Omega_2\}$ of $\{1, \dots, n\}$ is incompatible with M iff there exist $i_1, i_2 \in \{1, \dots, n_j\}, i_1 \neq i_2$, such that $\Gamma_{i_1}, \Gamma_{\pi(i_1)}, \Gamma_{i_2}, \Gamma_{\pi(i_2)} \neq \varnothing$ and there exists $k \in \{1, 2\}$:*

$$\left(\Gamma_{i_1} \bigcup \Gamma_{i_2} \bigcup \Gamma_{\pi(i_1)}\right) \bigcap \Omega_k = \varnothing, \Gamma_{\pi(i_2)} \bigcap \Omega_k = \Gamma_{\pi(i_2)} \tag{4.5}$$

Proof. Let $i_1, i_2 \in \{1, \dots, n_j\}, i_1 \neq i_2$, $\Gamma_{i_1}, \Gamma_{\pi(i_1)}, \Gamma_{i_2}, \Gamma_{\pi(i_2)} \neq \varnothing$, $k \in \{1, 2\}$ such that (4.5) is satisfied and consider the submatrices of M:

$$M_{i_1} = \begin{array}{c} \\ \Gamma_{i_1} \bigcap \Omega_1 \\ \Gamma_{i_1} \bigcap \Omega_2 \end{array} \begin{array}{c} \begin{array}{cc} \Gamma_{\pi(i_1)} \bigcap \Omega_1 & \Gamma_{\pi(i_1)} \bigcap \Omega_2 \end{array} \\ \begin{pmatrix} . & . \\ . & . \end{pmatrix} \end{array}$$

$$M_{i_2} = \begin{array}{c} \\ \Gamma_{i_2} \bigcap \Omega_1 \\ \Gamma_{i_2} \bigcap \Omega_2 \end{array} \begin{array}{c} \begin{array}{cc} \Gamma_{\pi(i_2)} \bigcap \Omega_1 & \Gamma_{\pi(i_2)} \bigcap \Omega_2 \end{array} \\ \begin{pmatrix} . & . \\ . & . \end{pmatrix} \end{array}$$

Then, letting $k' = 2$ if $k = 1$ and $k' = 1$ if $k = 2$, compatibility implies that the second block in the k'-th row of M_{i_1} must dominate a permutation, but also that the first block in the k'-th row of M_{i_2} must dominate a permutation. But then, if M is to act as a permutation over Ω, the first condition implies that the permutation must be the identity, and the second implies that the permutation must be cyclic. But no such permutation exists. Therefore Ω is incompatible with M.

In the reverse direction, let us suppose that Ω is not compatible with M. By hypothesis all non-empty $M_i = M_{\Gamma_i, \Gamma_{\pi(i)}}$ for $i \in \{1, \ldots, n_j\}$ are all-ones matrices. Clearly if all the $\Gamma_i \bigcap \Omega_k$ for $i \in \{1, \ldots, n_j\}$, $k = 1, 2$ are non-empty, there exists $M' \leqslant M$ and $\sigma \in S_2$ such that M' acts as σ over Ω and dominates a permutation matrix. The same is true if in each M_i at most one of the block rows or columns is empty. If both rows or both columns are empty then the submatrix is empty. We note then that we only have incompatibility if either (4.5) is satisfied, or the following is satisfied for some $i_1, i_2 \in \{1, \ldots, n_j\}$, such that $\Gamma_{i_1}, \Gamma_{\pi(i_1)}, \Gamma_{i_2}, \Gamma_{\pi(i_2)} \neq \varnothing$, and some $k \in \{1, 2\}$.

$$\left(\Gamma_{\pi(i_1)} \bigcup \Gamma_{\pi(i_2)} \bigcup \Gamma_{i_1}\right) \bigcap \Omega_k = \varnothing \text{ and } \Gamma_{i_2} \bigcap \Omega_k = \Gamma_{i_2} \tag{4.6}$$

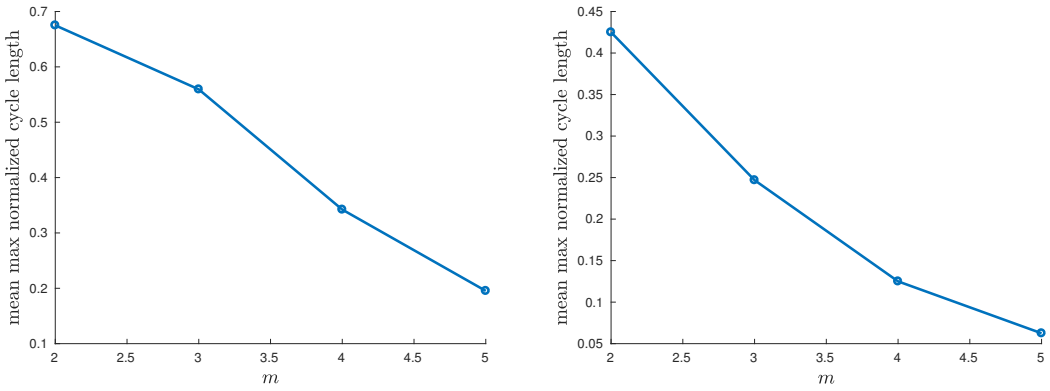
However, since π is an involution, (4.5) is satisfied for some value of i_1, i_2 iff (4.6) is satisfied by $\pi(i_1), \pi(i_2)$. Therefore (4.5) is necessarily satisfied. □

By using condition (4.5), we should be able to count the number of partitions compatible with a given matrix M using combinatorial methods (indeed finding the number of partitions satisfying (4.5) for fixed i_1, i_2 turns out to be identical to a standard problem of drawing colored balls from an urn without replacement). And since the number of 2-partitions of $\{1, \ldots, n\}$ is given by $(1/2)\binom{n}{n/2}$, we can then also compute the probability of finding a compatible partition. However as the same partition can satsify (4.5) for different values of i_1, i_2, it is not clear how to use this equation to count the number of incompatible partitions without overcounting. This problem has yet to be solved.

4.6 Numerical validation of the conjecture

In this section, we aim to show that numerical simulations support the hypothesis that the normalized cycle lengths of the permutation matrices produced by the

algorithm $\mathcal{C}$ decrease on average as a function of m. We implemented the algorithm $\mathcal{C}$ in `Matlab`, and a function that computes the maximum normalized cycle length of an arbitrary matrix set consisting of permutation matrices. We produced 15 sets for each of the values of m, for `met=2`, and `met=3`, using the seed vector $\mathbf{p} = [2, \ldots, 2]$ ($\mathbf{p}$ is a vector of length m).



(a) `met` $= 2$ (b) `met` $= 3$

Figure 4.1: Graphs showing the decrease of the mean normalized cycle length as a function of m.

As expected, the maximum normalized cycle length of the matrices in the primitive perturbed permutation sets produced by $\mathcal{C}$ is seen to decrease as a function of m. We also note that when using `met=3`, the decrease is a lot more rapid than with `met=2`, which is unsurprising given the results of section 4.2. Indeed, it seems quite plausible that the higher average pair digraph diameter of the automata produced by $\mathcal{C}$ using `met=3` as opposed to `met=2` observed in [11] can be explained by the fact that the cycle lengths are a lot smaller in the former case, thereby raising the likelihood of the algorithm producing the families of slowly synchronizing automata (which all have a max cycle length of 2) presented in this paper.

We note also that in the case `met` $= 3$, we seem to have the bound $1/2^{m-1}$ on the expectation of the max normalized cycle length, the proof of this fact for arbitrary m has yet to be attempted[10].

[10]This is easy to prove for $m = 2, 3$ since all 2-partitions are compatible and therefore the partitions are chosen according to the hypergeometric distribution, as described in the previous section.

Chapter 5

Heuristic algorithms for the approximation of reset thresholds and exponents

In the literature, there exist several heuristic methods for approximating the k-RT of automata. Since computing the true reset threshold is NP hard, these heuristics are very useful in practice, since they allow us to find a synchronizing word of an automaton (the reset threshold being equal to the n-RT of an automaton) in a reasonably short amount of time. These heuristics can be easily extended for the purposes of approximating the k-RT of matrix sets. One of the most widely used heuristics is the Eppstein heuristic, that will be described in the first section. In the second section, we present a new heuristic: the transpose graph heuristic. In the third section, we show how we can use these heuristics in order to approximate the exponent of a primitive matrix set. Heuristics for approximating the exponent of a primitive matrix set are, to the best of our knowledge, unknown in the literature. In the last section, we compare both heuristics on the automata generated by the randomized algorithm ($\mathcal{C}$) for generating extremal automata which was presented in the preceding chapter. We shall see that our new heuristic outperforms the Eppstein heuristic in this case, though it is somewhat slower.

5.1 Eppstein heuristic

In [13], Eppstein proposed an algorithm for finding short reset words of a synchronizing automaton. The idea is simple, given an automaton and its set of states Q, we produce the reset word τ in the following manner: at each iteration we pick two distinct states q_1, q_2 in Q, find a word τ' that sends them to the same state, and finally set τ to $\tau\tau'$ and Q to the image of Q by τ'.

At each iteration, there are of course different ways to choose the two distinct states q_1 and q_2, and different ways of choosing the word τ'. A simple strategy in each case is the greedy strategy. The greedy strategy for choosing τ' is to pick the shortest τ' such that $\delta(q_1, \tau') = \delta(q_2, \tau')$, and the greedy strategy for the choice of q_1 and q_2 consists in choosing whatever pair of distinct states in Q gives the shortest τ', the latter being chosen according to the greedy strategy for choosing τ'. In order to evaluate the quality of this strategy as a method for approximating the reset threshold, we introduce the notion of the *approximation ratio*.

Definition 30 (See [3], Section 1)**.** *Let Z be an algorithm giving the length of a (short) reset word for any synchronizing automaton so that for an arbitrary synchronizing automaton $\mathcal{A}$, $Z(\mathcal{A}) \geqslant \mathrm{rt}(\mathcal{A})$. The approximation ratio of Z is the following function of n:*

$$R_Z(n) = \sup\{Z(\mathcal{A})/\mathrm{rt}(\mathcal{A}) \mid \mathcal{A} \textit{ is a synchronizing automaton with } n \textit{ states}\}$$

It was shown in [3], Theorem 3.4 that the approximation ratio of the greedy Eppstein heuristic is at least $n/6$ for $n > 6$. Nevertheless, the reset words found by this heuristic are upper bounded by a cubic function of n (see [13], Theorem 7).

We extend the algorithm to binary $n \times n$ matrix sets with non zero rows that have a product with a positive column (we know from Lemma 5 that for those kind of matrix sets, there is a path from every non singleton to some singleton in the pair digraph) as follows. The idea is to use the greedy strategies for the Eppstein heuristic to find products with successively larger column weight at each iteration using the result of Lemma 6 until we arrive finally at a product with a positive column. Given a matrix set $\mathcal{M}$, for any nodes (i, j) and (k, l) in $\mathcal{SG}(\mathcal{M})$ (see Definition 17), let us denote by $d_{\mathcal{SG}(\mathcal{M})}\big[(i, j), (k, l)\big]$ the length of the shortest path from (i, j) to (k, l) in $\mathcal{SG}(\mathcal{M})$. We also denote the set $\{1, \ldots, n\}$ by the convenient shorthand $\big[n\big]$. The Eppstein heuristic for primitive sets is formalized by Algorithm 1.

Now since this algorithm increases the weight of the column i at each iteration of the while loop, it also provides us with an upper bound on the the k-RT of the corresponding matrix set for all k by producing a (short) product whose i-th column is of weight greater than or equal to k. So at each iteration we also check the weights of the rows of A in case the algorithm happens to produce a larger row weight than the maximal column weight, thus improving the bound on the k-RT.

Algorithm 3: Pseudo-code algorithm for the Eppstein heuristic

Input: A matrix set $\mathcal{M}$ which has a product with a positive column
Output: A matrix $M \in \mathcal{M}^*$ with a positive column

$A \longleftarrow \arg\max_{X \in \mathcal{M}} \max_{i \in [n]} |\mathrm{supp}(X_{*i})|$;
$i \longleftarrow \arg\max_{j \in [n]} |\mathrm{supp}(A_{*j})|$;
$S \longleftarrow \mathrm{supp}(A_{*i})$;

while $S \neq [n]$ **do**

$\mathcal{C} \longleftarrow \{j \in [n] : \mathrm{supp}(A_{*j}) \nsubseteq S\}$;
$j^* \longleftarrow \arg\min_{j \in \mathcal{C}} d_{\mathcal{SG}(\mathcal{M})}[(i,j),(s,s)]$ for $s \in \{1,...,n\}$ arbitrary;
$A_{p_1}, \ldots, A_{p_l} \longleftarrow$ labels of shortest path from (i,j) to (s,s);
$i \longleftarrow s$;
$A \longleftarrow AA_{p_1} \cdots A_{p_l}$;
$S \longleftarrow \mathrm{supp}(A_{*i})$;

end

return A;

5.2 A new heuristic: the transpose graph heuristic

We propose here another heuristic for finding short reset words of an automaton, or more generally a method of solving the following problem: given a matrix set $\mathcal{M}$ of m binary matrices satisfying the conditions of Lemma 5, to find a short product of matrices from $\mathcal{M}$ which gives a matrix that has a positive column. To the best of our knowledge, the heuristic presented in this section is new, we could not find it anywhere in the literature. Our heuristic is based on the simple idea stated in the lemma below.

Lemma 22. *The matrix $M = M_1 \ldots M_s$ has a positive column iff $M^T = M_s^T \ldots M_1^T$ has a positive row.*

Proof. Trivial. □

In light of the preceding lemma, instead of finding a product of matrices in $\mathcal{M}$ which gives a matrix with a positive column, we search for a product of matrices

in $\mathcal{M}^T$ which gives a matrix with a positive row. Then we can solve the original problem simply by reversing the order of this product and replacing the matrices by their transposes.

The advantage of using the transpose graph is that it allows us to build the synchronizing word *in reverse*. So we can use information about the end of our synchronizing word to make better decisions at the beginning of our synchronizing word. There is evidence suggesting that methods that build synchronizing words in reverse work better on average, see [24][1].

By Lemma 3, we know that the product $M^T = M_1^T \dots M_s^T$ has a positive row if and only if there exists $i \in \{1, \dots, n\}$ such that there is a path labelled by $M_1^T, \dots, M_s^T$ from i to j in $\mathcal{G}(\mathcal{M}^T)$, for all $j \in \{1, \dots, n\}$. Now in order to find such paths, we introduce some supplementary notions.

Definition 31. *Let $\mathcal{M}$ be a set of binary matrices, $\mathcal{G}(\mathcal{M}^T) = (V, E^T)$ its associated graph and let 2^V be the powerset of V. For ease of notation we define the function $* : 2^V \times \mathcal{M}^T \to 2^V : (U, M^T) \mapsto \{v \in V \mid \exists u \in U : (u, v) \in E^T \text{ with label } M^T\}$. We write $*(U, M^T)$ as $U * M^T$. We also extend the notation so that for all $U \subseteq V, M_1^T, M_2^T \in \mathcal{M}^T$, we have: $U * M_1^T M_2^T = (U * M_1^T) * M_2^T$.*

Definition 32. *Let $\mathcal{G}(\mathcal{M}^T) = (V, E^T)$. We call $u \in V$ a branching node if and only if there exist $v, w \in V, M^T \in \mathcal{M}^T$ such that the following three conditions are fulfilled:*

- $v \neq w$,
- $v \in \{u\} * M^T$,
- $w \in \{u\} * M^T$.

M^T is then called a branching letter for u.

The following lemma shows the relevance of these notions to the construction of the required paths.

Lemma 23. *Let $M^T = M_1^T \dots M_s^T$ have a row i of weight k, $k \in \{1, \dots, n\}$, where $M_1^T, \dots, M_s^T \in \mathcal{M}^T$. Let $U = \{u_1, \dots, u_k\} = supp((M^T)_{i*})$. Now if $M^T M_{s+1}^T \dots M_{s+t}^T$ has row i of weight $> k$, then there exists some $x \in \{0, \dots t-1\}$, some branching node $v_\star \in V$ and some branching letter $M_\star^T \in \mathcal{M}^T$ for v_* such that $v_* \in U * M_{s+1}^T \dots M_{s+x}^T$ and $M_{s+x+1}^T = M_\star^T$.*

[1]It is to be noted that the algorithm they present is similar to the transpose graph heuristic, but not quite the same.

Proof. Straightforward by contradiction. Indeed, unless you apply a branching letter to a branching node, the cardinality of the image set is always less than or equal to that of the set. □

This suggests a simple strategy for solving our problem. Start with a matrix that has a row, say row i, of weight at least 2, see what nodes the (at least two) corresponding edges lead to, and let S be the set of these nodes. Now for each node in S, find the shortest path sending it to a branching node, if the labels corresponding to this path give a matrix product that increases the maximum row weight then proceed to the next step, otherwise try again with some other node in S. If all nodes in S have been tried with no success, then we apply one step of the Eppstein algorithm to increase the weight and we set S to be the set of all nodes reached by the paths found so far. Now for each node in S we find the shortest paths sending it to a branching node ... and so until we finally have a positive row. This idea is captured by the following pseudocode algorithm.

Algorithm 4: Pseudo-code algorithm for the Transpose-graph heuristic

Input: A matrix set $\mathcal{M}$ which has a product with a positive column
Output: A matrix $M \in \mathcal{M}^*$ with a positive column

$A \longleftarrow \arg\max_{X \in \mathcal{M}} \max_{i \in [n]} |\mathrm{supp}(X_{*i})|$;
$i \longleftarrow \arg\max_{j \in [n]} |\mathrm{supp}(A_{*j})|$;
$S \longleftarrow \mathrm{supp}(A_{*i})$;
$T \longleftarrow \{v \in V \mid v \text{ is a branching node}\}$;

Continued on the next page...

while $S \neq [n]$ **do**

$B \longleftarrow 0$;
dists $\longleftarrow$ pairwise distances from S to T in $\mathcal{G}(\mathcal{M}^T)$;
paths $\longleftarrow$ the corresponding paths;

while $B = 0$ *and paths not empty* **do**

$A_{p_1}, \ldots, A_{p_l} \longleftarrow$ a path from $s \in S$ to $t \in T$ in paths of shortest distance in dists;
$A^* \longleftarrow$ a branching letter for t.
if $|S * A_{p_1} \cdots A_{p_l} A^*| > |S|$ **then**
$A \longleftarrow A A_{p_1} \cdots A_{p_l} A^*$;
$S \longleftarrow \mathrm{supp}(A_{*i})$;
$B \longleftarrow 1$;
end
else
remove this path from paths;
if *no more paths corresponding to this distance* **then**
remove this distance from dists;
end
end

end

if $B = 0$ **then**

apply a step of the Eppstein heuristic to increase row weight;
update S accordingly;

end

end

return A;

Clearly, at each iteration, if a successful path is found without recourse to an Eppstein step, then the length of this path is bounded by n (because it takes at most $n-1$ steps to reach a branching node, and then we multiply the matrix corresponding to the labels of the two branching edges of the branching node). Therefore if the algorithm never uses Eppstein, the length of the final synchronizing word found is bounded quadratically, by $n(n-2)+1 = (n-1)^2$.

5.3 Approximating the exponent

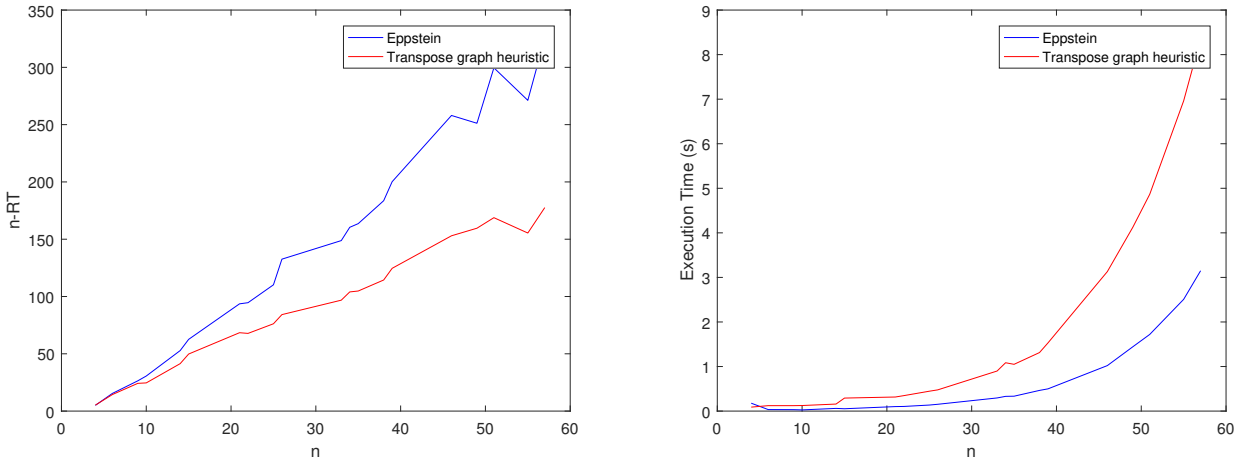
In the previous sections of this chapter, we presented some heuristics that allow us to find a product with a positive column. In what follows, we explain how we can use those heuristics in order to approximate the exponent of a primitive matrix set. Borrowing an idea from [7], Theorem 17, given a primitive matrix set $\mathcal{M}$ we can construct a positive matrix in the following manner. Let $\mathcal{H}$ stand for any of the preceding heuristics, we then apply $\mathcal{H}$ to $\mathcal{M}$ and $\mathcal{M}^T$ to obtain a matrix $M \in \mathcal{M}$ with a positive j-th column and another matrix $M' \in \mathcal{M}^T$ with a positive j-th column as well, for some index j. Then $(M')^T \in \mathcal{M}$, and it has a positive j-th row. We then have that $M(M')^T \in \mathcal{M}^*$ is positive, and the length of the product $M(M')^T$ gives an upper bound on the exponent of $\mathcal{M}$.

5.4 Comparisons

In this section, we compare the Eppstein heuristic and the transpose graph one. For this goal we use the randomized algorithm for generating extremal automata described in the previous chapter for multiple values of n. For all couples of prime numbers i, j such that $ij \leqslant n_{max} = 57$, we generate 5 primitive sets using this algorithm. For each different value of n Figure 5.1 shows the mean of the value of the length of the product that gives a positive column of those 5 primitive sets (left) and the mean of the execution time (right), for both techniques. Figure 5.2 gives the mean of the approximation of the exponent for both techniques.

As we can see, the transpose graph heuristic gives a smaller upper bound on the n-RT than the Eppstein one. But the computation time needed to compute it grows a lot faster than the Eppstein's heuristic.

However, it seems that in some special cases, the transpose graph heuristic does not give a smaller bound than the Eppstein heuristic. Indeed, we tested both heuristics on a family of primitive matrix sets based on a slowly synchronizing automaton (we just change one of its zero entries to a 1, in order to make the set



(a) (b)

Figure 5.1: Mean of the value of the length of the product that gives a positive column of 5 primitive sets generated from the randomized algorithm for generating extremal automata (left) and the mean of the execution time (right), for the Eppstein's heuristic and the transpose graph one.

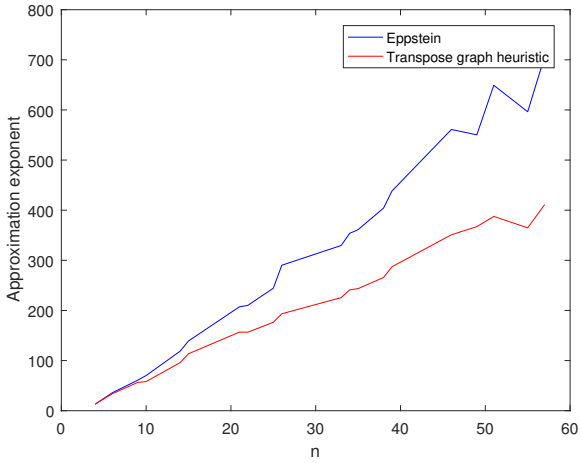


Figure 5.2: Mean of the approximation of the exponent of 5 primitive sets generated from the randomized algorithm for generating extremal automata for the Eppstein's heuristic and the transpose graph one.

$\mathcal{NZ}$ and primitive). The following family of matrices is as follows: $\mathcal{M} = \{A, B\}$,

where:

$$A_{i,j} = \begin{cases} 1 & \text{if } j = i(\text{mod n}) + 1 \quad \forall i \in \{1, ..., n\} \text{ or } i = n, j = 2 \\ 0 & \text{otherwise.} \end{cases}.$$

And,

$$B_{i,j} = \begin{cases} 1 & \text{if } j = i(\text{mod n}) + 1 \quad \forall i \in \{1, ..., n\} \\ 0 & \text{otherwise.} \end{cases}.$$

In this case, both heuristics seem to give the same bound for any value of n.

Conclusion

At the end of this manuscript we would like to point out that the theory of primitive matrix sets seems to be quite closely related to the theory of synchronizing automata, and allows us to leverage results from one theory to attack problems in the other theory. This was seen for example in the exploitation of the Protasov-Voynov theorem in the conception of algorithm $\mathcal{C}$ to generate slowly synchronizing automata, or in the use of synchronization algorithms like Eppstein's to find positive products of primitive sets. Studying the two simultaneously seems therefore to be a profitable endeavour, and we hope that some of the results developed here, notably those of Chapter 3 can be used in this fashion. We conclude our thesis with a short summary of our results.

- In Chapter 3 we proved a link between the k-rendezvous times of primitive matrix sets and associated automata. We also showed how the study of the pair digraph of NZ primitive matrix sets allows us to prove bounds on the k-RT of these sets. We believe our results in this section may serve to shed light on the synchronization process by analyzing what classes of automata satisfy the conditions required in proofs of the kind we present (based on the pair digraph) of an asymptotic linear bound on the k-RT for fixed k. Moreover, in the two last sections, we proposed some methods of improving the bound on the k-RT we found and the arguments we presented there can be developed further, and we proposed some ways to do so.

- In Chapter 4 we showed how a graph-theoretical analysis of the associated graphs of the primitive matrix sets produced by algorithm $\mathcal{C}$ could be used to prove nontrivial results about the kind of matrices these sets may contain. We also showed how a matrix-oriented approach studying the structure of these matrices over the intersection partitions elucidates the notion of compatibility, and enables us to derive necessary and sufficient conditions on partitions compatible with these matrices. Finally, we note that combinatorial analysis and probability theory allowed us to derive results on the expected behaviour of these matrices.

Although we only inspected the problem in detail for a particular case, we believe that these same methods can be employed to study the behaviour of $\mathcal{C}$ in more general scenarios. It may be possible that a more in-depth analysis would allow us to derive a better method of choosing permutations than the ones we investigated. In any case, sampling from a distribution other than the uniform distribution indeed seems advisable given the results of Section 4.2.

- Finally, in Chapter 5 we developed a new heuristic for approximating the k-RT of NZ primitive sets, and showed that it works quite well, at least for the sets produced by $\mathcal{C}$. Our new heuristic is slower than the Eppstein heuristic, but this is to be expected, given that it does, in general seem to give a better approximation. Indeed, for large n the difference can be fairly significant.

  Based on these heuristics, we also presented a new heuristic for approximating the exponent of NZ primitive sets. Heuristics for approximating the exponent of a primitive matrix set are, to the best of our knowledge, unknown in the literature.

  We note however that it is unlikely that the transpose-graph heuristic would perform as well for more general classes of automata/matrix sets than the ones tested, for three reasons:

  - The method is to some extent based on trial-and-error, and this seems (based on some preliminary numerical experiments not included here) to be less costly (in terms of the length of the final reset word found) for such sets than in the general case.
  - For automata, where one of the transition matrices (say M) contains a zero column, we have nodes in the transpose graph that do not map to any node under M reducing the number of ways to increase the row weight of the transposed product we're working with at a given iteration.
  - For matrix sets with multiple similar letters, there can be exponentially many labelled sequences corresponding to the same path in the transpose graph, and since the algorithm checks each of them until it finds a path that increases the weight of the row it's working on, this can be very slow. This problem is easily avoided however by imposing a limit on the maximum number of labelled sequences tried corresponding to a given path, making the algorithm resort to using an Eppstein step at this iteration.

It seems likely that by incorporating some sort of statistical heuristic governing the order in which labelled sequences corresponding to a given path are chosen (say by giving lesser preference to sequences containing a transition matrix with fewer 1s) the behaviour of our heuristic could be improved as regards both time complexity and accuracy of estimation, this is left for future work. Another direction worth exploring is to use the transpose square-graph rather than the transpose graph, this heuristic can be expected to be slow, but preliminary numerical investigations suggest that this approach is quite promising with regards to estimation quality. We could not, regrettably, include those results here.

We hope that both our presentation of these exciting problems and our results will have been of interest to our readers, and are extremely grateful to the latter for their patience in the reading of this manuscript. We also apologize for any accidental mistakes/lack of clarity/dryness in the quality of our writing, and look forward to the opportunity to learn from any comments and criticism on their part.

Bibliography

[1] Al'pin, Y.A., Al'pina, V.S.: Combinatorial properties of irreducible semigroups of nonnegative matrices. J. of Mathematical Sciences **191**(1), 4–9 (2013)

[2] Al'pin, Y.A., Al'pina, V.S.: Combinatorial properties of entire semigroups of nonnegative matrices. J. of Math. Sci. **207**(5), 674–685 (2015)

[3] Ananichev, D.S., Gusev, V.V.: Approximation of reset thresholds with greedy algorithms. Fundamenta Informaticae **145**(3), 221–227 (2016)

[4] Ananichev, D.S., Volkov, M.V., Gusev, V.V.: Primitive digraphs with large exponents and slowly synchronizing automata. J. of Math. Sci. **192**(3), 263–278 (2013)

[5] Azfar, U., Catalano, C., Charlier, L., Jungers, R.M.: A linear bound on the k-rendezvous time for primitive sets of NZ matrices. CoRR **abs/1903.10421** (2019), `http://arxiv.org/abs/1903.10421`

[6] Benenson, Y., Adar, R., Paz-Elizur, T., Livneh, Z., Shapiro, E.: Dna molecule provides a computing machine with both data and fuel. Proceedings of the National Academy of Sciences **100**(5), 2191–2196 (2003)

[7] Blondel, V., Jungers, R.M., Olshevsky, A.: On primitivity of sets of matrices. Automatica **61**, 80–88 (2015)

[8] Bowles, S.: Technical change and the profit rate: a simple proof of the Okishio theorem. Cambridge Journal of Economics **5**(2), 183–186 (1981)

[9] Braich, R.S., Chelyapov, N., Johnson, C., Rothemund, P.W.K., Adleman, L.: Solution of a 20-variable 3-sat problem on a dna computer. Science **296**(5567), 499–502 (2002)

[10] Büchi, J.R.: Weak second-order arithmetic and finite automata. Mathematical Logic Quarterly **6**(1-6), 66–92 (1960)

[11] Catalano, C., Jungers, R.M.: On randomized generation of slowly synchronizing automata. In: Mathematical Foundations of Computer Science. pp. 48:1–48:21 (2018)

[12] Chomsky, N.: Three models for the description of language. Journal of Symbolic Logic **23**(1), 71–72 (1958)

[13] Eppstein, D.: Reset sequences for monotonic automata. SIAM J. on Computing **19**(3), 500–510 (1990)

[14] Frankl, P.: An extremal problem for two families of sets. European J. of Combinatorics **3**(3), 125 – 127 (1982)

[15] Frobenius, G.: Über matrizen aus nicht negativen elementen. Berl.Ber. p. 456–477 (1912)

[16] Golomb, S.W., Gaal, P.: On the number of permutations on n objects with greatest cycle length k. Advances in Applied Mathematics **20**(1), 98–107 (1998)

[17] Gonze, F., M. Jungers, R.: On the synchronizing probability function and the triple rendezvous time for synchronzing automata. SIAM Journal on Discrete Mathematics **30** (2014)

[18] Hartfiel, D.J.: Nonhomogeneous matrix products. World Scientific Publishing (2002)

[19] Hennion, H.: Limit theorems for products of positive random matrices. The Annals of Probability **25**(4), 1545–1587 (1997)

[20] Kari, J.: A counter example to a conjecture concerning synchronizing words in finite automata. Bulletin of the EATCS **73**, 146 (2001)

[21] Kari, J.: Synchronizing finite automata on eulerian digraphs. Theoretical Computer Science **295**(1), 223 – 232 (2003)

[22] Kisielewicz, A., Kowalski, J., Szykuła, M.: Computing the shortest reset words of synchronizing automata. Journal of Combinatorial Optimization **29**(1), 88–124 (2015)

[23] Kleene, S.C.: Representation of events in nerve nets and finite automata. Automata Studies (C. E. Shannon and J. Mc Carty, Eds), Princeton University Press, p. 3–41 (1956)

[24] Kowalski, J., Szykula, M.: A new heuristic synchronizing algorithm. CoRR **abs/1308.1978** (2013), `http://arxiv.org/abs/1308.1978`

[25] Kudłacik, R., Roman, A., Wagner, H.: Effective synchronizing algorithms. Expert Syst. Appl. **39**(14), 11746–11757 (2012)

[26] Laemmel, A., Rudner, B.: Study of the application of coding theory. Report PIBEP-69-034, Polytechnic Inst. Brooklyn, Dept. Electrophysics, Farmingdale (1969)

[27] McCulloch, W.S., Pitts, W.H.: A logical calculus of the ideas immanent in nervous activity. Bulletin of Mathematical Biophysics **5**(4), 115–133 (1943)

[28] Meyer, C.D.: Matrix analysis and applied linear algebra. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA pp. 683–702 (2000)

[29] Natarajan, B.K.: An algorithmic approach to the automated design of parts orienters. In: SFCS. pp. 132–142 (1986)

[30] O Rabin, M., Scott, D.: Finite automata and their decision problems. IBM Journal of Research and Development **3**, 114–125 (1959)

[31] Perron, O.: Zur theorie der matrices. Mathematische Annalen **64**(2), 248–263 (1907)

[32] Pin, J.E.: On two combinatorial problems arising from automata theory. In: International Colloquium on Graph Theory and Combinatorics. vol. 75, pp. 535–548 (1983)

[33] Protasov, V.Y.: Invariant functions for the lyapunov exponents of random matrices. Sbornik: Mathematics **202**(1), 101 (2011)

[34] Protasov, V.Y., Voynov, A.S.: Sets of nonnegative matrices without positive products. Linear Algebra and its Applications **437**, 749–765 (2012)

[35] Roman, A.: Genetic algorithm for synchronization pp. 684–695 (2009)

[36] Roman, A.: Synchronizing finite automata with short reset words. Applied Mathematics and Computation **209**, 125–136 (2009)

[37] Schneider, H.: Wielandt's proof of the exponent inequality for primitive nonnegative matrices. Linear Algebra and its Applications **353**(1), 5 – 10 (2002)

[38] Schützenberger, M.: On the synchronizing properties of certain prefix codes. Information and Control **7**(1), 23–36 (1964)

[39] Seneta, E.: Non-negative Matrices and Markov Chains. Springer-Verlag New York, 2 edn. (1981)

[40] Seneta, E.: Non-negative Matrices and Markov Chains. Springer Series in Statistics, Springer New York (2006)

[41] Steinberg, B.: The averaging trick and the Černý conjecture. In: Developments in Language Theory. pp. 423–431 (2010)

[42] Szykuła, M.: Improving the upper bound the length of the shortest reset words. In: Symposium on Theoretical Aspects of Computer Science. vol. 96, pp. 56:1–56:16 (2018)

[43] Tannenbaum, M., Fischler, M.A.: Synchronizing and representation problems for sequential machines with masked outputs. In: 2013 IEEE 54th Annual Symposium on Foundations of Computer Science. pp. 97–103 (1970)

[44] Trahtman, A.N.: An efficient algorithm finds noticeable trends and examples concerning the cerny conjecture. CoRR **abs/0709.1197** (2007)

[45] Turing, A.M.: On computable numbers, with an application to the entscheidungsproblem. Proceedings of the London Mathematical Society **s2-42**(1), 230–265 (1937)

[46] Černý, J.: Poznámka k homogénnym eksperimentom s konečnými automatami. Matematicko-fysikalny Casopis SAV **14**(14), 208–216 (1964)

[47] Černý, J., Piricka, A., Rosenaueriva, B.: On directable automata. Kybernetika p. 289–298 (1971)

[48] Volkov, M.V.: Synchronizing automata preserving a chain of partial orders. In: Int. Conf. on Implementation and Application of Automata. pp. 27–37 (2007)

[49] Volkov, M.V.: Synchronizing automata and the Černý conjecture. In: Language and Automata Theory and Applications. pp. 11–27 (2008)

UNIVERSITÉ CATHOLIQUE DE LOUVAIN
École polytechnique de Louvain
Rue Archimède, 1 bte L6.11.01, 1348 Louvain-la-Neuve, Belgique | **www.uclouvain.be/epl**