On Randomized Generation of Slowly Synchronizing Automata

Costanza Catalano
Gran Sasso Science Institute
Viale Francesco Crispi 7, L'Aquila, Italy
costanza.catalano@gssi.it

Raphaël M. Jungers[1]
ICTEAM Institute, UCLouvain
Avenue Georges Lemaîtres 4-6, Louvain-la-Neuve, Belgium
raphael.jungers@uclouvain.be

Abstract

Motivated by the randomized generation of slowly synchronizing automata, we study automata made of permutation letters and a merging letter of rank $n-1$. We present a constructive randomized procedure to generate synchronizing automata of that kind with (potentially) large alphabet size based on recent results on *primitive* sets of matrices. We report numerical results showing that our algorithm finds automata with much larger reset threshold than a mere uniform random generation and we present new families of automata with reset threshold of $\Omega(n^2/4)$. We finally report theoretical results on randomized generation of primitive sets of matrices: a set of permutation matrices with a 0 entry changed into a 1 is primitive and has exponent of $O(n \log n)$ with high probability in case of uniform random distribution and the same holds for a random set of binary matrices where each entry is set, independently, equal to 1 with probability p and equal to 0 with probability $1-p$, when $np - \log n \to \infty$ as $n \to \infty$.

2012 ACM Subject Classification Mathematics of computing → Combinatorics, Mathematics of computing → Random graphs, Theory of computation → Randomness, geometry and discrete structures

Keywords and phrases Synchronizing automata, random automata, Černý conjecture, automata with simple idempotents, primitive sets of matrices

Digital Object Identifier 10.4230/LIPIcs.MFCS.2018.48

Acknowledgements The authors thank François Gonze and Vladimir Gusev for significant suggestions and fruitful discussions on the topic.

1 Introduction

A (complete deterministic finite) *automaton* $\mathcal{A}$ on n states can be defined as a set of m binary row-stochastic[2] matrices $\{A_1, \ldots, A_m\}$ that are called the *letters* of the automaton. We say that $\mathcal{A}$ is *synchronizing* if there exists a product of its letters, with repetitions allowed, that has an all-ones column[3] and the length of the shortest of these products is called the *reset*

[1] R. M. Jungers is a FNRS Research Associate. He is supported by the French Community of Belgium, the Walloon Region and the Innoviris Foundation.

[2] A *binary* matrix is a matrix with entries in $\{0, 1\}$. A *row-stochastic* matrix is a matrix with nonnegative entries where the entries of each row sum up to 1. Therefore a matrix is *binary* and *row-stochastic* if each row has exactly one 1.

[3] A column whose entries are all equal to 1.

Table 1 Table on upper bounds on the reset threshold for some classes of automata and examples of automata with large reset threshold belonging to these classes, up to date.

Classes	**Upper b. on rt**	**Families with quadratic rt**
Eulerian automata	n^2-3n+3 Kari [16]	$(n^2-3)/2$ Szykuła and Vorel [27] (4 letters)
Automata with full transition monoid	$2n^2-6n+5$ Gonze et. al. [14]	$n(n-1)/2$ Gonze et. al. [14] (n letters)
One cluster automata	n^2-7n+7 Béal et al. [4]	$(n-1)^2$ Černý [28] (2 letters)
Strongly connected weakly monotone automata	$\lfloor n(n+1)/6 \rfloor$ Volkov [29]	?
Automata with simple idempotents	$2(n-1)^2$ Rystov [25]	$(n-1)^2$ Černý [28] (2 letters) $\geq (n^2+3n-6)/4$ for $n=4k+3$ [Conjectured $(n^2-1)/2$] $\geq (n^2+3n-8)/4$ for $n=4k+1$ [Conjectured $(n^2-1)/2$] $\geq (n^2+2n-4)/4$ for $n=4k$ [Conjectured $(n^2-2)/2$] $\geq (n^2+2n-12)/4$ for $n=4k+2$ [Conjectured $(n^2-10)/2$] **Our contribution** (3 letters)

threshold ($rt(\mathcal{A})$) of the automaton. In other words, an automaton is synchronizing if there exists a word that brings the automaton into a particular state, regardless of the initial one. Synchronizing automata appear in different research fields; for example they are often used as models of error-resistant systems [10, 7] and in symbolic dynamics [18]. For a brief account on synchronizing automata and their other applications we refer the reader to [30]. The importance of synchronizing automata also arises from one of the most longstanding open problems in this field, the Černý conjecture, which affirms that any synchronizing automaton on n states has reset threshold at most $(n-1)^2$. If it is true, the bound is sharp due to the existence of a family of 2-letter automata attaining this value, family discovered by Černý in [28]. Despite great effort, the best upper bound for the reset threshold known so far is $(15617n^3+7500n^2+9375n-31250)/93750$, recently obtained by Szykuła in [26] and thereby beating the 30 years-standing upper bound of $(n^3-n)/6$ found by Pin and Frankl in [11, 22]. Better upper bounds have been obtained for certain families of automata and the search for automata attaining quadratic reset threshold within these families have been the subject of several contributions in recent years. These results are (partly) summarized in Table 1.
Exhaustive search confirmed the conjecture for small values of n (see [3, 9]). The hunt for a possible counterexample to the conjecture turned out not to be an easy task as well; the search space is wide and calculating the reset threshold is computationally hard (see [10, 21]). Automata with reset thresholds close to $(n-1)^2$, called *extremal* or *slowly synchronizing* automata, are also hard to detect and not so many families are known; Bondt et. al. [9] make a thorough analysis of automata with small number of states and we recall, among others, the families found by Ananichev et al. [3], by Gusev and Pribavkina [15], by Kisielewicz and Szykuła [17] and by Dzyga et. al. [19]. These last two examples are, in particular, some

of the few examples of slowly synchronizing automata with more than two letters that can be found in the literature. Almost all the families of slowly synchronizing automata listed above are closely related to the Černý automaton $\mathcal{C}(n) = \{a, b\}$, where a is the cycle over n vertices and b the letter that fixes all the vertices but one, which is mapped to the same vertex as done by a; indeed all these families present a letter that is a cycle over n vertices and the other letters have an action similar to the one of letter b. As these examples seem to have a quite regular structure, it is natural to wonder whether a randomized procedure to generate automata could obtain less structured automata with possibly larger reset thresholds. This probabilistic approach can be rooted back to the work of Erdős in the 60's, where he developed the so-called *Probabilistic Method*, a tool that permits to prove the existence of a structure with certain desired properties by defining a suitable probabilistic space in which to embed the problem; for an account on the probabilistic method we refer the reader to [1]. The simplest way to randomly generate an automaton of m letters is to uniformly and independently sample m binary row-stochastic matrices: unfortunately, Berlinkov first proved in [5] that two uniformly sampled random binary row-stochastic matrices synchronize with high probability (i.e. the probability that they form a synchronizing automaton tends to 1 as the matrix dimension tends to infinity), then Nicaud showed in [20] that they also have reset threshold of order $O(n \log^3 n)$ with high probability. We say that an automaton is *minimally synchronizing* if any proper subset of its letters is not synchronizing; what just presented before implies that a uniformly sampled random automaton of m letters has low reset threshold and is *not* minimally synchronizing with high probability. Summarizing:

- slowly synchronizing automata cannot be generated by a mere uniform randomized procedure;
- minimally synchronizing automata with more than 2 letters are especially of interest as they are hard to find and they do not appear often in the literature, so the behaviour of their reset threshold is still unclear.

With this motivation in place, our paper tackles the following questions:

Q1 Is there a way to randomly generate (minimally) slowly synchronizing automata (with more than two letters)?

Q2 Can we find some automata families with more than two letters, quadratic reset threshold and that do not resemble the Černý family?

Our Contribution. In this paper we give positive answers to both questions **Q1** and **Q2**. For the first one, we rely on the concept of *primitive* set of matrices, introduced by Protasov and Voynov in [24]: a finite set of matrices with nonnegative entries is said to be *primitive* if there exists a product of these matrices, with repetitions allowed, with all positive entries. A product of this kind is called *positive* and the length of the shortest positive product of a primitive set $\mathcal{M}$ is called the *exponent* ($exp(\mathcal{M})$) of the set. Although the Protasov-Voynov primitivity has gained a lot of attention in different fields as in stochastic switching systems [23] and consensus for discrete-time multi-agent systems [8], we are interested in its connection with automata theory. In the following, we say that a matrix is *NZ* if it has neither zero-rows nor zero-columns[4]; a matrix set is said to be *NZ* if all its matrices are NZ.

▶ **Definition 1.** Let $\mathcal{M} = \{M_1, \ldots, M_m\}$ be a binary NZ-matrix set. The *automaton associated to* the set $\mathcal{M}$ is the automaton $\mathcal{A}(\mathcal{M})$ whose letters are all the binary row-stochastic matrices that are entrywise not greater than at least one matrix in $\mathcal{M}$.

[4] Thus a NZ-matrix must have a positive entry in every row and in every column.

Figure 1 The automata $\mathcal{A}(\mathcal{M})$ and $\mathcal{A}(\mathcal{M}^T)$ of Example 2.

▶ **Example 2.** We here provide an example of a primitive set $\mathcal{M} = \{M_1, M_2\}$ and the associated automata $\mathcal{A}(\mathcal{M})$ and $\mathcal{A}(\mathcal{M}^T)$ in both their matrix and graph representations (Figure 1), where $\mathcal{M}^T = \{M_1^T, M_2^T\}$.

$$\mathcal{M} = \left\{ \begin{pmatrix} 0&1&0\\1&0&0\\0&0&1 \end{pmatrix}, \begin{pmatrix} 1&0&1\\0&0&1\\0&1&0 \end{pmatrix} \right\}, \mathcal{A}(\mathcal{M}) = \left\{ \begin{pmatrix} 0&1&0\\1&0&0\\0&0&1 \end{pmatrix}, \begin{pmatrix} 1&0&0\\0&0&1\\0&1&0 \end{pmatrix}, \begin{pmatrix} 0&0&1\\0&0&1\\0&1&0 \end{pmatrix} \right\} = \{a, b, c\},$$

$$\mathcal{A}(\mathcal{M}^T) = \left\{ \begin{pmatrix} 0&1&0\\1&0&0\\0&0&1 \end{pmatrix}, \begin{pmatrix} 1&0&0\\0&0&1\\0&1&0 \end{pmatrix}, \begin{pmatrix} 1&0&0\\0&0&1\\1&0&0 \end{pmatrix} \right\} = \{a, b, c'\}.$$

The following theorem summarizes two results proved by Blondel et. al. ([6], Theorems 16-17) and a result proved by Gerencsér et al. ([12], Theorem 8). Note that we state it for sets of *binary* NZ-matrices but it more generally holds for any set of NZ-matrices with nonnegative entries; this relies on the fact that in the notion of primitivity what counts is the position of the nonnegative entries within the matrices of the set and not their the actual values. In this case we should add to Definition 1 the request of setting to 1 all the positive entries of the matrices of $\mathcal{M}$ before building $\mathcal{A}(\mathcal{M})$.

▶ **Theorem 3.** *Let $\mathcal{M} = \{M_1, \ldots, M_m\}$ a set of binary NZ-matrices of size $n \times n$ and $\mathcal{M}^T = \{M_1^T, \ldots, M_m^T\}$. It holds that $\mathcal{M}$ is primitive if and only if $\mathcal{A}(\mathcal{M})$ (equiv. $\mathcal{A}(\mathcal{M}^T)$) is synchronizing. If $\mathcal{M}$ is primitive, then it also holds that:*

$$\max\Big\{rt\big(\mathcal{A}(\mathcal{M})\big), rt\big(\mathcal{A}(\mathcal{M}^T)\big)\Big\} \leq exp(\mathcal{M}) \leq rt\big(\mathcal{A}(\mathcal{M})\big) + rt\big(\mathcal{A}(\mathcal{M}^T)\big) + n - 1. \tag{1}$$

▶ **Example 4.** For the matrix set $\mathcal{M}$ defined in Example 2, it holds that $exp(\mathcal{M}) = 8$, $rt\big(\mathcal{A}(\mathcal{M})\big) = 4$ and $rt\big(\mathcal{A}(\mathcal{M}^T)\big) = 2$.

Theorem 3 will be extensively used throughout the paper. It shows that primitive sets can be used for generating synchronizing automata and Equation (1) tells us that the presence of a primitive set with large exponent implies the existence of an automaton with large reset threshold; in particular the discovery of a primitive set $\mathcal{M}$ with $exp(\mathcal{M}) \geq 2(n-1)^2 - n + 1$ would disprove the Černý conjecture. On the other hand, the upper bounds on the automata reset threshold mentioned before imply that $exp(\mathcal{M}) = O(n^3)$.
One advantage of using primitive sets is the Protasov-Voynov characterization theorem (see Theorem 6 in Section 2) that describes a combinatorial property that a NZ-matrix set must have in order *not* to be primitive: by constructing a primitive set such that each of its proper subsets has this property, we can make it *minimally primitive*[5].
We decided to focus our attention on what we call *perturbed permutation* sets, i.e. sets made of permutation matrices (binary matrices having exactly one 1 in every row and in every column) where a 0-entry of one of these matrices is changed into a 1. These sets have many interesting properties:

[5] Thus a *minimally primitive* set is a primitive set that does not contain any proper primitive subset.

- they have the least number of positive entries that a NZ-primitive set can have, which intuitively should lead to sets with large exponent;
- the associated automaton $\mathcal{A}(\mathcal{M})$ of a perturbed permutation set $\mathcal{M}$ can be easily computed. It is made of permutation letters and a letter of rank $n-1$ and its alphabet size is just one unit more than the cardinality of $\mathcal{M}$;
- if the matrix set $\mathcal{M}$ is minimally primitive, the automaton $\mathcal{A}(\mathcal{M})$ is minimally synchronizing (or it can be made minimally synchronizing by removing one known letter, as shown in Proposition 9);
- primitivity is easily checked by the Protasov-Voynov algorithm ([24], Proposition 2), and primitivity of $\mathcal{M}$ assures that $\mathcal{A}(\mathcal{M})$ is synchronizing (Theorem 3).

The characterization theorem for primitive sets and the above properties are the main ingredients of our randomized algorithm that finds minimally synchronizing automata of 3 and 4 letters (and can eventually be generalized to m letters); to the best of our knowledge, this is the first time where a constructive procedure for finding minimally synchronizing automata is presented. This is described in Section 3 where numerical results are reported. The random construction let us also find new families of 3-letters automata, presented in Section 4, with reset threshold $\Omega(n^2/4)$ and that do not resemble the Černý automaton, thus answering question **Q2**. Finally, in Section 5 we extend a result on perturbed permutation sets obtained by Gonze et al. in [13]: we show that a random perturbed permutation set is primitive with high probability for any matrix size n (and not just when n is a prime number as in [13]) and that its exponent is of order $O(n \log n)$ still with high probability. A further generalization is then presented for sets of random binary matrices: if each entry of each matrix is set to 1 with probability p and to 0 with probability $1-p$, independently from each other, then the set is primitive and has exponent of order $O(n \log n)$ with high probability for any p such that $np - \log n \to \infty$ as $n \to \infty$.
The proofs of the results presented in this paper have been omitted due to length restrictions.

2 Definitions and notation

In this section we briefly go through some definitions and results that will be needed in the rest of the paper.
We indicate with $[n]$ the set $\{1, \dots, n\}$ and with S_k the set of permutations over k elements; with a slight abuse of notation S_k will also denote the set of the $k \times k$ permutation matrices. An n-state automaton $\mathcal{A} = \{A_1, \dots, A_m\}$ can be represented by a labelled digraph on n vertices with a directed edge from vertex i to vertex j labelled by A_k if $A_k(i,j) = 1$; in this case we also use the notation $iA_k = j$. We remind that a matrix M is *irreducible* if there does not exist a permutation matrix P such that PMP^T is block-triangular; a set $\{M_1, \dots, M_m\}$ is said to be *irreducible* iff the matrix $\sum_{i=1}^m M_i$ is irreducible. The *directed graph associated to* an $n \times n$ matrix M is the digraph D_M on n vertices with a directed edge from i to j if $M(i,j) > 0$. A matrix M is irreducible if and only if D_M is strongly connected, i.e. if and only if there exists a directed path between any two given vertices. A *primitive* set $\mathcal{M}$ is a set of m matrices $\{M_1, \dots, M_m\}$ with nonnegative entries where there exists a product $M_{i_1} \cdots M_{i_l} > 0$ entrywise, for $i_1, \dots, i_l \in [m]$. The length of the shortest of these products is called the *exponent* ($exp(\mathcal{M})$) of the set. Irreducibility is a necessary (but not sufficient) condition for a matrix set to be primitive (see [24], Section 1). Primitive sets of NZ-matrices can be characterized as follows:

▶ **Definition 5.** Let $\Omega = \dot{\bigcup}_{l=1}^{k} \Omega_l$ be a partition of $[n]$ with $k \geq 2$. We say that an $n \times n$ matrix M has a *block-permutation structure on the partition* Ω if there exists a permutation $\sigma \in S_k$ such that $\forall\, l = 1, \ldots, k$ and $\forall\, i \in \Omega_l$, if $M(i,j) > 0$ then $j \in \Omega_{\sigma(l)}$. We say that a set of matrices *has a block-permutation structure* if there exists a partition on which *all* the matrices of the set have a block-permutation structure.

▶ **Theorem 6** ([24], Theorem 1)**.** *An irreducible set of NZ matrices of size $n \times n$ is* not *primitive if and only if the set has a block-permutation structure.*

We end this section with the last definition and our first observation (Proposition 8).

▶ **Definition 7.** A matrix A *dominates* a matrix B if $A(i,j) \geq B(i,j),\ \forall\, i,j$.

▶ **Proposition 8.** *Consider an irreducible set $\{M_1, \ldots, M_m\}$ in which every matrix dominates a permutation matrix. If the set has a block-permutation structure, then all the blocks of the partition must have the same size.*

3 Minimally primitive sets and minimally synchronizing automata

In this section we focus on *perturbed permutation* sets, i.e. matrix sets made of permutation matrices where a 0-entry of one matrix is changed into a 1. We represent a set of this kind as $\mathcal{M} = \{P_1, \ldots, P_{m-1}, P_m + \mathbb{I}_{i,j}\}$, where $P_1, \ldots, P_m$ are permutation matrices, $\mathbb{I}_{i,j}$ is a matrix whose (i,j)-th entry is equal to 1 and all the others entries are equal to 0 and $P_m(i,j') = 1$ for a $j' \neq j$. The first result states that we can easily generate minimally synchronizing automata starting from minimally primitive perturbed permutation sets:

▶ **Proposition 9.** *Let $\mathcal{M} = \{P_1, \ldots, P_{m-1}, P_m + \mathbb{I}_{i,j}\}$ be a minimally primitive perturbed permutation set and let $j' \neq j$ be the integer such that $P_m(i,j') = 1$. The automaton $\mathcal{A}(\mathcal{M})$ (see Definition 1) can be written as $\mathcal{A}(\mathcal{M}) = \{P_1, \ldots, P_{m-1}, P_m, M\}$ with $M = P_m + \mathbb{I}_{i,j} - \mathbb{I}_{i,j'}$. If $\mathcal{A}(\mathcal{M})$ is not minimally sychronizing, then $\bar{\mathcal{A}} = \{P_1, \ldots, P_{m-1}, M\}$ is.*

3.1 A randomized algorithm for constructing minimally primitive sets

If we want to find minimally synchronizing automata, Proposition 9 tells us that we just need to generate minimally primitive perturbed permutation sets; in this section we implement a randomized procedure to build them.

Theorem 6 says that a matrix set is *not* primitive if all the matrices share the same block-permutation structure, therefore a set of m matrices is minimally primitive iff every subset of cardinality $m-1$ has a block-permutation structure on a certain partition; this is the condition we will enforce. As we are dealing with perturbed permutation sets, Proposition 8 tells us that we just need to consider partitions with blocks of the same size; if the blocks of the partition have size n/q, we call it a *q-partition* and we say that the set has a *q-permutation structure*. Given $R, C \subset [n]$ and a matrix M, we indicate with $M(R,C)$ the submatrix of M with rows indexed by R and columns indexed by C. The algorithm first generates a set of permutation matrices satisfying the requested block-permutation structures and then a 0-entry of one of the obtained matrices is changed into a 1; while doing this last step, we will make sure that such change will preserve all the block-permutation structures of the matrix. We underline that our algorithm finds perturbed permutation sets that, *if* are primitive, are minimally primitive. Indeed, the construction itself only ensures minimality and not primitivity: this latter property has to be verified at the end.

3.1.1 The algorithm

For generating a set of m matrices $\mathcal{M} = \{M_1, \dots, M_m\}$ we choose m prime numbers $q_1 \geq \cdots \geq q_m \geq 2$ and we set $n = \prod_{i=1}^{m} q_i$. For $j = 1, \dots, m$, we require the set $\{M_1, \dots, M_{j-1}, M_{j+1}, \dots, M_m\}$ (the set obtained from $\mathcal{M}$ by erasing matrix M_j) to have a q_j-permutation structure; this construction will ensure the minimality of the set. More in detail, for all $j=1, \dots, m$ we will enforce the existence of a q_j-partition $\Omega_{q_j} = \dot{\bigcup}_{i=1}^{q_j} \Omega_i^j$ of $[n]$ on which, for all $k \neq j$, the matrix M_k has to have a block-permutation structure. As stated by Definition 5, this request means that for every $k = 1, \dots, m$ and for every $j \neq k$ there must exist a permutation $\sigma_j^k \in S_{q_j}$ such that for all $i=1, \dots, q_j$ and $l \neq \sigma_j^k(i)$, $M_k(\Omega_i^j, \Omega_l^j)$ is a zero matrix.

The main idea of the algorithm is to initialize every entry of each matrix to 1 and then, step by step, to set to 0 the entries that are not compatible with the conditions that we are requiring. As our final goal is to have a set of permutation matrices, at every step we need to make sure that each matrix dominates at least one permutation matrix, despite the increasing number of zeros among its entries.

▶ **Definition 10.** Given a matrix M and a q-partition $\Omega_q = \dot{\bigcup}_{i=1}^{q} \Omega_i^q$, we say that a permutation $\sigma \in S_q$ is *compatible* with M and Ω_q if for all $i = 1, \dots, q$, there exists a permutation matrix Q_i such that

$$M\left(\Omega_i^q, \Omega_{\sigma(i)}^q\right) \geq Q_i. \tag{2}$$

The algorithm itself is formally presented in Listing 1; we here describe in words how it operates. Each entry of each matrix is initialized to 1. The algorithm has two for-loops: the outer one on $j = 1, \dots, m$, where a q_j-partition $\Omega_{q_j} = \dot{\bigcup}_{i=1}^{q_j} \Omega_i^j$ of $[n]$ is uniformly randomly sampled and then the inner one on $k = 1, \dots, m$ with $k \neq j$ where we verify whether there exists a permutation $\sigma_j^k \in S_{q_j}$ that is compatible with M_k and Ω_{q_j}. If it does exist, then we choose one among all the compatible permutations and the algorithm moves to the next step $k + 1$. If such permutation does not exist, then the algorithm exits the inner for-loop and it randomly selects another q_j-partition Ω'_{q_j} of $[n]$; it then repeats the inner loop for $k = 1, \dots, m$ with $k \neq j$ with this new partition. If after $T1$ steps it is choosing a different q_j-partition Ω'_{q_j} the existence, for each $k \neq j$, of a permutation $\sigma_j'^k \in S_{q_j}$ that is compatible with M_k and Ω'_{q_j} is not established, we stop the algorithm and we say that *it did not converge*. If the inner for-loop is completed, then for each $k \neq j$ the algorithm modifies the matrix M_k by keeping unchanged each block $M_k\left(\Omega_i^j, \Omega_{\sigma_j^k(i)}^j\right)$ for $i = 1, \dots, q_j$ and by setting to zero all the other entries of M_k, where σ_j^k is the selected compatible permutation; the matrix M_k has now a block-permutation structure over the sampled partition Ω_{q_j}. The algorithm then moves to the next step $j + 1$. If it manages to finish the outer for-loop, we have a set of binary matrices with the desired block-permutation structures. We then just need to select a permutation matrix $P_k \leq M_k$ for every $k = 1, \dots, m$ and then to randomly change a 0-entry into a 1 in one of the matrices without modifying its block-permutation structures: this is always possible as the blocks of the partitions are non trivial and a permutation matrix has just n positive entries. We finally check whether the set is primitive.

Here below we present the procedures that the algorithm uses:

(a) $[p, P] = Extractperm(M, met)$

This is the key function of the algorithm. It returns $p=1$ if the matrix M dominates a permutation matrix, it returns $p=0$ and $P=M$ otherwise. In the former case it also returns a permutation matrix P selected among the ones dominated by M according to met; if $met = 2$ the matrix P is sampled uniformly at random, while if $met = 3$ we

make the choice of P deterministic. More in detail, the procedure works as follows: we first count the numbers of 1s in each column and in each row of the matrix M. We then consider the row or the column with the least number of 1s; if this number is zero we stop the procedure and we set $p = 0$, as in this case M does not dominate a permutation matrix. If this number is strictly greater than zero, we choose one of the 1-entries of the row or the column attaining this minimum: if $met = 2$ (**method 2**) the entry is chosen uniformly at random while if $met = 3$ (**method 3**) we take the first 1-entry in the lexicographic order. Suppose that such 1-entry is in position (i, j): we set to zero all the other entries in row i and column j and we iterate the procedure on the submatrix obtained from M by erasing row i and column j. We can prove that this procedure is well-defined and in at most n steps it produces the desired output: $p = 0$ if and only if M does not dominate a permutation matrix and, in case $p = 1$, method 2 indeed sample uniformly one of the permutations dominated by M, while method 3 is deterministic and the permutation obtained usually has its 1s distributed around the main diagonal. Method 3 will play an important role in our numerical experiments in Section 3.2 and in the discovery of new families of automata with quadratic reset threshold in Section 4.

(b) $[a, A] = DomPerm(M, \Omega, met)$
It returns $a = 1$ if there exists a permutation compatible with the matrix M and the partition $\Omega = \dot{\bigcup}_{i=1}^{q} \Omega_i^q$, it returns $a = 0$ and $A = M$ otherwise. In the former case it chooses one of the compatible permutations, say σ, according to met and returns the matrix A equal to M but the entries not in the blocks defined by (2) that are set to zero; A has then a block-permutation structure on Ω. More precisely, $DomPerm$ acts in two steps: it first defines a $q \times q$ matrix B such that, for all $i, k = 1, \ldots, q$,

$$B(i,k) = \begin{cases} 1 & \text{if } M(\Omega_i^q, \Omega_k^q) \text{ dominates a permutation matrix} \\ 0 & \text{otherwise} \end{cases};$$

this can be done by calling $ExtractPerm$ with input $M(\Omega_i^q, \Omega_k^q)$ and met for all $i, k = 1, \ldots, q$. At this point, asking if there exists a permutation compatible with M and Ω is equivalent of asking if B dominates a permutation matrix. Therefore, the second step is to call again $[p, P] = ExtractPerm(B, met)$: if $p = 0$ we set $a = 0$ and $A = M$, while if $p = 1$ we set $a = 1$ and A as described before with $\sigma = P$ (i.e. $\sigma(i) = j$ iff $P(i, j) = 1$); indeed the permutation P is one of the permutations compatible with M and Ω.

(c) $Mset = Addone(P_1, ..., P_m)$
It changes a 0-entry of one of the matrices $P_1, ..., P_m$ into a 1 preserving all its block-permutation structures. The matrix and the entry are chosen uniformly at random and the procedure iterates the choice till it finds a compatible entry (which always exists); it then returns the final perturbed permutation set $Mset$.

(d) $pr = Primitive(Mset)$
It returns $pr = 1$ if the matrix set $Mset = \{M_1, \ldots, M_m\}$ is primitive and $pr = 0$ otherwise. It first verifies if the set is irreducible by checking the strong connectivity of the digraph D_N where $N = \sum_{i=1}^{k} M_i$ (see Section 2) via breadth-first search on every node, then if the set is irreducible, primitivity is checked by the Protasov-Voynov algorithm ([24], Section 4).

All the above routines have polynomial time complexity in n, apart from routine $Primitive$ that has time complexity $O(mn^2)$.

▶ Remark.

1. In all our numerical experiments the algorithm always converged, i.e. it always ended before reaching the stopping value $T1$, for $T1$ large enough. This is probably due to the fact that the matrix dimension n grows exponentially as the number of matrices m increases, which produces enough degrees of freedom. We leave the proof of this fact for future work.

2. A recent work of Alpin and Alpina [2] generalizes Theorem 6 for the characterization of primitive sets to sets that are allowed to be reducible and the matrices to have zero columns but not zero rows. Clearly, automata fall within this category. Without going into many details (for which we refer the reader to [2], Theorem 3), Alpin and Alpina show that an n-state automaton is *not* synchronizing if and only if there exist a partition $\dot{\bigcup}_{j=1}^{s}\Omega_j$ of $[n]$ such that it has a block-permutation structure on a *subset* of that partition. This characterization is clearly less restrictive: it just suffices to find a subset $I \subset [s]$ such that for each letter A of the automaton there exists a permutation $\sigma \in S_I$ such that for all $i \in I$, if $A(i,j) = 1$ then $j \in \Omega_{\sigma(i)}$. Our algorithm could leverage this recent result in order to directly construct minimal synchronizing automata. We also leave this for future work.

Listing 1 Algorithm for finding minimally primitive sets.

```
Input: q_1,...,q_m,T1,met
Initialize M_1,...,M_m as all-ones matrices
for j:=1 to m do
        t1=0
        while t1<T1 do
                t1=t1+1
                choose a q_j-partition Omega_j
                for k=1 to m and k!=j do
                   [a,A_k]=DomPerm(M_k,Omega_j,met)
                   if a==0
                         exit inner for-loop
                   end
                end
                if a==1
                   exit while-loop
                end
        end
        if t1==T1
            display 'does not converge', exit procedure
        else
            set M_k=A_k for all k=1,...,m and k!=j
        end
end
for i:=1 to m do
    [p_i,P_i]=Extractperm(M_i,met)
end
Mset=Addone(P_1,...,P_m)
pr=Primitive(Mset)
return Mset, pr
```

3.2 Numerical results

We have seen that once we have a minimally primitive perturbed permutation set, it is easy to generate a minimally synchronizing automaton from it, as stated by Proposition 12. Our goal is to generate automata with large research threshold, but checking this property on many randomly generated instances is prohibitive. Indeed, we recall that computing the reset threshold of an automaton is in general NP-hard [10]. Instead, as a proxy for the reset threshold, we compute the *diameter of the square graph*, which we now introduce:

▶ **Definition 11.** The *square graph* $S(\mathcal{A})$ of an n-state automaton $\mathcal{A}$ is the labelled directed graph with vertex set $V=\{(i,j) : 1 \leq i \leq j \leq n\}$ and edge set E such that $e=\{(i,j),(i',j')\} \in E$ if there exists a letter $A \in \mathcal{A}$ such that $A(i,i') > 0$ and $A(j,j') > 0$, or $A(i,j') > 0$ and $A(j,i') > 0$. In this case, we label the directed edge e by A (multiple labels are allowed). A vertex of type (i,i) is called a *singleton*.

A well-known result ([30], Proposition 1) states that an automaton is synchronizing if and only if in its square graph there exists a path from any non-singleton vertex to a singleton one; the proof of this fact also implies that

$$diam\big(S(\mathcal{A})\big) \leq rt(\mathcal{A}) \leq n \cdot diam\big(S(\mathcal{A})\big), \tag{3}$$

where $diam\big(S(\mathcal{A})\big)$ denotes the *diameter* of $S(\mathcal{A})$ i.e. the maximum length of the shortest path between any two given vertices, taken over all the pairs of vertices. The diameter can be computed in polynomial time, namely $O(mn^2)$ with m the number of letters of the automaton.

We now report our numerical results based on the diameter of the square graph. We compare three methods of generating automata: we call **method 1** the uniform random generation of 2-letter automata made of one permutation matrix and a matrix of rank $n-1$, while **method 2** and **method 3**, already introduced in the previous paragraph, refer to the different ways of extracting a permutation matrix from a binary one in our randomized construction. We set $T1 = 1000$ and for each method and each choice of n we run the algorithm $it(n) = 50n^2$ times, thus producing each time $50n^2$ sets. This choice for $it(n)$ has been made by taking into account two facts: on one hand, it is desirable to keep constant the rate $it(n)/k_m(n)$ between the number of sampled sets $it(n)$ and the cardinality $k_m(n)$ of the set of the perturbed permutation sets made of m matrices. Unfortunately, $k_m(n+1)/k_m(n)$ grows approximately as n^m and so $k_m(n)$ explodes very fast. On the other hand, we have to deal with the limited computational speed of our computers. The choice of $it(n) = 50n^2$ comes as a compromise between these two issues, at least when $n \leq 70$.
Among the $it(n)$ generated sets, we select the primitive ones and we generate their associated automata (Definition 1); we then check which ones are not minimally synchronizing and we make them minimally synchronizing by using Proposition 9. Finally, we compute the square graph diameter of all the minimally synchronizing automata obtained. Figure 2 reports on the y axis the maximal square graph diameter found for each method and for each matrix dimension n when n is the product of three prime numbers (left picture) and when it is a product of four prime numbers (right picture). We can see that our randomized construction manages to reach higher values of the square graph diameter than the mere random generation; in particular, method 3 reaches quadratic diameters in case of three matrices.

We also report in Figure 3 (left) the behavior of the *average* diameter of the minimally synchronizing automata generated on $50n^2$ iterations when n is the product of three prime numbers: we can see that in this case method 2 does not perform better than method 1, while method 3 performs just slightly better. This behavior could have been expected since our primary goal was to randomly generate *at least one* slowly synchronizing automata; this is indeed what happens with method 3, that manages to reach quadratic reset thresholds most of the times.
A remark can be done on the percentage of the generated sets that are *not* primitive; this is reported in Figure 3 (right), where we divide nonprimitive sets into two categories: reducible sets and *imprimitive* sets, i.e. irreducible sets that are not primitive. We can see that the percentage of nonprimitive sets generated by method 1 goes to 0 as n increases, behavior

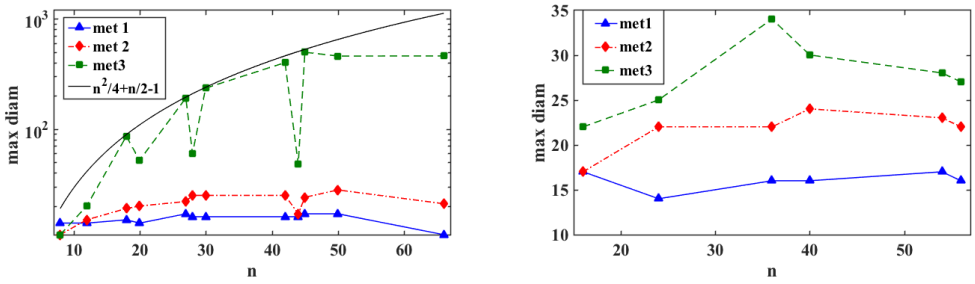

Figure 2 Comparison of our methods (met 2 and 3) with the naive method (met 1) with respect to the maximal diameter found on $50n^2$ iterations. Left: n is the product of three prime numbers; the y axis is in logarithmic scale. Right: n is the product of four prime numbers.

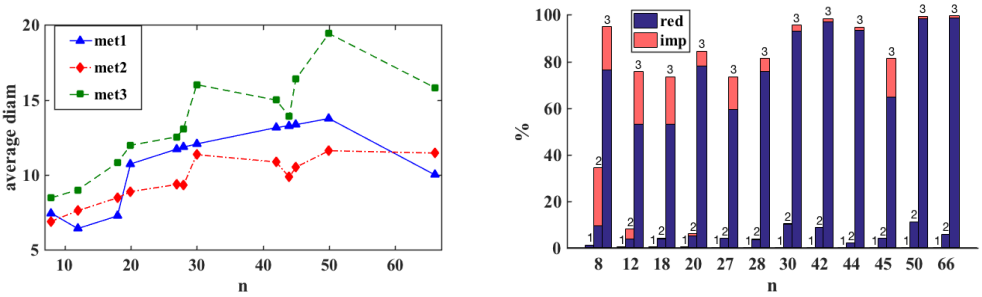

Figure 3 Left: Average diameter found by methods 1, 2 and 3 when n is the product of three prime numbers. Right: Percentage of nonprimitive sets (divided into reducible and imprimitive sets) generated by methods 1, 2 and 3 (indicated above each bar) when n is the product of three prime numbers. For instance, on sets of dimension $n = 20$, method 1 generates 0.35% of nonprimitive sets (0.35% reducible, 0% imprimitive), method 2 generates 6.15% of nonprimitive sets (5.18% reducible, 0.97% imprimitive) and method 3 generates 84.5% of nonprimitive sets (77.9% reducible, 6.6% imprimitive).

that we expected (see Section 5, Theorem 18), while method 2 seems to always produce a not negligible percentage of nonprimitive sets, although quite small. The behavior is reversed for method 3: most of the generated sets are not primitive. This can be interpreted as a good sign. Indeed, nonprimitive sets can be seen as sets with *infinite* exponent; as we are generating a lot of them with method 3, we intuitively should expect that, when a primitive set is generated, it has high chances to have large diameter.

The slowly synchronizing automata found by our randomized construction are presented in the following section. We believe that some parameters of our construction, as the way a permutation matrix is extracted from a binary one or the way the partitions of $[n]$ are selected, could be further tuned or changed in order to generate new families of slowly synchronizing automata; for example, we could think about selecting the ones in the procedure *Extractperm* according to a given distribution. We leave this for future work.

4 New families of automata with quadratic reset threshold

We present here four new families of (minimally synchronizing) 3-letter automata with square graph diameter of order $\Omega(n^2/4)$, which represents a lower bound for their reset threshold. These families are all made of two symmetric permutation matrices and a matrix of rank $n-1$ that merges two states and fixes all the others (a perturbed identity matrix): they thus

lie within the class of automata *with simple idempotents*, class introduced by Rystsov in [25] in which every letter A of the automaton is requested either to be a permutation or to satisfy $A^2 = A$. These families have been found via the randomized algorithm described in Section 3.1 using the deterministic procedure to extract a permutation matrix from a binary one (method 3). The following proposition shows that primitive sets made of a perturbed identity matrix and two symmetric permutations must have a very specific shape; we then present our families, prove closed formulas for their square graph diameter and finally state a conjecture on their reset thresholds. With a slight abuse of notation we identify a permutation matrix Q with its underlying permutation, that is we say that $Q(i) = j$ if and only if $Q(i, j) = 1$; the identity matrix is denoted by I. Note that a permutation matrix is symmetric if and only if its cycle decomposition is made of fixed points and cycles of length 2.

▶ **Proposition 12.** *Let $\mathcal{M}_{ij} = \{\bar{I}_{ij}, Q_1, Q_2\}$ be a matrix set of $n \times n$ matrices where $\bar{I}_{ij} = I + \mathbb{I}_{ij}$, $j \neq i$, is a perturbed identity and Q_1 and Q_2 are two symmetric permutations. If $\mathcal{M}$ is irreducible then, up to a relabelling of the vertices, Q_1 and Q_2 have the following form:*

- *if n is even*

$$Q_1(i) = \begin{cases} 1 & \text{if } i = 1 \\ i+1 & \text{if } i \text{ even}, 2 \leq i \leq n-2 \\ i-1 & \text{if } i \text{ odd}, 3 \leq i \leq n-1 \\ n & \text{if } i = n \end{cases}, \quad Q_2(i) = \begin{cases} i-1 & \text{if } i \text{ even} \\ i+1 & \text{if } i \text{ odd} \end{cases} \tag{4}$$

or

$$Q_1(i) = \begin{cases} n & \text{if } i = 1 \\ i+1 & \text{if } i \text{ even}, 2 \leq i \leq n-2 \\ i-1 & \text{if } i \text{ odd}, 3 \leq i \leq n-1 \\ 1 & \text{if } i = n \end{cases}, \quad Q_2(i) = \begin{cases} i-1 & \text{if } i \text{ even} \\ i+1 & \text{if } i \text{ odd} \end{cases} \tag{5}$$

- *if n is odd*

$$Q_1(i) = \begin{cases} 1 & \text{if } i = 1 \\ i+1 & \text{if } i \text{ even} \\ i-1 & \text{if } i \text{ odd}, 3 \leq i \leq n \end{cases}, \quad Q_2(i) = \begin{cases} i-1 & \text{if } i \text{ even} \\ i+1 & \text{if } i \text{ odd}, 1 \leq i \leq n-2 \\ n & \text{if } i = n \end{cases}. \tag{6}$$

▶ **Proposition 13.** *A matrix set $\mathcal{M}_{ij} = \{\bar{I}_{ij}, Q_1, Q_2\}$ of type (5) is never primitive.*

▶ **Definition 14.** We define $\mathcal{A}_{ij} = \{\underline{I}_{ij}, Q_1, Q_2\}$ to be the associated automaton $\mathcal{A}(\mathcal{M}_{ij})$ of $\mathcal{M}_{ij}$ (see Definition 1), where $\underline{I}_{ij} = I + \mathbb{I}_{ij} - \mathbb{I}_{ii}$.

It is clear that $\mathcal{A}_{ij}$ is with simple idempotents. Figure 4 represents $\mathcal{A}_{1,6}$ with $n = 8$. We set now $\mathcal{E}_n = \mathcal{A}_{1,n-2}$ for $n = 4k$ and $k \geq 2$, $\mathcal{E}'_n = \mathcal{A}_{1,n-4}$ for $n = 4k+2$ and $k \geq 2$, $\mathcal{O}_n = \mathcal{A}_{\frac{n-1}{2},\frac{n+1}{2}}$ for $n = 4k+1$ and $k \geq 1$, $\mathcal{O}'_n = \mathcal{A}_{\frac{n-1}{2},\frac{n+1}{2}}$ for $n = 4k+3$ and $k \geq 1$. The following theorem holds:

▶ **Theorem 15.** *The automaton $\mathcal{E}_n$ has square graph diameter (SGD) of $(n^2 + 2n - 4)/4$, $\mathcal{E}'_n$ has SGD of $(n^2 + 2n - 12)/4$, $\mathcal{O}_n$ has SGD of $(n^2 + 3n - 8)/4$ and $\mathcal{O}'_n$ has SGD of $(n^2 + 3n - 6)/4$. Therefore all the families $\mathcal{E}_n$, $\mathcal{E}'_n$, $\mathcal{O}_n$ and $\mathcal{O}'_n$ have reset threshold of $\Omega(n^2/4)$.*

Figure 5 represents the square graph of the automaton $\mathcal{E}_8$, where its diameter is colored in red. All the singletons but the one that belongs to the diameter have been omitted.

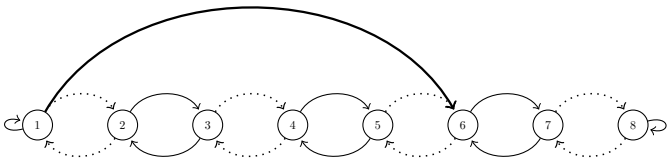

Figure 4 The automata $\mathcal{A}_{1,6}$ with $n = 8$; $rt(\mathcal{A}_{1,6})$=31. Dashed arrows refer to matrix Q_2, normal arrows to matrix Q_1 and bold arrows to matrix $\underline{I}_{1,6}$ where its selfloops have been omitted.

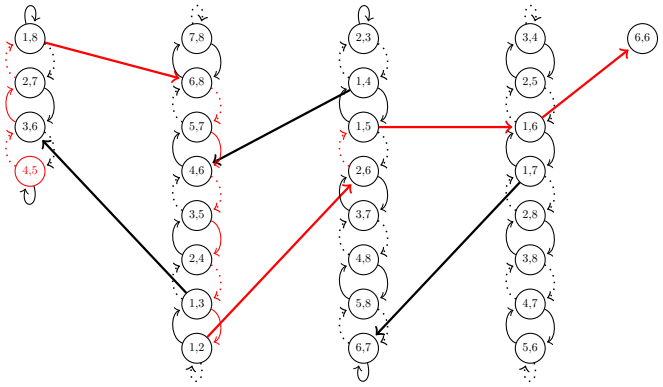

Figure 5 Square graph of automaton $\mathcal{E}_8$, $diam\big(S(\mathcal{E}_8)\big) = 19$. Normal arrows refer to matrix Q_1, dotted arrows to matrix Q_2 and bold arrows to matrix $\underline{I}_{1,6}$, where its selfloops have been omitted. The red path is the diameter.

▶ **Conjecture 16.** *The automaton $\mathcal{E}_n$ has reset threshold of $(n^2-2)/2$, $\mathcal{E}'_n$ has reset threshold of $(n^2-10)/2$ and $\mathcal{O}_n$ and $\mathcal{O}'_n$ have reset threshold of $(n^2-1)/2$. Furthermore, they represent the automata with the largest possible reset threshold among the family $\mathcal{A}_{ij}$ for respectively $n = 4k$, $n = 4k+2$, $n = 4k+1$ and $n = 4k+3$.*

We end this section by remarking that, despite the fact that the randomized construction for minimally primitive sets presented in Section 3 works just when the matrix size n is the product of at least three prime numbers, here we found an extremal automaton of quadratic reset threshold for *any* value of n.

5 Primitivity with high probability

We call *random perturbed permutation* set a perturbed permutation set of $m \geq 2$ matrices constructed with the following randomized procedure:

▶ **Procedure 17.**

1. *We sample m permutation matrices $\{P_1, \ldots, P_m\}$ independently and uniformly at random from the set S_n of all the permutations over n elements;*
2. *A matrix P_i is uniformly randomly chosen from the set $\{P_1, \ldots, P_m\}$. Then, one of its 0-entry is uniformly randomly selected among its 0-entries and changed into a 1. It becomes then a perturbed permutation matrix $\bar{P}_i$;*
3. *The final random perturbed permutation set is the set $\{P_1, \ldots, P_{i-1}, \bar{P}_i, P_{i+1}, \ldots, P_m\}$.*

This procedure is also equivalent to choosing independently and uniformly at random $m-1$ permutation matrices from S_n and one *perturbed* permutation matrix from $\bar{S}_n$ with $\bar{S}_n = \{\bar{P} : \bar{P} = P + \mathbb{I}_{i,j},\ P \in S_n,\ P(i,j') = 1,\ j \neq j'\}$. We say that a property X holds for a random matrix set with *high probability* if the probability that property X holds tends to 1 as the matrix dimension n tends to infinity.

▶ **Theorem 18.** *A random perturbed permutation set constructed via Procedure 17 is primitive and has exponent of order $O(n \log n)$ with high probability.*

Theorem 18 can be extended to random sets of binary matrices. It is clear that focusing just on binary matrices is not restrictive as in the definition of primitivity what counts is just the position of the positive entries within the matrices and not their actual values. Let $B(p)$ denote a random binary matrix where each entry is independently set to 1 with probability p and to 0 with probability $(1-p)$ and let $\mathcal{B}_m(p)$ denote a set of $m \geq 2$ matrices obtained independently in this way. Under some mild assumptions over p, we still have primitivity with high probability:

▶ **Theorem 19.** *For any fixed integer $m \geq 2$, if $np - \log n \longrightarrow \infty$ as $n \to \infty$, then $\mathcal{B}_m(p)$ is primitive and $exp\big(\mathcal{B}_m(p)\big) = O(n \log n)$ with high probability.*

It is interesting to compare this result with the one obtained by Gerencsér et al. in [12]: they prove that, if $exp(n)$ is the maximal value of the exponent among all the binary primitive sets of $n \times n$ matrices, then $\lim_{n\to\infty}(\log exp(n))/n = (\log 3)/3$. This implies that, for n big enough, there must exist some primitive sets whose exponent is close to $3^{n/3}$, but these sets must be very few as Theorem 19 states that they are almost impossible to be detected by a mere random generation. We conclude with a result on when a set of two random binary matrices is *not* primitive with high probability:

▶ **Proposition 20.** *For any fixed integer $m \geq 2$, if $2np - \log n \longrightarrow -\infty$ as $n \to \infty$, then $\mathcal{B}_m(p)$ is reducible with high probability. This implies that $\mathcal{B}_m(p)$ is not primitive with high probability.*

6 Conclusion

In this paper we have proposed a randomized construction for generating slowly minimally synchronizing automata. Our strategy relies on a recent characterization of primitive sets (Theorem 6), together with a construction (Definition 1 and Theorem 3) allowing to build (slowly minimally) synchronizing automata from (slowly minimally) primitive sets. We have obtained four new families of automata with simple idempotents with reset threshold of order $\Omega(n^2/4)$. The *primitive sets approach* to synchronizing automata seems promising and we believe that out randomized construction could be further refined and tweaked in order to produce other interesting automata with large reset threshold, for example by changing the way a permutation matrix is extracted by a binary one. As mentioned at the end of Section 3.1, it would be also of interest to apply the minimal construction directly to automata by leveraging the recent result of Alpin and Alpina ([2], Theorem 3).

References

1 N. Alon and J. H. Spencer. *The Probabilistic Method.* Wiley Publishing, 4th edition, 2016.
2 Yu. Alpin and V. Alpina. Combinatorial properties of entire semigroups of nonnegative matrices. *Journal of Mathematical Sciences*, 207(5):674–685, 2015.

3 D. S. Ananichev, M. V. Volkov, and V. V. Gusev. Primitive digraphs with large exponents and slowly synchronizing automata. *Journal of Mathematical Sciences*, 192(3):263–278, 2013.

4 M.-P. Béal, M. V. Berlinkov, and D. Perrin. A quadratic upper bound on the size of a synchronizing word in one-cluster automata. In *International Journal of Foundations of Computer Science*, pages 277–288, 2011.

5 M.V. Berlinkov. *On the Probability of Being Synchronizable*, pages 73–84. Springer International Publishing, 2016.

6 V.D. Blondel, R.M. Jungers, and A. Olshevsky. On primitivity of sets of matrices. *Automatica*, 61:80–88, 2015.

7 Y.-B. Chen and D. J. Ierardi. The complexity of oblivious plans for orienting and distinguishing polygonal parts. *Algorithmica*, 14(5):367–397, 1995.

8 P.-Y. Chevalier, J.M. Hendrickx, and R.M. Jungers. Reachability of consensus and synchronizing automata. In *4th IEEE Conference on Decision and Control*, pages 4139–4144, 2015.

9 M. de Bondt, H. Don, and H. Zantema. Dfas and pfas with long shortest synchronizing word length. In *Developments in Language Theory*, pages 122–133, 2017.

10 D. Eppstein. Reset sequences for monotonic automata. *SIAM Journal on Computing*, 19(3):500–510, 1990.

11 P. Frankl. An extremal problem for two families of sets. *European Journal of Combinatorics*, 3:125–127, 1982.

12 B. Gerencsér, V. V. Gusev, and R. M. Jungers. Primitive sets of nonnegative matrices and synchronizing automata. *Siam Journal on Matrix Analysis and Applications*, 39(1):83–98, 2018.

13 F. Gonze, B. Gerencsér, and R.M. Jungers. Synchronization approached through the lenses of primitivity. In *35th Benelux Meeting on Systems and Control*, 2016.

14 F. Gonze, V.V. Gusev, B. Gerencsér, R.M. Jungers, and M.V. Volkov. *On the Interplay Between Babai and Černý's Conjectures*, pages 185–197. Springer International Publishing, 2017.

15 V. V. Gusev and E. V. Pribavkina. Reset thresholds of automata with two cycle lengths. In *Implementation and Application of Automata*, pages 200–210, 2014.

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

17 A. Kisielewicz and M. Szykuła. Synchronizing automata with extremal properties. In *Mathematical Foundations of Computer Science 2015*, pages 331–343, 2015.

18 A. Mateescu and A. Salomaa. Many-valued truth functions, Černý's conjecture and road coloring. In *EATCS Bull.*, page 134–150, 1999.

19 M. Michalina Dzyga, R. Ferens, V.V. Gusev, and M. Szykuła. Attainable values of reset thresholds. In *42nd International Symposium on Mathematical Foundations of Computer Science*, volume 83, pages 40:1–40:14, 2017.

20 C. Nicaud. Fast synchronization of random automata. In *Approximation, Randomization, and Combinatorial Optimization*, volume 60 of *Leibniz International Proceedings in Informatics*, pages 43:1–43:12, 2016.

21 J. Olschewski and M. Ummels. *The Complexity of Finding Reset Words in Finite Automata*, pages 568–579. Springer Berlin Heidelberg, 2010.

22 J.-E. Pin. On two combinatorial problems arising from automata theory. In *Proceedings of the International Colloquium on Graph Theory and Combinatorics*, volume 75, pages 535–548, 1983.

23 V.Yu. Protasov and R.M. Jungers. Lower and upper bounds for the largest lyapunov exponent of matrices. *Linear Algebra and its Applications*, 438:4448–4468, 2013.

24 V.Yu. Protasov and A.S. Voynov. Sets of nonnegative matrices without positive products. *Linear Algebra and its Applications*, 437:749–765, 2012.

25 I. K. Rystsov. Estimation of the length of reset words for automata with simple idempotents. *Cybernetics and Systems Analysis*, 36(3):339–344, 2000.

26 M. Szykuła. Improving the upper bound the length of the shortest reset words. In *STACS*, 2018.

27 M. Szykuła and V. Vorel. An extremal series of eulerian synchronizing automata. In *Developments in Language Theory*, pages 380–392, 2016.

28 J. Černý. Poznámka k homogénnym eksperimentom s konečnými automatami. *Matematicko-fysikalny Casopis SAV*, 14:208–216, 1964.

29 M. V. Volkov. Synchronizing automata preserving a chain of partial orders. In *Implementation and Application of Automata*, pages 27–37, 2007.

30 M.V. Volkov. *Synchronizing automata and the Černý conjecture*, volume 5196, pages 11–27. Springer, 2008.