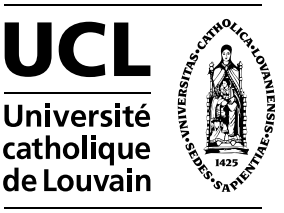


Université catholique de Louvain
Institute of Communication Technologies, Electronics and Applied Mathematics
Department of Applied Mathematics

Subset Properties and Synchronization of Finite State Automata

François Gonze

Thesis submitted in partial fulfillment of the requirements for the Ph.D. Degree in Engineering Sciences

Thesis Committee

Advisor -	Raphaël Jungers	*Université catholique de Louvain*
Jury -	Mikhail Volkov	*Ural Federal University, Russia*
	Balázs Gerencsér	*Alfréd Rényi Insitute of Mathematics, Hungary*
	Michel Rigo	*Université de Liège*
	Charles Pecheur	*Université catholique de Louvain*
Chairman -	Philippe Lefèvre	*Université catholique de Louvain*

Louvain-la-Neuve, 2019.

Préambule

Cher Lecteur,

Cet ouvrage présente, en anglais, les résultats des recherches que j'ai menées durant les cinq dernières années dans le cadre de ma thèse. Il vous plongera dans le monde fascinant des automates, et plus précisément dans les problèmes liés aux sous-ensembles d'états et à la synchronisation de ceux-ci. Le domaine dans lequel ces sujets s'inscrivent est l'informatique théorique. Néanmoins, les automates sont utilisés dans de nombreux autres domaines, dont le traitement du language et la modélisation de systèmes physiques. De plus, les méthodes utilisées pour arriver à nos résultats utilisent notamment l'optimisation convexe, la combinatoire et la théorie des graphes.

Le premier chapitre de cette thèse présente une introduction générale du sujet. Les quatre chapitres suivants reprennent les résultats de mes recherches, et ont en général été publiés sous forme d'articles dans des revues scientifiques. Le schéma à la page 24 présente une synthèse de leur contenu. Les quatre chapitres de résultats sont conçus de sorte à pouvoir être lus indépendamment.

Je vous souhaite une agréable lecture.

François Gonze,
le 21 juin 2019

Remerciements

Cette thèse de doctorat a commencé il y a cinq ans, et c'est maintenant une aventure qui s'achève. Durant ces années et celles qui ont précédé, j'ai eu la chance d'être entouré de personnes formidables, qui m'ont permis de mener mes différents projets à bien, et en particulier le présent ouvrage.

Je tiens d'abord à remercier les personnes qui m'ont, de par leurs conseils ou leur exemple, motivé et poussé à entreprendre cette aventure. En particulier Elodie, Pierre-Yves C., Adeline, Emilie, Adrien, Thomas G. et mes parents. Je remercie également mes anciens collègues de chez TW pour les bons moments, mais aussi pour l'expérience et la rigueur que j'y ai acquises, qui furent déterminantes pour mener à bien et structurer mes travaux de recherche.

Durant ces années à l'Euler, j'ai eu la chance de profiter d'un environnement de travail incomparable. Je remercie mes collègues Romain, Nicolas B., Nicolas D., Adeline, Emilie, Maguy et Pierre-Alain, Estelle, Arnaud, Matthew et Catherine, Corentin VdK., Corentin D. et Florence, Allan, David, Joachim, Vladimir, Andrea, Nikos, Pierre-Yves G. et Stéphanie G., Gabriel, Benjamin V., Benjamin C., Luc M., Luc R., Adrien S., Benoit L., Pierre-Alexandre, Guillaume Ber., Guillaume O., Loic, Cécile et Valentin, avec qui la vie à l'université est toujours un plaisir. J'ai également une pensée particulière pour mes anciens collègues de bureau: Balázs, Pierre-Yves, Adrien T. et Charles M.. Etant assistant, j'ai pu participer à l'encadrement de nombreux cours, ce qui fut une expérience particulièrement formatrice. Pour cela, je remercie les professeurs François Gl., Enrico V., Roland K., Vincent W., Olivier P., Kouider B.N., Pierre-Antoine A., Julien H., Olivier D., Paul V.D. et Benoit R.. J'ai de plus eu la chance de bénéficier d'un support administratif et technique d'une efficacité et d'une disponibilité remarquables. Je tiens à remercier Etienne, François W., Nathalie, Astrid, Marie-Christine et Pascale, qui permettent à tous de mener leur recherche dans les meilleures conditions.

En dehors du travail, j'ai pu compter sur le soutien de nombreux amis. Tout

d'abord mes amis d'études: Thomas G., Pierre-Yves C. et Anais, Gabriel et Maëlle, Adrien T., Pierre-Antoine A., Emilie, Adeline et Yannick, et Olivier A.. Mes amis de Louvain: Aubry et Virginie, ma filleule Anaëlle, Sophie, Anouk et Eli, Isa, Nash et Aimeric, Céline, Oli L. et Audrey, Guillaume Bec. et Maureen. Mes anciens cokotteurs Oli V., Max, Thomas F. et Amandine, Robin, Sophie, Sandrine et Anais. Mes anciens colocataires Alex, Julien, Sylvie, Guillaume, Nuwan et Fiona. Les membres du club de Go: Bernard, Yves, Gabriel, Domitien, Thomas S. et Thibault P.. Les membres des différentes chorales: Clément, Charlotte, Julien, Thomas, Gabriel, Pierre-Yves C., Fanny L., Pauline D., Catherine, Corentin D., Myriam, Caroline, Pierre-Yves G. et Stéphanie G.. Les anciens: Benoit K., Michaël et Céline. Et enfin Numa et Amandine, Nicolas et Marie-Sophie, Stéphanie D., Charlotte et Dimitri, Hélène et Fabien, Lara, Céline M. et Ludo, Philippe et Claire, Sabine et Jean-Marie, Patrick et Pascale, ainsi que Docteur Benoit.

La rédaction en anglais demandant un état d'esprit complètement différent de la rédaction française, j'ai souvent eu recours à des avis extérieurs quant à mes textes. Ainsi, je tiens à remercier les personnes suivantes pour leur aide et leurs conseils lors de la rédaction de mes articles et du présent ouvrage: Xavier, Jean, Elodie, Costanza, Vladimir, Myriam, Charles, Matthew, Adrien T. et Balázs.

Le travail de recherche s'insère dans un domaine pré-existant, dont la découverte est une composante importante. L'accompagnement académique est donc crucial pour mener à bien un tel projet. Je tiens à remercier les membres de mon comité d'accompagnement, les professeurs Charles P. et Michel R., les membres extérieurs de mon jury, les professeurs Balázs G., Philippe L. et Mikhail V., pour leur conseils et leur soutien, ainsi que mon promoteur, le professeur Raphaël J., pour son support et sa motivation sans faille tout au long de cette thèse.

Enfin, je remercie ma famille: mes parents Brigitte et Xavier, qui en plus de m'avoir toujours soutenu, m'ont donné le goût des mathématiques et la notion du travail bien fait. Je remercie Mamy, Bonne Mamy et Bon Papa, mes beaux-parents Anne et Jean, Mamou, Marraine Sylvie et Jean, Jean-Louis, mes (beaux-)frères et (belles-)soeurs Myriam, Bert, Philippe, Caroline, Damien, Jonathan, Aurore et Laurent, mes oncles et tantes, cousins et cousines, neveux et nièces. J'ai finalement une pensée pour ma fille Louise, sans qui la rédaction de cette thèse aurait été bien trop facile, ma merveilleuse épouse Elodie qui a su me soutenir toutes ces années, ainsi que notre (nos?) enfant(s?) à venir.

Abstract

Finite state systems, also called automata, are key tools for modelling physical systems, for emulating computers and for text processing. These systems are composed of a set of possible states and a set of letters. The letters represent the commands which can be applied to the system, and their effect depends on the current state. The control of such systems raises many theoretical questions about reachability and synchronization of subsets of states, one of them being well known as the Černý conjecture. This conjecture, which has been open for 55 years, states that a synchronizing automaton has a quadratic reset threshold.

In this thesis, we study open problems related to subsets of states while targeting the Černý conjecture. Our results are twofold. On the one hand we disprove several conjectures that, if true, would have led to proving the Černý conjecture. More precisely, we first disprove a conjecture on the synchronizing probability function. Second, we provide a counterexample to a tentative improvement of the best general bound on the reset threshold, based on subset avoidance. Finally, we disprove a conjecture on subset reachability.

On the other hand, we prove positive results toward the Černý conjecture. We first prove an upper bound on the triple rendezvous time, the length of the shortest word synchronizing three states of an automaton. Secondly, we prove upper and lower bounds for 2-Transitivity of permutation sets. Thirdly, we provide an algorithmic characterization of completely reachable automata. Finally we prove that automata generating the whole transformation semigroup, which is the widest possible transformation set, have quadratic reset threshold.

Contents

Chapter 1

Introduction and Background

Finite state systems, or *automata*, are classical tools in many fields of science. They allow to parse texts and efficiently find letter sequences [BR10, Kar13]. In language theory, they are the basic tools defining formal languages and context-free languages [Rig14a, Rig14b]. In theoretical computer science, automata provide simple models for the behaviour of computing devices. More recently, finite state systems have been at the core of synthesis, model checking and verification of complex automated systems [PCC02, Jav02, CP09, BPQR14].

One key property that these systems can have is the *synchronizing property*. This property states that there exists a control sequence driving the system to a fixed final state, whichever the starting state was. In other words, it allows to regain control over the system. The term "synchronizing automata" was first used in [Liu63]. This property was formalized in the 60s and studied in parallel in [Liu63, Lae63, Černý64]. Synchronizing automata appear in various branches of mathematics and are related to synchronizing codes [BPR09], part orienting problems [Nat86, Nat89], substitution systems [FS07], primitive sets of matrices [GGJ16a], synchronizing groups [ACS17], convex optimization [GJ16], and consensus theory [CHJ15]. Synchronization has recently gained a lot of attention in theoretical computer science as major developments have been achieved on two problems concerning it. The first of these problems is the *Černý conjecture* on *synchronizing automata*, which has been open since 1964 [Černý64] and the second one is the *road coloring conjecture*, which was open since 1970 [Adl70]. The road coloring conjecture was proved by Trahtman in 2009 [Tra09].

This event created a lot of interest in the field and new hopes for solving the Černý conjecture. Since 2000, much progress has been made and new research directions have been proposed successfully. In particular, the Černý conjecture has been proven for particular classes of automata, and the best general upper bound on the reset threshold of automata has been improved in 2018 [Szy18].

In this thesis, we focus on problems related to synchronization and leading to the Černý conjecture. More precisely, the chosen approaches focus on subsets of states of automata. Our results include solutions to open problems on properties of subsets, results on the reset threshold for particular families of automata and refutations of open conjectures.

The remaining of this chapter provides the background of this thesis. In Section 1.1, we give a formal introduction to finite state systems and their graph representation. Section 1.2 explains automata synchronization: we present the Černý conjecture and the two most classical methods used to approach it, the *compression method* and the *extension method*. In Section 1.3, we present practical computational aspects of synchronizing automata, i.e. the representation as matrices and the algorithmic aspects. In Section 1.4, we present the open problems and conjectures that have driven our research, as well as the most recent results on them. Finally, we present the outline of the thesis in Section 1.5.

1.1 Finite State Systems

Finite state systems, or automata, are a natural way to model systems that can be in multiple different states. In this work, we consider automata for which the effect of letters is a function of the state, namely *deterministic, finite state, complete* automata.

Formally, a *deterministic, finite state, complete automaton* (DFA) is a triple (Q, Σ, δ) with Q the set of *states*, with Σ an alphabet of *letters* and with δ a transition function $\delta : Q \times \Sigma \to Q$ defining the effect of the letters on the states. It is convenient and often much easier to represent a DFA with a directed graph. In this case, each state is represented by a node and each transition $\delta(q_i, l) = q_j$ is represented by a directed edge from q_i to q_j labelled by letter l. We say that an automaton is strongly connected if its graph representation is strongly connected. In the following, we often directly use a graph to describe an automaton since we can recover the formal description (Q, Σ, δ) of the automaton from the

graph representation[1].

Example 1.1.1. *Figure 1.1 shows an example of an automaton in its graph representation. This example is used repeatedly through this thesis, and we refer to it as $\mathscr{E}$. In this case, the set of states is $Q = \{q_1, q_2, q_3, q_4\}$, the letters are $\Sigma = \{a, b\}$, and the effect of these letters is $\delta(q_1, a) = q_2$, $\delta(q_1, b) = q_1$, $\delta(q_2, a) = q_1$, $\delta(q_2, b) = q_3$, $\delta(q_3, a) = q_3$, $\delta(q_3, b) = q_4$, $\delta(q_4, a) = q_1$ and $\delta(q_4, b) = q_2$.*

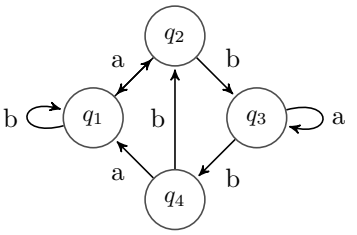


Figure 1.1: Graph representation of the automaton $\mathscr{E}$.

In this context, we define a *word* as a finite sequence of letters of Σ, and the set of all the finite words on the alphabet Σ is denoted by Σ^*. Transition functions are recursively extended to words as follows: given a word $w' = lw$ in Σ^*, with l a letter and w a word, we set $\delta(q_i, lw) = \delta(\delta(q_i, l), w)$. We call *prefix* (resp. *suffix*) of a word w the words w' with $|w'| < |w|$ that are composed of the first letters of w (resp. the last letters of w), in the same order. In graph terms, applying a word to a state corresponds to following the path labelled by this word in the graph. Similarly, transition functions are extended to sets of states as follows: for a set $S \subset Q$ and a word $w \in \Sigma^*$, $\delta(S, w) = \{q_j | q_j = \delta(q_i, w), q_i \in S\}$. In order to simplify the notation, we write $q_i w$ for $\delta(q_i, w)$ and Sw for $\delta(S, w)$. We call *transition subsets* of Sw the subsets Sw', with w' being a prefix of w.

Example 1.1.2. *Let us illustrate the concept of words applied on a state or on a set of states. If the automaton $\mathscr{E}$ begins in state q_1, applying the word ab leads to state q_3, as shown in Fig.1.2. For subsets, if we apply the word ab to the set $\{q_1, q_3\}$, we obtain the set $\{q_3, q_4\}$, as shown in Fig.1.3.*

[1]In this section, the states of automata are coloured in white, gray or darker gray to illustrate the effect of letters and word on states, and the synchronizing processes. Later in this thesis, all states are coloured in gray for aesthetic reasons.

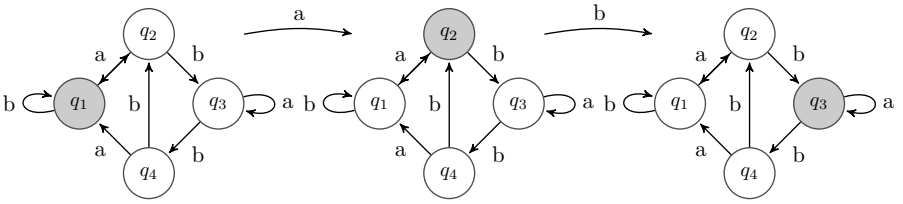


Figure 1.2: The word ab applied to $\mathscr{E}$ starting in state q_1.

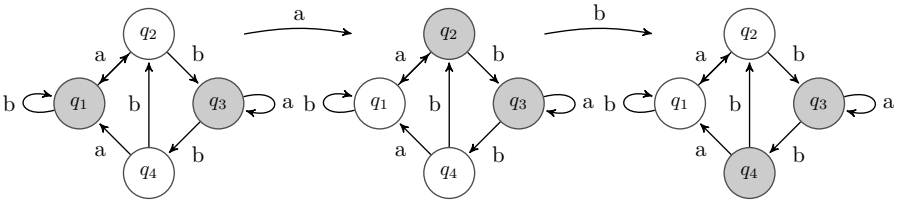


Figure 1.3: The word ab applied to the set $\{q_1, q_3\}$ of $\mathscr{E}$.

The *image* of a state q by a word w is the state qw. We notice that, in a DFA, every state has exactly one image. By extension, the image of a set S by a word w is the set Sw. We notice that the cardinality of the image of a set S is always smaller or equal to the cardinality of S since every state of S has exactly one image and since two states can have the same image. Similarly, we call the *image* of a word w the set Qw. The *rank* of a word w is the cardinality of its image $|Qw|$. Conversely, for the ease of notation, we call *deficiency* of a word w the value $n - |Qw|$, and call this word $(n - |Qw|)$-*deficient*. We call rank of an automaton the rank of its lowest rank word.

Example 1.1.3. *To illustrate these concepts, consider the word $w = ab$ of the automaton in Fig.1.1. The image of q_1 by w is q_3, the image of q_3 by w is q_4 and the image of both q_2 and q_4 by w is q_1. So the image of Q by w is $\{q_1, q_3, q_4\}$. Therefore $w = ab$ is of rank 3, or 1-deficient.*

1.2 Synchronization

In a finite state system, it can be desirable to have a particular input sequence which would ensure a known final state, independently of the initial one. Indeed, applying this sequence would allow to recover control over the

device. Automata with this property are called *synchronizing*. Synchronizing automata first appeared in computers and relay control systems in the 60s [Moo56, Gin58]. In the 80s and 90s, this subject found applications in robotics and in the industry [Nat86, Nat89]. More recently, it has been used to model consensus theory [CHJ15] and theoretical results obtained for synchronization have been applied successfully to primitivity of matrix sets (see [BJO15, GGJ16a]).

Formally, an automaton $\mathscr{A} = (Q; \Sigma; \delta)$ is synchronizing if there exists a word $w \in \Sigma^*$ such that Qw is a single state, or in other words such that w has rank 1. Then, we call w a *synchronizing word*. We also say that a word w synchronizes a subset $S \subset Q$ if Sw has rank 1.

One way of representing synchronization is to look at the *power automaton* of an automaton $\mathscr{A} = (Q; \Sigma; \delta)$. The power automaton $P(\mathscr{A})$ is defined as the automaton obtained by setting all the possible non empty subsets of Q as states, by maintaining the same letters as $\mathscr{A}$ and with the transition function equal to the transition function of $\mathscr{A}$ on its sets of states, so $P(\mathscr{A}) = (2^Q; \Sigma; \delta_{2^Q})$. Therefore the power automaton illustrates the effect of letters on any set of states of the automaton. Synchronization of the automaton $\mathscr{A}$ is therefore equivalent to the existence of a path from Q to a singleton in $P(\mathscr{A})$.

Example 1.2.1. *Figure 1.4 represents $P(\mathscr{E})$, the power automaton of $\mathscr{E}$. We notice that the word $w = abbababba$ connects Q to q_1, which means that $Qw = q_1$, therefore it is a synchronizing word.*

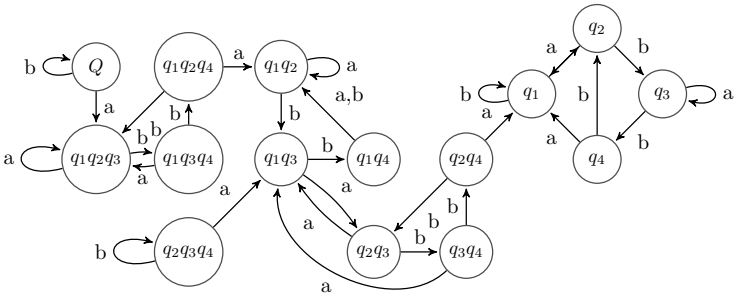


Figure 1.4: The power automaton $P(\mathscr{E})$.

1.2.1 The Černý Conjecture

We now come to the main open problem in this field, the Černý Conjecture. It tries to answer the following question: if an automaton is synchronizing, does it have a short synchronizing word?

Jan Černý conjectured that any synchronizing automaton has a synchronizing word of quadratic length, more precisely:

Conjecture 1.2.2. *[The Černý Conjecture [Černý64, ČernýPR71]]*
Let $\mathscr{A} = (Q; \Sigma; \delta)$ be a n-states synchronizing automaton. Then there is a synchronizing word $w \in \Sigma^*$ of length at most $(n-1)^2$.

Although this conjecture has a simple statement, it is still unproven. If the conjecture is true, then $(n-1)^2$ is also a tight bound. Indeed, in [Černý64], Černý proposes a family of automata attaining it for any number of states. We refer to this family as the *Černý family* of automata $\mathscr{C}_n$, represented in Fig. 1.5. In the following, we call the length of the shortest synchronizing word of an automaton $\mathscr{A}$ its *reset threshold*, and we write it $rt(\mathscr{A})$. Synchronizing automata attaining the bound of Conjecture 1.2.2 or having a reset threshold close to it are very infrequent (see [AGV10, Kar01, Rom08, AVZ07] for examples). On a brighter side, another longstanding open problem based on synchronizing automata, the road-coloring problem, was recently solved by Trahtman (see [Tra09]). Nevertheless, many problems mixing road-coloring and the Černý conjecture are still open (see [VR15, Car14]).

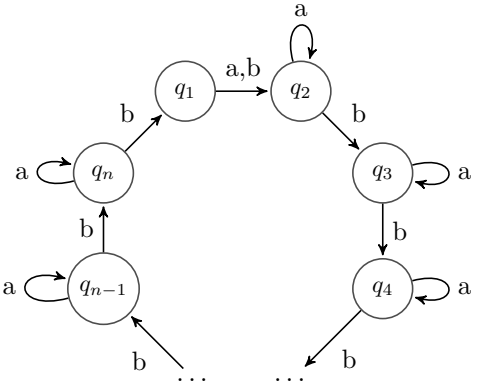


Figure 1.5: The automaton of Černý family with n states $\mathscr{C}_n$.

In the last decades, Conjecture 1.2.2 has been the subject of intense re-

search. It has been proven to hold for several families of automata (see [Kar03, Epp90, Dub98, GK13, Vol09, AV04, Tra07, AV05, BBP11, CD13, Černý64]), including cyclic and Eulerian. However, the best general upper bound on the reset threshold of an automaton with n states is close to $(n^3 - n)/6$. This bound was obtained by Pin and Frankl [Fra82, Pin83], and rediscovered independently in [KRS87]. Although it had been holding for more than 30 years, it was improved by a factor 10^{-5} in 2017 [Szy18]. An overview on the state of the art is given by Volkov [Vol08].

1.2.2 Constructive Approaches

Two constructive methods are often used in order to bound the reset threshold or to prove Conjecture 1.2.2 for particular automata classes: the *compression method* and the *extension method*. They both consist in building a short synchronizing word by a recursive process. The compression method starts with the whole set of states Q, and search recursively for words reducing the number of states in the current set. The extension method does the opposite: it starts with a single state and searches recursively for words sending a larger set of states on the current set, ending with Q.

We summarize here the two methods and show how the problems we tackle are related to them.

The Compression Method

In the compression method, the set of states is recursively compressed, starting with the set of all states. More precisely, one searches for words $w_1, \ldots, w_k$ and sets of states $S_1, \ldots, S_k$ with $k < n$ such that $Qw_1 = S_1$, $S_i w_{i+1} = S_{i+1}$ for $1 \leq i < k$, with $|S_{i+1}| < |S_i|$, and $|S_k| = 1$. The concatenation of the words w_i is a synchronizing word mapping Q to S_k.

Example 1.2.3. *Let us apply this method to $\mathscr{E}$ and illustrate it in Fig. 1.6. Starting with Q, the word $w_1 = a$ allows to obtain the set $S_1 = \{q_1, q_2, q_3\}$. Then, the word $w_2 = bba$ allows to obtain the set $S_2 = \{q_1, q_2\}$, and finally $w_3 = babba$ allows to obtain the set $S_3 = q_1$. Therefore, this method leads to the synchronizing word abbababba.*

In order to bound the length of this synchronizing word, one bounds the length of each of the subwords w_i. This is the method used by Pin and Frankl [Pin83, Fra82] to obtain a general upper bound on the reset threshold of any synchronizing automaton. They showed that in a synchronizing automaton,

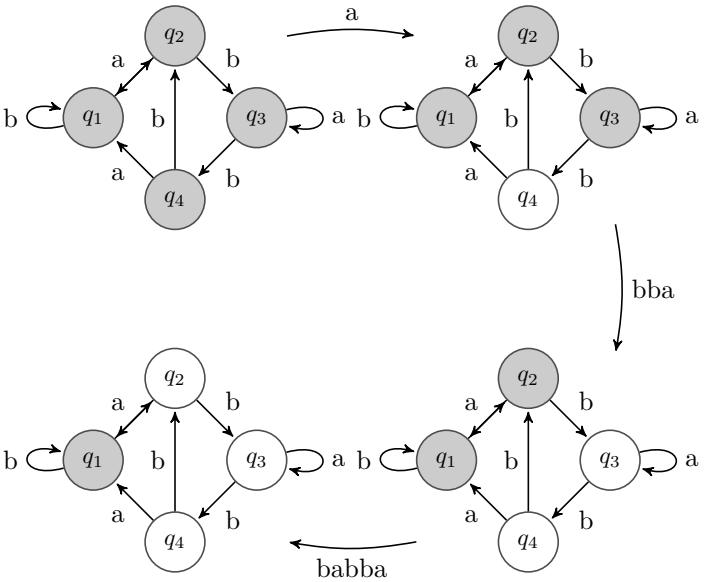


Figure 1.6: The compression method applied to the automaton $\mathscr{E}$.

any set of size k can be compressed by a word of length at most $\frac{(n-k)(n-k-1)}{2}$. Summing this value from 2 to n provides the bound $\frac{n^3-n}{6}$. Slight improvements to this bound, still using this method, were obtained recently, as we will see in section 1.4.

Compressing a set of states implies that two of its states are synchronized. Therefore, the shortest word compressing a set is also the shortest word synchronizing a pair of states of the set. This is the argument used in the proof of Pin and Frankl where they showed that, in a set with k states, one of the pairs could always be synchronized with a word of length $\frac{(n-k)(n-k-1)}{2}$. One way of representing the synchronization of pairs is to use the graph of pairs of the automaton, that we call *square graph*. We define the square graph of $\mathscr{A}$ as the automaton with pairs of $\mathscr{A}$ plus single states toward which pairs are synchronized as states[2], the same letters as $\mathscr{A}$, and the transition functions defined for the pairs. We write $SG(\mathscr{A}) = (Q^2 \cup Q', \Sigma, \delta_{Q^2 \cup Q'})$ with Q' the single states kept, and $\delta_{Q^2 \cup Q'}$ the transition function defined on pairs of states. Figure 1.7 represents the square graph of $\mathscr{E}$.

[2]Later in this thesis, we sometimes include the whole set of single states in the square graph for a better understanding.

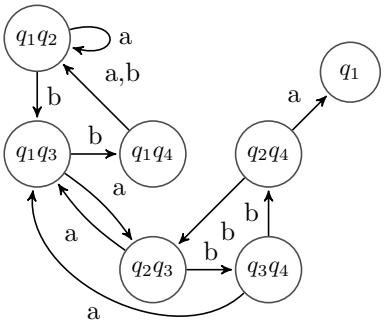


Figure 1.7: The square graph $SG(\mathscr{E})$.

In terms of square graph, finding the shortest word compressing a set is equivalent to finding the shortest path in the square graph from a pair of the set to a single state.

Example 1.2.4. *Let us illustrate the compression process of Example 1.2.3 in terms of square graph. Figure 1.8 shows the steps of the process. Each graph represents in light gray the pairs which are in the set and in dark gray the pairs which have a shortest path towards a single state. Initially, all the states are in the set, therefore we must consider all the pairs, and the one with the shortest path towards q_1 is q_2q_4. This provides the first compressing word $w_1 = a$, which leads to the set $S_1 = \{q_1, q_2, q_3\}$. The pair in S_1 with the shortest path towards q_1 is q_2q_3. This provides the second compressing word $w_2 = bba$, which leads to the set $S_2 = \{q_1, q_2\}$. Finally, S_2 has only a single pair, and the shortest path from this par to q_1 is labelled by $w_3 = babba$.*

Recall that the *diameter* of a graph is the maximal length of the shortest path between two of its nodes. We notice that any pair can be synchronized with a word shorter or equal to the diameter of the square graph. For a synchronizing automaton with n states, it implies that there exists a synchronizing word of length $n - 1$ times this diameter. Therefore, understanding the structure of the square graph could lead to improvements to the Pin-Frankl bound. This motivated the study done in chapter 3, where we analyze the structure of the square graph based on the effect of the letters on single states. The proof of Pin also motivated the excluding state question, which was first introduced by Trahtman in a tentative improvement of the Pin-Frankl bound [Tra11]. This question is studied in section 4.3.

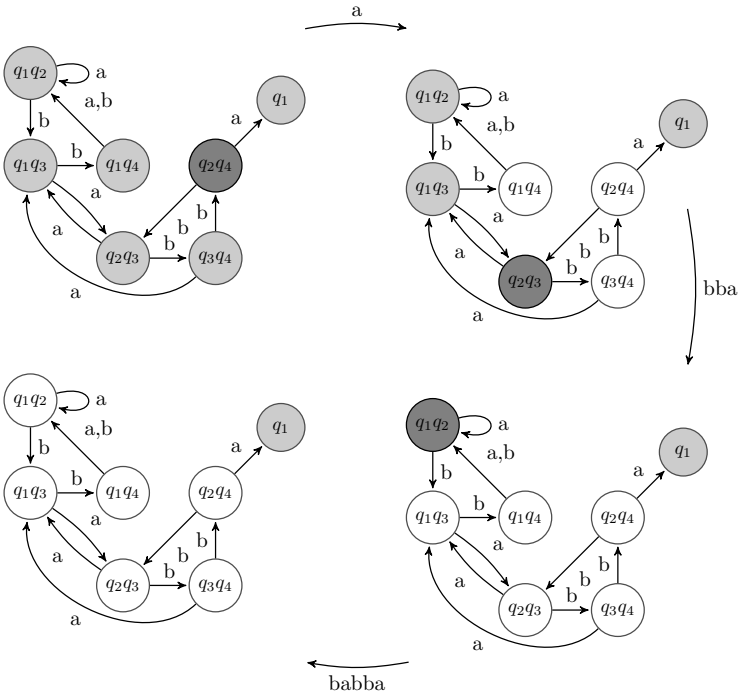


Figure 1.8: Synchronization of $\mathscr{E}$ using the compression method based on $SG(\mathscr{E})$.

The Extension Method

In the extension method, the set of states is recursively extended, starting with a single state. In order to formalize this method, we need the notion of *preimage* of a set S by a word w. It is defined as the set of states which are sent on S by w, and we write it Sw^{-1} i.e.

$$Sw^{-1} = \{q \in Q | qw \in S\}.$$

For preimage, we call transition sets the sets Sw'^{-1}, with w' any suffix of w. In order to build a synchronizing word, starting from a single state q, one will search for words $w_1, \ldots, w_k$ and sets of states $S_1, \ldots, S_k$ with $k < n$ such that $qw_1^{-1} = S_1$, $S_i w_i^{-1} = S_{i+1}$ for $1 \leq i < k$, with $|S_{i+1}| > |S_i|$, and $S_k = Q$. The concatenation of the words w_i is a synchronizing word mapping Q to q.

Example 1.2.5. *Let us apply this method to $\mathscr{E}$ and illustrate it in Fig. 1.9.*

Starting with q_1, the word $w_1 = a$ provides the set $S_1 = \{q_2, q_4\}$. Then, the word $w_2 = ababb$ provides the set $S_2 = \{q_1, q_2, q_4\}$, and finally $w_3 = abb$ provides the set $S_3 = Q$. Therefore, this method leads to the synchronizing word abbababba.

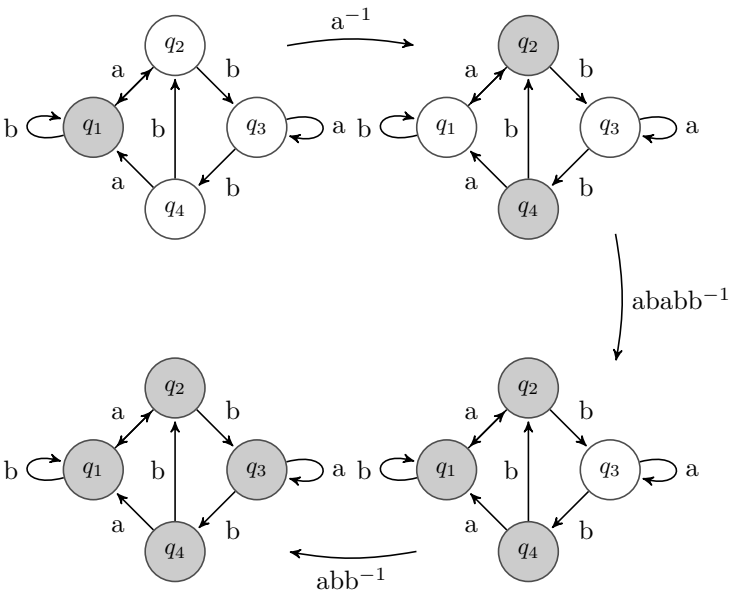


Figure 1.9: The extension method applied to $\mathscr{E}$.

In order to bound the length of this synchronizing word, one bounds the length of each subword w_i. We notice that in any synchronizing automaton, there exists a single state which can be extended by a single letter. Then, if it is possible to prove that any other subset can be extended by a word of length n, it provides a synchronizing word of length $(n-2) \times n + 1 = (n-1)^2$, which is the bound of Conjecture 1.2.2.

An important difference between the contraction method and the extension method is the asymmetry between image and preimage of sets. Indeed, the image of any set by any letter has a cardinality lower than or equal to the cardinality of the set. This implies that, in the compression method, it is always possible to contract a set with the transition subsets being of the same size or lower. Oppositely, the cardinality of the preimage of a set can be of higher cadinality, but also of lower cardinality. Indeed, some states might have no preimage. Therefore, in the extension method, it is not always possible to extend a set with transition sets being of the same size or larger than the

initial set. For example, in $\mathscr{E}$, state q_4 has no preimage by letter a, so the set $\{q_2, q_3, q_4\}$ cannot be extended without transiting through sets of lower cardinality, as shown in the power graph in Fig. 1.4.

In the compression method, showing that a pair in a subset could be synchronized by a word w was enough to ensure that w was a compressing word for this subset. However, a state s having two preimages by word w is not enough to guarantee that w is extending any set containing s. Indeed, in order to guarantee this, all the other states of S must also have at least one preimage, which could not be the case. The compression method allowed to obtain the bound $\frac{n^3-n}{6}$ for the reset threshold of any synchronizing automaton. Oppositely, proofs using the extension method can hardly be applied to all synchronizing automata, but allowed to prove quadratic bounds for many particular classes of automata. Some classes for which this method allowed to prove Conjecture 1.2.2 are circular automata [Dub98], solvable automata [Rys97], Eulerian automata [Kar03], and quasi Eulerian automata [Ste11]. Moreover, quadratic bounds close to the conjecture were obtained for one-cluster automata [BBP11] and simple idempotent automata [Rys00]. Most of these results are obtained by proving, for any set of states, the existence of a short word extending it. This general framework is explained in detail by Steinberg in [Ste11]. A tool allowing to easily use the extension method is the Γ-graph of automata. In chapter 5, we study this graph and prove a quadratic bound for automata generating the full monoid of transformations T_n.

1.3 Matrix Representation and Algorithmic Aspects

In many applications, algorithms providing a short synchronizing word are needed. On the theoretical side, if Conjecture 1.2.2 was false, a promising track would be to build a counterexample algorithmically. This naturally raises algorithmic questions about synchronization of automata: is the synchronization problem decidable? Is it possible to find a short synchronizing word or even the shortest synchronizing word? Is it possible to determine the reset threshold or to approximate it? Moreover, is it possible to answer these questions in a short computational time?

In the following, in order to analyze the algorithms using the algorithmic complexity, we use the classical asymptotic notation $o(f(x))$ and $O(f(x))$.

They are defined as follows: we say $f(x) = o(g(x))$ if we have that

$$\lim_{x\to\infty} \frac{f(x)}{g(x)} = 0,$$

and $f(x) = O(g(x))$ if and only if there exists a positive real number M and a real number x_0 such that

$$|f(x)| \leq Mg(x) \text{ for all } x \geq x_0.$$

We say that an algorithm on automata runs in quadratic time if its time complexity is $O(n^2)$, with n being the number of states of the automaton. We notice that for many algorithms, the complexity also depends on the number of letters $|\Sigma|$.

Verifying that an automaton is synchronizing is equivalent to verifying that there is a path in the power automaton from Q to a singleton. Therefore, a naive approach is to use a breadth-first search algorithm in the power automaton. Since the size of the power automaton is exponential in the number of states, this method can be of exponential time. However, there is a much easier way to verify that an automaton is synchronizing. Indeed, we know that, if every pair can be synchronized, then it is possible to recursively synchronize pairs of the set and to obtain a synchronizing word, as in the method of Pin. Verifying that any pair can be synchronized is equivalent to verifying that any pair is connected to a singleton in the square graph, and this can be done with computational time of order $O(n^2|\Sigma|)$ [Vol08].

Now, knowing that an automaton is synchronizing, the next step is to compute a synchronizing word. Eppstein proposed an algorithm running in time $O(n^2|\Sigma|)$ and using a working space of size $O(n^2 + n|\Sigma|)$ ([Epp90], Theorem 6). Once it has been proven that an automaton is synchronizing, it is possible to build a synchronizing word of length $O(n^3)$.

We saw that proving that an automaton is synchronizing and finding a synchronizing word can both be done in polynomial time. However, determining the reset threshold is much harder. It was shown in [Epp90] that finding the reset threshold was NP-complete. Moreover, approximating it is also hard [GS15]:

Theorem 1.3.1 (Theorem 5.1 in [GS15])**.** *For every constant $\epsilon > 0$,*

$$SYNAPPX(\{0, 1, 2\}, n^{1-\epsilon})$$

is not solvable in polynomial time, unless $P = NP$.

In this statement, $SYNAPPX(\{0,1,2\}, n^{1-\epsilon})$ is defined as the problem of finding a synchronizing word of length at most $n^{1-\epsilon}$ times the reset threshold of an automaton with alphabet $\{0,1,2\}$.

The article [OU10] provides a detailed overview these questions. These results imply that when the size of automata grows, it is very hard to test if they have a short synchronizing word or not. The computational complexity of problems about reaching subsets of states or synchronizing subsets of states are studied in [BFS18]. It was proven that most of these problems were computationally hard. The packages TESTAS by Trahtman [Tra02] and COMPAS by Chmiel and Roman [CR10] allow to compute the reset threshold of small automata.

Conjecture 1.2.2 has been algorithmically verified for any automaton up to $n = 4$ with unrestricted alphabet and for $n = 10$ with two letters by Trahtman [Tra06]. In 2013, this latter case was extended to $n = 11$ by Kisielewicz and Szykuła [KS13]. In 2017, de Bondt et al. verified that the conjecture was true for automata up to 6 states with alphabet of any size [DBDZ17].

In order to work numerically with synchronizing automata, it is necessary to represent them in a way that is easily used in algorithms. To do so, it is convenient to represent them as sets of matrices, where each letter is represented by its adjacency matrix. In that setting, they are defined as follows.

Definition 1.3.2. *A deterministic, finite state, complete* automaton *(DFA) is a set of m row-stochastic* [3] *matrices $\Sigma \subset \{0,1\}^{n\times n}$.*

In this definition, m and n are respectively the number of *letters* in the alphabet and the number of *states* of the automaton. Each letter corresponds to a matrix $L \in \Sigma$ with binary entries, which satisfies $Le^T = e^T$, where e is the $1 \times n$ all-ones vector. We notice that, in this setting, the matrix corresponding to a word is equal to the product of the matrices corresponding to the letters of the word. We write Σ^t for the set of products of length t of matrices taken in Σ, and Σ^* for the set of all possible products of matrices. We refer to these products as *words*. States and sets of states are represented by their $1 \times n$ characteristic vector in the canonical way. With this setting, the definition of synchronization is the following.

Definition 1.3.3. *An automaton $\Sigma \subset \{0,1\}^{n\times n}$ is* synchronizing *if there is*

[3] Recall that stochastic means that the values are all non-negative and the sum of the elements in each row is equal to one.

an index $1 \leq i \leq n$ *and a finite product* $W = L_{c_1} \cdots L_{c_s} : L_{c_j} \in \Sigma$ *which satisfy*

$$W = e^T e_i,$$

where e_i *is the* i^{th} *standard* $1 \times n$ *basis vector.*
In this case, the sequence of letters $L_{c_1} \cdots L_{c_s}$ *is said to be a* synchronizing word.

Example 1.3.4. *The two letters of the example automaton are the following matrices:*

$$a = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix} b = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \end{pmatrix}$$

and we have the synchronizing word:

$$abbababba = \begin{pmatrix} 1 & 1 & 1 & 1 \end{pmatrix}^T \begin{pmatrix} 1 & 0 & 0 & 0 \end{pmatrix}.$$

The Černý conjecture expressed in matricial terms is the following:

Conjecture 1.3.5 (The Černý conjecture, 1964 [Černý64, ČernýPR71])**.** *Let* $\Sigma \subset \{0,1\}^{n \times n}$ *be a synchronizing automaton. Then, there is a synchronizing word of length at most* $(n-1)^2$.

In order to study the synchronization process, a game theoretic approach leading to a linear program has been proposed by Jungers in [Jun12]. The results of this program provides the *synchronizing probability function* (SPF) of an automaton. The SPF is a tool quantifying the synchronizing efficiency that can be reached for a defined word length. More precisely, if the argument of the function is higher than the automaton reset threshold, the function provides the value 1, while for arguments lower than the reset threshold, the function is non-decreasing. In Chapter 2, we study the synchronizing probability function and disprove a conjecture about it ([Jun12], Conjecture 2). A natural extension field from this matrix approach is the primitivity of sets of matrices. Indeed, [BJO15] obtained that primitive sets of matrices with long shortest positive product could lead to a counterexample to Conjecture 1.2.2. In this thesis, we obtained preliminary results on generation of primitive sets with long shortest positive product [GGJ16b]. This track was then successfully explored in [CJ18a, GGJ16a]. Noticeably, the synchronizing probability function has recently been extended to primitive sets of matrices in [CJ18a, CJ18b, Cat19].

1.4 Open Problems and Recent Results

In order to understand synchronization of automata, some more general conjectures on synchronizing automata have been made. A recurrent idea in these conjectures is to provide milestones in the synchronizing process. Trying to prove or disprove these milestones led to promising results, as we will see. Some of these conjectures, such as the Pin-Černý conjecture, are directly linked with the compression method. Others, such as the Don conjecture, are linked to the extension method. Finally, some are based on other concepts, which can involve subsets or other types of approaches. In this section, we describe these problems as well as the most recent results related to Conjecture 1.2.2.

The Pin-Černý Conjecture A first natural research track is to try to show that the compression method is a quadratic process. In his thesis [Pin78], Pin proved that the first three steps of this method could always be achieved by short words:

Theorem 1.4.1 ([Pin78])**.** *Let $\mathscr{A}$ be an automaton with n states and let $0 \leq k \leq 3$. If there exists a word of rank $\leq n-k$ in $\mathscr{A}$, there exists such a word of length $\leq k^2$.*

Extending this result to the whole compression process would then have led directly to Conjecture 1.2.2, and Pin conjectured the following:

Conjecture 1.4.2 ([Pin83, Pin81], Conjecture (C))**.** *Let $\mathscr{A}$ be an automaton with n states and let $0 \leq k \leq n-1$. If there exists a word of rank $\leq n-k$ in $\mathscr{A}$, there exists such a word of length $\leq k^2$.*

However, although this conjecture was very promizing and held for 20 years, a counterexample was provided by Kari in 2001 [Kar01]. The automaton presented is represented in Fig. 1.10. Indeed, it turns out that this automaton with 6 states cannot be compressed to a set of two states with a word shorter than 17 letters, while Conjecture 1.4.2 implied that it was possible with a word of 16 letters.

This automaton is so far the only known counterexample to Conjecture 1.4.2. Moreover, it has the particularity that it does not satisfy it for a value $n-k$ that is not its minimal rank. Indeed, the automaton is in fact of rank 1 and has a synchronizing word of length 25. An alternative version of the conjecture, with the condition that the rank has to be minimal, is also attributed to Pin by Rystsov ([AS09, Rys92]):

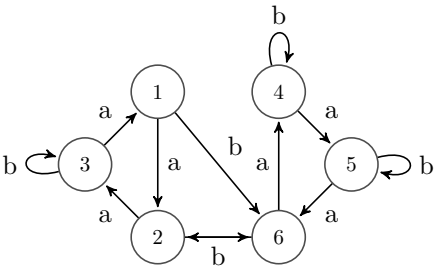


Figure 1.10: Kari's automaton.

Conjecture 1.4.3 ([Rys92], Conjecture 1)**.** *Let $\mathscr{A}$ be an automaton with n states and rank r, then there exists a word of rank r of length $\leq (n-r)^2$.*

This conjecture is open and it would also imply that Conjecture 1.2.2 is true.

Synchronization of subsets with short words After compressing Q to a subset S with $|S| = m < n$, we are interested in finding the shortest word synchronizing S. Therefore, a natural second question is: what is the length of the shortest word synchronizing a set of m states? A proof strategy for Conjecture 1.2.2 behind this question is that, if we can prove that a subset of two states can be synchronized with a word of some length, then extend this result to a set of three states, and continue recursively for sets of higher cardinality, we will finally have the set Q. However, on the one hand, for two states alone, the worst possible case is reached by the automata of the Černý family, where two of its states need a word of length $n(n-1)/2$ to be synchronized. On the other hand, Conjecture 1.2.2 states a bound of $(n-1)^2$ on the reset threshold. The ratio $1/2$ between synchronizing the worst pair and synchronizing the full automaton is quite bad, and therefore this method is not very promising. However, the length of the shortest word synchronizing a subset of an automaton has been conjectured by Cardoso to be lower that the following threshold:

Conjecture 1.4.4 ([Car14], Conjecture 5.1.1)**.** *Given a synchronizing automaton $\mathscr{A} = (Q; \Sigma; \delta)$ with n states and a subset S of Q with m elements, there is a synchronizing word $w \in \Sigma^*$ for S, whose length is at most*

$$f(n,m) = (n-1)^2 - \left\lceil \frac{n-m}{m} \right\rceil \left(2n - m\left\lceil \frac{n}{m} \right\rceil - 1\right).$$

If it is true, this conjecture is tight for any size of subset. Indeed, it is possible to find automata of the Černý family and subsets of their states that reach the bound for any chosen parameters. The synchronization of subsets of states is also studied in [MN17], in which the authors reformulate Conjecture 1.4.4 and focus specifically on its 3-states version:

Conjecture 1.4.5 ([MN17], section 4)**.** *Given a synchronizing automaton $\mathscr{A} = (Q; \Sigma; \delta)$ with n states and a subset S of Q with 3 elements, there is a synchronizing word $w \in \Sigma^*$ for S, whose length is at most $\frac{2n^2}{3}$.*

Reachability of subsets with short words A complementary question to the synchronization of subsets is of course the reachability of subsets. If a subset is reachable, is it possible to reach it with a short word? This question was explicitly stated by Don:

Conjecture 1.4.6 (Conjecture 18 in [Don16])**.** *Let $\mathscr{A} = (Q; \Sigma; \delta)$ be an n-state automaton. If $S \subset Q$ is a set of size k and there exists a word w such that $Qw = S$, then there exists a word with this property of length at most $n(n-k)$.*

This conjecture is stronger than Conjecture 1.2.2. Indeed, any pair would be reachable with a word of length $n(n-2)$. Since there is also a pair which can be synchronized by a single letter, it provides a synchronizing word of length $(n-1)^2$.

It turns out that this conjecture was too strong in general. Indeed, it is possible to find automata with subsets that cannot be reached with words shorter than an exponential length, as we will see in Chapter 4. However, the conjecture might be true for some classes of automata. In particular, it looks very natural for automata in which every subset of states is reachable, namely *completely reachable automata*:

Problem 1.4.7. *Let $\mathscr{A} = (Q; \Sigma; \delta)$ be a n-state completely reachable automaton. For any $0 < k < n$ and any set $S \in Q$ of size k, is it true that there exists a word w such that $Qw = S$ and $|w| \leq n(n-k)$?*

We notice that Conjecture 1.2.2 is not yet proven for the class of completely reachable automata. Solving Problem 1.4.7 would prove it for this class.

On subset extension The extension method has led to prove Conjecture 1.2.2 for many classes of automata. Therefore, conjectures trying to generalize it appeared naturally. One attempt was made by Rystsov in 1995. He first showed

with the extension method that the reset threshold for *regular automata*[4] was quadratic:

Theorem 1.4.8 ([Rys95], Theorem 5)**.** *Let $\mathscr{A} = (Q; \Sigma; \delta)$ be a n-state regular automaton. Then there is a reset word for $\mathscr{A}$ whose length is at most $2(n-1)^2$.*

Based on this setting, Rystsov conjectured the following:

Conjecture 1.4.9 ([Rys95], Hypothesis)**.** *Any strongly connected automaton is regular.*

This conjecture does not imply Conjecture 1.2.2, but a quadratic bound twice higher. A similar conjecture was proposed by Carpi and d'Alessandro in 2009 [CD09]. However, these two conjectures were disproved by Berlinkov in 2011 [Ber11]. Indeed, he provided a family of strongly connected synchronizing automata which were not regular. Moreover, following the same idea, in [KS15], the authors build synchronizing automata for which there exists subsets of states that can not be extended by words shorter than a quadratic length, therefore deceiving the possibility to use this method to get a general quadratic bound on the reset threshold. However, they proposed the following conjecture, in which the subsets for which the extension method is applied are only the reachable ones:

Conjecture 1.4.10 ([KS15], Conjecture 1)**.** *There exists a constant c such that, for every strongly connected synchronizing automaton $(Q; \Sigma; \delta)$ and every non-empty $S \subset Q$ with $Qw = S$ for some $w \in \Sigma^*$, there exists a word u of length at most cn and a subset $T \subset Q$ such that $T \subseteq Su^{-1}, Qv = T$ for some $v \in \Sigma^*$,and $|T| > |S|$.*

However, the authors showed that the constant c in the conjecture has to be at least $3/2$, therefore Conjecture 1.2.2 cannot be directly proved through this conjecture.

The synchronizing probability function growth The synchronizing probability function of an automaton is a function of a natural number returning a probability of synchronization, i.e. $k_{\mathscr{A}} : \mathbb{N} \to [0, 1]$. The SPF is intended as a tool to understand the synchronizing process of an automaton. Indeed, it is

[4] A regular automaton is an automaton which has a regular collection of words. A collection of words W is called regular if it contains the empty word and there is a natural number $k > 1$ such that for every pair of states s, t, there are exactly k words in W which take the state s into the state t. For example, the automaton $\mathscr{C}_n$ is regular because the set $\{\emptyset, b, \ldots, b^{2n-1}\}$ is a regular collection of words with $k = 2$.

non-decreasing and if $k_{\mathscr{A}}(t) = 1$, then $t \geq rt(\mathscr{A})$. It was noticed that the SPF of all known examples achieving the bound of Conjecture 1.2.2 was growing linearly, and this was also confirmed by numerical experiments. This led to the following conjecture, which states that the synchronizing probability function of a synchronizing automaton grows regularly.

Conjecture 1.4.11 (Conjecture 2 in [Jun12])**.** *In a synchronizing automaton $\mathscr{A}$ with n states, for any $1 \leq j \leq n-1$,*

$$k_{\mathscr{A}}(1 + (j-1)(n+1)) \geq j/(n-1).$$

This conjecture implies that, at each multiple of $n+1$ plus one, the SPF must have reached a threshold.

We notice that this conjecture is stronger than the Černý conjecture (Theorem 4 in [Jun12]). Indeed, for $j = n-2$, the conjecture implies that $k_{\mathscr{A}}((n+1)^2 - 3) \geq (n-2)/(n-1)$. However, the author also showed that if $k_{\mathscr{A}}(t) > (n-2)/(n-1)$ for some t, then $k_{\mathscr{A}}(t+3) = 1$. Therefore, it implies that $k_{\mathscr{A}}((n+1)^2) = 1$, which is Conjecture 1.2.2.

Conjecture 1.4.11 was one of the main motivations at the beginning of this thesis (see Chapter 2). A counterexample was obtained for $j = 2$, however it is not known if the conjecture holds for other values:

Problem 1.4.12. *In a synchronizing automaton Σ with n states, for any $3 \leq j \leq n-1$, is*

$$k_{\Sigma}(1 + (j-1)(n+1)) \geq j/(n-1)?$$

Proving this would still imply Conjecture 1.2.2.

Improvements of the Pin-Frankl bound The last noticeable recent results we will discuss are the improvements on the bound obtained by Pin and Frankl in 1982. This general bound on the reset threshold had been holding for more than 30 years, and was improved twice in the last 3 years.

The first key element to understand these results is to recall the greedy process proposed by Pin: for each k, bound the length of a compressing word for any set of size k. In fact, in the first steps, if $k > n/2$, it turns out that it is possible to find words that compress the set of shorter length than the bound obtained by Pin.

The first observation of this possibility was due to Trathman in [Tra07]. He stated that, if the following statement was true, then it would be possible

to compress Q to a set of states of size $n/2$ with a word of quadratic length. Therefore, the full process would be of length $O(7n^3/48)$ instead of $O(n^3/6) = O(8n^3/48)$.

Statement 1.4.13 (Lemma 3 in [Tra07]). *Let Q be the set of states of a synchronizing strongly connected n-state DFA. Then for any state q there exists a word w of length not greater than n such that $q \notin Qw$. For any $k < n$ there are at least k states $q_1, \ldots, q_k$ and words $w_1, \ldots, w_k$ of length not greater than k such that for any $1 \leq i \leq k$, $q_i \notin Qw_i$.*

As we will see in Chapter 4, this statement is false. However, the bound $O(7n^3/48)$ would still be true with a weaker version of Statement 1.4.13. This weaker version is the possibility to exclude a state from a set with a word of linear length cn, for a constant c. This was proposed as an open problem:

Problem 1.4.14 (Open problem in [GJT15]). *Let Q be the set of states of a synchronizing strongly connected $n-$state DFA.*

Is there a constant c such that, for any state $q \in Q$, there exists a word w of length not greater than cn such that $q \notin Qw$?

This problem has not been solved as it is. However, Szykuła showed that, if a set of states of an automaton was not satisfying Statement 1.4.13, then it is possible to find a word that improves the compression process:

Lemma 1.4.15 (Lemma 2 in [Szy18]). *Let $\mathscr{A} = (Q; \Sigma; \delta)$ be a n-state automaton. Consider a non-empty subset $S \in Q$ and a non-empty proper subset $S_2 \subsetneq S$. Suppose that there is a word $w \in \Sigma^*$ such that $A \nsubseteq Sw$. Then there exists a word w of length at most $n - |S_2|$ satisfying either*

1. *$A \subsetneq Sw$, or*
2. *$|Sw| < S$*

This Lemma, although less powerful than Statement 1.4.13 and Problem 1.4.14, allows to slightly improve the compression of Q to a set of size $n/2$, which leads to the following improved general bound:

Theorem 1.4.16 (Theorem 11 in [Szy18]). *For any n-state synchronizing automaton $\mathscr{A}$, we have:*

$$rt(\mathscr{A}) \leq (85059n^3 + 90024n^2 + 196504n - 10648)/511104.$$

This is an improvement of a factor 1/10000 compared to the bound of Pin. However, starting from this, a new improvement using the same tools was proposed by Shitov two months before the submission of this Thesis. In fact, he showed that the compression word obtained in the process could be linked with each other:

Theorem 1.4.17 (Theorem 2 in [Shi19]). *Let $u \in \Sigma^*$ be a word of length l and deficiency $r \in \{1, n/2 - 1\}$. Assume that, for an integer λ, there exists a word v of length λ such that $rkv \leq rkv'$ for any word v' of length at most $\lambda + 2r$. Then there is a word of length at most $l + \lambda + 2r$ and deficiency at least $r + 1$.*

Using this, he improved the bound as follows:

Theorem 1.4.18 (Corollary 6 and Lemma 7 in [Shi19]). *For any n-state synchronizing automaton $\mathscr{A}$, we have:*

$$rt(\mathscr{A}) \leq \left(\frac{7}{48} + \frac{2 \times 15625}{1597536}\right) n^3 + o(n^3).$$

Which is of the order of $0.1654n^3$, while the original bound of Pin was of the order of $0.1666n^3$.

This improvement is based on the fact that a short-cut is possible in the first $n/2$ steps of the compression process. We notice that it does not improve the second part of Pin's greedy algorithm, which is longer and seems to be harder. In Chapter 3, we focus on the second part of the process by providing an approach based on the square graph.

Some very active research topics and open questions linked to automata are not directly relevant for this thesis and will not be covered here. For example, the generalization and alternative problems based on the road colouring theorem. These problems are studied in [GPS17, VR15, BP14, CD10]. Other very interesting classes of problems are the ones related to collapsing words, synchronization of *partial automata*, in which some edges are undefined (which is called *careful synchronization*), or synchronization of *undeterministic automata*, in which letters can map states on more than one state. Problems of this type are studied in [ACV02, Vor16, Mar13, GING09]. [5]

[5] We notice that the results obtained for this kind of automata are completely different than the ones obtained with DFAs. Indeed, the bounds for careful synchronization [Vor16] and for i-directability of undeterministic automata for $i > 2$ [GING09] are exponential, while the bound for Conjecture 1.2.2 is polynomial.

1.5 Thesis Outline

The outline of the thesis is synthesized in Fig. 1.11. Most of the results have been published in journal or conference papers. In this thesis, we grouped them into four main chapters.

In Chapter 2, we study two concepts which are highly interdependent. The first one is the *synchronizing probability function* of an automaton, a tool introduced by Jungers [Jun12] that reinterprets the synchronizing phenomenon as a two-player game, and allows to obtain optimal strategies through a Linear Program. The second one is the *triple rendezvous time*, i.e., the length of the shortest word synchronizing a set of three states. It is a natural intermediate problem toward Conjecture 1.2.2, and represents the worst-case scenario in the second step of the extension method. We notice that a similar problem has recently been solved for sets of primitive matrices [ACCJ19].

Our contribution is twofold. First, we obtain a new upper bound on the triple rendezvous time by coupling two different approaches based on the synchronizing probability function and properties of linear programming. Second, we disprove a conjecture on the triple rendezvous time by exhibiting a family of counterexamples, which leads to simultaneously disprove a conjecture on the growth of the SPF. We then suggest natural follow-ups towards the Černý conjecture.

In Chapter 3, we study the 2-transitivity of permutation sets and the diameter of the square graph of automata. We first provide a theoretical upper bound on the square graph of permutation sets diameter and a construction reaching a diameter of $\frac{n^2}{4} + o(n^2)$. Second, using this construction, we refine a theorem on the joint spectral radius. Finally, we show that the square graph diameter of a permutation set cannot be equal to the theoretical bound stated before. We then show how this result could be extended to synchronizing automata in order to improve the compression method.

In Chapter 4, we study subset reachability and its counterpart, state exclusion. First, we provide an automata family with subsets that cannot be reached by words shorter than $2^n/n$, thus disproving Conjecture 1.4.6. We then propose and analyze relaxed versions of this conjecture. Second, we provide a counterexample to a lemma used in a recent tentative improvement of the Pin-Frankl bound for synchronizing automata [Tra11]. This example naturally leads us to formulate an open question, whose answer could fix the line of the proof, and improve the bound.

Chapter 5 studies the class of automata which are completely reachable,

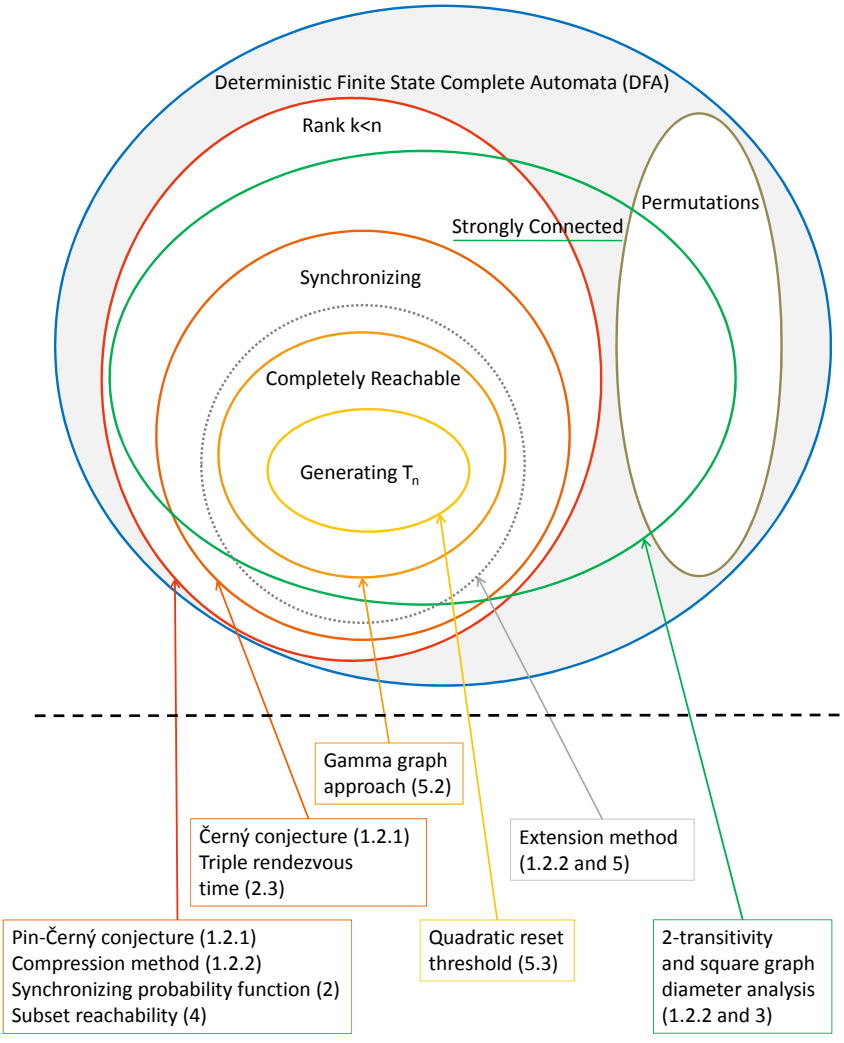


Figure 1.11: Thesis outline. The upper part represents the classes of automata that are met through this thesis and the relations between them. The lower part shows the topics which are studied in the thesis and the sections references. The arrows show on which classes of automata the topics apply.

i.e. in which any subset can be reached. These automata are very particular because it is possible to control them in order to reach any set of states. Within this class, we also focus on automata with the transition monoid equal to the full monoid of transformations.

First, we analyze the Γ_1-graph construction for completely reachable automata. The Γ_1-graph is derived from 1-deficient words, and is a key tool for applying the extension method. We introduce the concept of roots of 1-deficient words, which allows to state explicit concatenation rules for these words. Based on these results, we provide a polynomial-time algorithm for constructing the Γ_1-graph. Then, we disprove a conjecture by Bondar and Volkov [BV16] linking the strong connectivity of this graph and the concatenation of 1-deficient words of completely reachable automata. Finally, we prove an alternative version of this conjecture.

Second, motivated by the Babai conjecture and the Černý conjecture, we study the reset thresholds of automata with the transition monoid equal to the full monoid of transformations of the state set. For automata with n states in this class, we prove that the reset threshold is upper-bounded by $2n^2 - 6n + 5$ and can attain the value $\frac{n(n-1)}{2}$.

Chapter 2

The Triple Rendezvous Time and the Synchronizing Probability Function of Automata

Abstract

In this chapter, we study two concepts which allow to approach the Černý conjecture. The first one is the *triple rendezvous time*, i.e., the length of the shortest word mapping three states onto a single one. The second one is the *synchronizing probability function* of an automaton, a recently introduced tool which reinterprets the synchronizing phenomenon as a two-player game, and allows one to obtain optimal strategies through a Linear Program.

Our contribution is twofold. First, by coupling two different approaches based on the synchronizing probability function and properties of linear programming, we obtain a new upper bound on the triple rendezvous time. Second, by exhibiting a family of counterexamples, we disprove a conjecture on the growth of the synchronizing probability function. We then suggest natural follow-ups towards the Černý conjecture. [1] [2]

[1] Most results presented in this chapter have been published in [GJ15] and [GJ16].
[2] The definitions and notation have been presented in Section 1.3.

2.1 Overview

Let us recall that Conjecture 1.2.2 (the Černý Conjecture) states that any n-state synchronizing automaton has a synchronizing word of length $(n-1)^2$. The philosophy behind this conjecture is to bound the length of the shortest word mapping all the states onto a single one. Based on this idea, one could wonder what is the length of the shortest word for which there exists a set of states *of a given cardinality* mapped onto a single state by this word. For cardinality two the problem is solved, as in any synchronizing automaton there always exists a single letter mapping two states onto a single state. Therefore in that case the answer is "one". For higher values, to the best of our knowledge, the question is open. We analyze the case of cardinality three (introduced in [Jun12]), which we coin the *triple rendezvous time (T_3)*.

The main tool we focus on, the *Synchronizing Probability Function* (SPF), was introduced by Jungers in 2012 [Jun12]. This tool allows the reformulation of the synchronizing property as a game theoretical problem whose solution can be obtained through convex optimization. Convex optimization is a mature discipline with strong theoretical basis (see for instance [BV04]). Our hope is that in this framework, properties of synchronization can be better understood and proved using tools that have not been used on DFA yet.

In Section 2.2 we recall the main properties of the SPF. In Section 2.3, we introduce the concept of triple rendezvous time and, by making use of the SPF, we obtain a new upper bound on this value. In Section 2.4, we first refute a conjecture on the SPF (Conjecture 2 in [Jun12]) by presenting a particular family of automata which does not satisfy it. Second, we analyze the growth of the "relaxed" SPF, if $t < T_3$ and if we only consider the constraints on the underlying linear program. We obtain a tight bound on this growth.

2.2 A Game Theoretical Framework

In this section, we recall definitions and properties of the synchronizing probability function needed to develop our results. A more complete introduction to the SPF and the details of the proofs can be found in [Jun12]. This concept is based on the following two-player game, which gives another perspective on the synchronization of an automaton. For a given automaton and a length t chosen in advance, the rules are as follows:

1. Player Two secretly chooses a state e_j of the automaton.

2. Player One chooses a word W of length at most t.

3. Player One guesses the final state e_jW. If it is the right state, he wins. Otherwise, Player Two wins.

In this game, if the length t is larger or equal to the length of a synchronizing word, a winning strategy for Player One is to take this synchronizing word and the state on which the automaton is mapped. Oppositely, if t is zero, Player One can only choose randomly one of the states, and he has probability $1/n$ of winning. We consider that both players can choose probabilistic strategies.

The *policy of Player Two* is defined as a probability distribution over the states, that is, any vector $p \in \mathbb{R}_+^n, ep^T = 1$. Player Two chooses the state e_j with probability p_j, in which case the automaton ends up at the state corresponding to e_jW. Since Player One wants to maximize the probability of choosing the right final state, he picks up the state where the probability for the automaton to end is maximal, that is,

$$\operatorname*{argmax}_i (pW)e_i^T.$$

Therefore the probability of winning for Player One is

$$\max_{i,W} (pW)e_i^T. \tag{2.1}$$

The aim of Player Two is to minimize that probability.

In the following, $\Sigma^{\leq t}$ is the set of products of length at most t of matrices taken in Σ. By convention, and for the ease of notation, it contains the product of length zero, which is the identity matrix.

Definition 2.2.1 (SPF, Definition 2 in [Jun12])**.** *Let $n \in \mathbb{N}$ and $\Sigma \subset \{0,1\}^{n\times n}$ be an automaton. The* synchronizing probability function *(SPF) of Σ is the function $k_\Sigma : \mathbb{N} \to \mathbb{R}_+$:*

$$k_\Sigma(t) \quad = \quad \min_{p\in\mathbb{R}_+^n,\ ep^T=1} \left\{ \max_{W\in\Sigma^{\leq t},i} (pW)e_i^T \right\}. \tag{2.2}$$

The SPF gives the probability of winning the game that Player One can achieve with parameter t if Player Two plays optimally. If there is no ambiguity on the automaton, we use $k(t)$ for $k_\Sigma(t)$.

The synchronizing probability function is non-decreasing with respect to t. Moreover, its value is one if and only if there is a synchronizing word of length smaller or equal to t. This leads to the following reformulation of Conjecture 1.2.2:

Proposition 2.2.2 (Proposition 1 in [Jun12])**.** *The following conjecture is equivalent to Conjecture 1.2.2:*

If $\Sigma \subset \{0,1\}^{n\times n}$ *is a synchronizing automaton, then,*

$$\forall t \geq (n-1)^2, \quad k_\Sigma(t) = 1.$$

Example 2.2.3. *In order to illustrate the SPF concept, consider the Cerný family automaton with four states. It is represented in Fig. 2.1 and is composed of the following letters:*

$$a = \begin{pmatrix} 1&0&0&0\\ 0&1&0&0\\ 0&0&1&0\\ 1&0&0&0 \end{pmatrix} \; b = \begin{pmatrix} 0&1&0&0\\ 0&0&1&0\\ 0&0&0&1\\ 1&0&0&0 \end{pmatrix}$$

Figure 2.2 represents the SPF of this automaton. For $t = 0$*, Player One can only choose a state randomly, and has probability* $1/4$ *of winning, therefore* $k(0) = 1/4$*. Conversely, as the word abbbabbba is a synchronizing word, if* $t \geq 9$ *Player One can apply this word and ensure the win. This imples that* $k(t) = 1$ *for* $t \geq 9$*.*

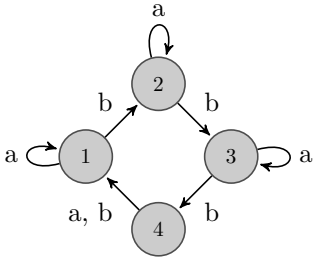


Figure 2.1: The Cerný family automaton with four states.

In order to use the SPF, we need an explicit algorithmic construction of the optimal strategies for both players, which allows us to compute its value. Each basic strategy of Player One, i.e., the choice of a word and a final state,

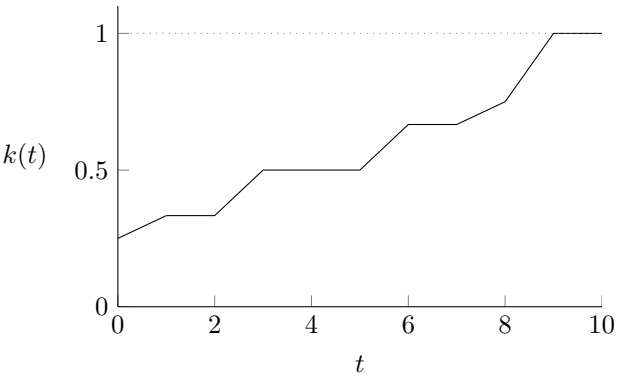


Figure 2.2: The synchronizing probability function of the automaton in Fig. 2.1.

is equivalent to choosing a column in this word. Therefore, we consider the set of all the different columns reached in words of length at most t.

Definition 2.2.4. *We say that a vector is* reachable *at t if it is equal to any column of the words in $\Sigma^{\leq t}$. We denote by $A(t)$ the set of all the reachable vectors at t. We represent $A(t)$ as a $n \times m(t)$ matrix, where $m(t)$ is the number of different reachable vectors at t.*

We notice that $A(t-1) \subseteq A(t)$. In order to have a unique matrix representation for $A(t)$, $t \geq 1$, we choose the first block of $A(t)$ equal to $A(t-1)$, we sort the $m(t) - m(t-1)$ last columns of $A(t)$ by lexicographical order, and we choose $A(0)$ equal to the identity matrix. When there is no ambiguity on t, we use A for $A(t)$.

Example 2.2.5. *Let us go back to the automaton presented in Example 2.2.3. At $t = 3$, $A(3)$ is given by:*

$$A(3) = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 \end{pmatrix}.$$

The first four columns corresponds to $A(0)$, the fifth column comes from the word a, the sixth from the word ba and the seventh from the word bba.

The *policy of Player One* is defined as a probability distribution over the columns of $A(t)$, that is, any column vector $q \in \mathbb{R}_+^{m(t)}$ such that $eq = 1$.

It turns out that the SPF can be efficiently computed thanks to the following linear programs[3].

Theorem 2.2.6 (Theorem 1 in [Jun12])**.** *The* synchronizing probability function $k_\Sigma(t)$ *of a DFA* Σ *is given by*

$$\begin{aligned} \min_{p,k} \quad & k \\ s.t. \quad & pA \leq ke \\ & ep^T = 1 \\ & p \geq 0. \end{aligned} \tag{2.3}$$

It is also given by:

$$\begin{aligned} \max_{q,k} \quad & k \\ s.t. \quad & Aq \geq ke^T \\ & eq = 1 \\ & q \geq 0. \end{aligned} \tag{2.4}$$

In the equations above, A denotes the set of reachable vectors at t (see Definition 2.2.4), p is a $1 \times n$ vector, q is a $m(t) \times 1$ vector, e represents all-ones vectors of the appropriate dimension, 1 is a scalar, and 0 represents zero vectors of the appropriate dimensions.

The linear Program (2.4) is the dual of Program (2.3). In the primal (2.3), the optimal objective value $k(p)$ is obtained with the strategy of Player Two, p, which is a probability distribution on the states. In the dual (2.4), the optimal objective value $k(q)$ is obtained with the strategy of Player One, q, which is a probability distribution on the set of reachable vectors. For any primal feasible solution p and any dual feasible solution q, the objective value $k(p)$ of Program (2.3) and the objective value $k(q)$ of Program (2.4) satisfy $k(p) \geq k(q)$. Therefore, if the objective value k is the same for both programs with feasible solutions p and q, this value is the optimum (see [BV04] for more details on convex optimization and linear programming).

Example 2.2.7. *Let us consider the automaton presented in Example 2.2.3 and $t = 3$. The set of reachable vectors is given in Example 2.2.5. On the one*

[3]The following inequalities are entrywise.

hand

$$p = (1/4, 1/4, 1/4, 1/4)$$

is an admissible solution for Program (2.3) (i.e., a probability distribution on the states for Player Two), which gives as objective value $k(p) = 1/2$*. On the other hand*

$$q = (0, 0, 0, 0, 1/2, 0, 1/2)^T$$

is an admissible solution for Program (2.4) (i.e., a probability distribution on the columns of words in $\Sigma^{\leq 3}$*), which also gives the objective value* $k(q) = 1/2$*. Therefore, the SPF at* $t = 3$ *is equal to* $k(3) = 1/2$ *(as shown in Fig. 2.2). In other words, this means that if both players play optimally, with words of length at most 3, Player One has probability* $1/2$ *of winning the game.*

By leveraging classical results from convex optimization, one can derive strong properties on the optimal strategies in the above-defined game.

Theorem 2.2.8 (Theorem 2 in [Jun12])**.** *For any pair of optimal solutions* (p^*, q^*) *of Programs (2.3) and (2.4), we have*
$q_j^*(k - (p^*A)_j) = 0$ *for all* $1 \leq j \leq m(t)$ *and*
$p_i^*((Aq^*)_i - k) = 0$ *for all* $1 \leq i \leq n$*.*

In the following, our main arguments are based on the dimension of the set of optimal strategies of Program (2.4):

Definition 2.2.9 (Definition 3 in [Jun12])**.** *Let* Σ *be an automaton and* t *be a positive integer. The polytopes* P_t *and* Q_t *are the sets of optimal solutions of respectively Program (2.3) and Program (2.4). We call* dimension *of a polytope the dimension of the smallest affine subspace containing the polytope.*

Example 2.2.10. *In Example 2.2.5,* P_3 *and* Q_3 *are the following sets:*

$$P_3 = \left\{ (1/4 + x, 1/4 - x, 1/4 + y, 1/4 - y) \;\middle|\; \begin{array}{l} -1/4 \leq x \leq 1/4, \\ -1/4 \leq y \leq 1/4, \\ x - y \leq 0 \end{array} \right\}$$

$$Q_3 \quad = \quad \{(0, 0, 0, 0, 1/2, 0, 1/2)^T\}.$$

These are the only solutions allowing for an objective value $k = 1/2$*.*

Moreover, if the SPF does not increase, we have the following result on P_t:

Lemma 2.2.11 (Lemma 1 in [Jun12]). *If $k(t) = k(t+1)$ then $P_{t+1} \subset P_t$.*

With these tools in hand, we can get to our contributions.

2.3 A New Bound on the Triple Rendezvous Time

The *triple rendezvous time (T_3)* is the length of the shortest word mapping three states of the automaton onto a single one. Although it is a very natural concept, we are not aware of any attempts to bound its value for synchronizing automata. In what follows, the *weight* of a vector is the number of its non-zero elements.

Definition 2.3.1. *For a synchronizing automaton Σ, the* triple rendezvous time $T_{3,\Sigma}$ *is defined as the smallest integer t such that $A(t)$ contains a column of weight superior or equal to 3.*

In other words, the triple rendezvous time is the length of the shortest word W such that there exist states q_i, q_j and q_k with $q_iW = q_jW = q_kW$. In the following, we use T_3 for $T_{3,\Sigma}$ when there is no ambiguity on the automaton. In terms of the power automaton, the T_3 is equal to the length of the shortest path between a triple and a single state.

Example 2.3.2. *The example automaton of Figure 2.1 has a triple rendezvous time of 5. Indeed the word*

$$abbba = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}$$

has a column of weight 3. It is the shortest word mapping three states onto a single one, as shown in the power automaton in Fig.2.3.

Of course, we can extend this concept to T_l, the length of the shortest word which maps l states onto a single one, i.e., the length of the shortest word W such that there exist l states $q_i, q_j, \ldots$ such that $q_iW = q_jW = \ldots$. We notice that T_n is the length of the shortest synchronizing word, and that for any synchronizing automaton, $T_2 = 1$.

Our motivations for studying the triple rendezvous time are numerous. First, it is a natural problem related to Conjecture 1.2.2, and it allows for

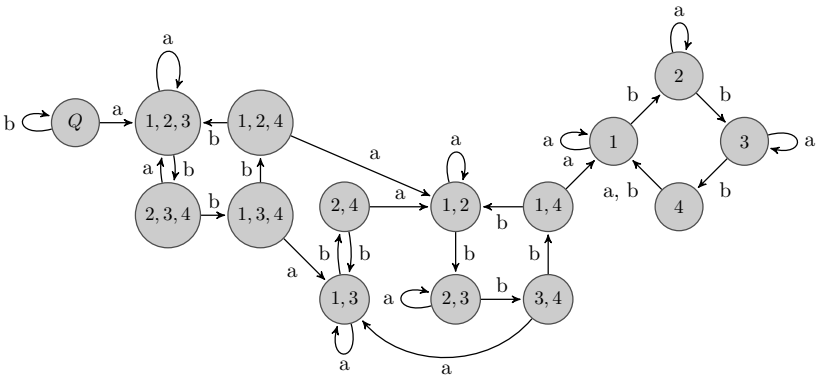


Figure 2.3: Power automaton of the automaton of Figure 2.1

new approaches to study synchronization properties of an automaton. Second, the T_3 is directly linked with the evolution of the synchronizing probability function (Proposition 6 and Conjecture 4 in [Jun12]), and is also related to the $k-$extension property developed in [Ber11][4]. Third, the triple rendezvous time can also be used as an indicator to find automata with large reset threshold. Indeed, for many classes of automata, the value of the T_3 seems empirically to be correlated with the length of the shortest reset word: for random synchronizing automata (from the framework of [ST11]), the reset threshold is small and $T_3 = 1$ with high probability[5][6]. Oppositely, all the automata known in the literature which achieve the bound of Conjecture 1.2.2 do have a large T_3 value [7][8](close to the number of states). As the triple rendezvous time is much easier to compute than the shortest synchronizing word, it can be used as a heuristic filter in order to generate automata with large reset threshold.

[4]In the terminology of [Ber11], the T_3 can be defined as the smallest integer such that there is a pair of states which are synchronized by some single letter, and which is $(T_3 - 1) -$ *extendable*.

[5]An event that occurs with high probability is one whose probability depends on a certain number n and such that its limit is 1 as n goes to infinity, i.e. it can be made as close as desired to 1 by making n big enough. In our case, we consider the size n of the automaton, and the event that $T_3 = 1$.

[6]In the framework of [ST11], automata are synchronizing with high probability and have with high probability three states mapped on a single one by one letter, which implies that $T_3 = 1$ with high probability.

[7]The reader can easily check that the automata of the Černý family have $T_3 = n + 1$, the "Kari automaton" (an automaton with 6 states, see [Kar01]) has $T_3 = 5$ and the "Roman automaton" (an automaton with 5 states, see [Rom08]) has $T_3 = 5$.

[8]It is possible to build automata with quadratic reset threshold and low T_3, see [AVZ07] for such examples.

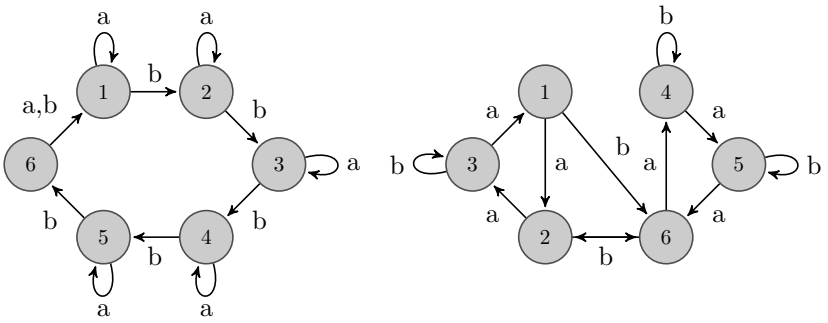


Figure 2.4: On the left the automaton of the Černý family with 6 states, on the right the "Kari automaton".

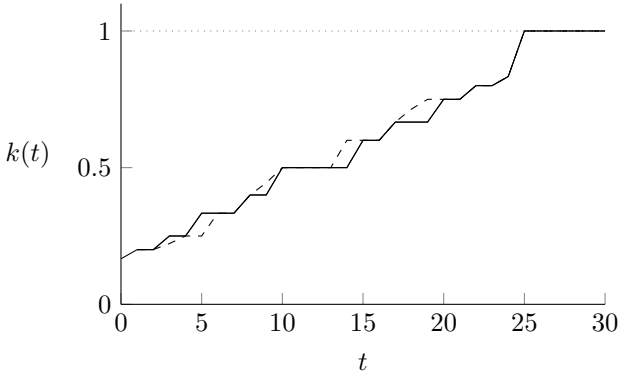


Figure 2.5: Synchronizing probability function for the automaton of the Černý family with 6 states (solid curve), and for the "Kari automaton" (dashed curve).

Figure 2.4 shows the automaton of the Černý family with 6 states and the "Kari automaton" [Kar01], two automata achieving the bound of Conjecture 1.2.2, with synchronizing words of length 25. Figure 2.5 shows their SPF. For these automata, the T_3 is equal to 7 and 5 respectively.

The following conjecture has been made on the behaviour of the SPF:

Conjecture 2.3.3 (Conjecture 2 in [Jun12])**.** *In a synchronizing automaton* Σ *with* n *states, for any* $1 \leq j \leq n-1$*,*

$$k_\Sigma(1 + (j-1)(n+1)) \geq j/(n-1).$$

This conjecture is stronger than the Černý conjecture (Theorem 4 in [Jun12]). Conjecture 2.3.3 would also imply (see [Jun12]) that the following conjecture about the triple rendezvous time is true:

Conjecture 2.3.4 (Conjecture 4 in [Jun12])**.** *In a synchronizing automaton Σ with n states,*

$$T_{3,\Sigma} \leq n + 2.$$

In Section 2.4, we provide a family of automata which are counterexamples for both Conjecture 2.3.3 and Conjecture 2.3.4.

2.3.1 One-Sided Approach of the Triple Rendezvous Time

We now focus on bounding the T_3. A first upper bound can be easily obtained without using the SPF:

Proposition 2.3.5. *In a synchronizing automaton Σ with n states,*

$$T_{3,\Sigma} \leq \frac{n(n-1)}{2} + 1.$$

Proof. Firstly, there are only $n+n(n-1)/2$ possible different columns of weight one or two in A. Secondly, for any positive integer t smaller than the reset threshold (therefore also smaller than T_3), $A(t+1)$ must contain columns that are not in $A(t)$ (Lemma 1 in [Jun12]). Therefore, as $A(0)$ includes the n columns of weight one, $A(n(n-1)/2+1)$ includes at least $n+n(n-1)/2+1$ columns. As all the columns in A are different, it implies that $A(n(n-1)/2+1)$ contains at least a column of weight superior or equal to 3. □

In order to obtain a better upper bound on the T_3, we study the evolution of the SPF and $A(t)$ with respect to t, for $t < T_3$. In that situation, $A(t)$ only contains columns of weight one or two. In the following, we name *subset of columns* of a matrix $A \in \mathbb{R}^{n\times m}$ any matrix $A' \in \mathbb{R}^{n\times m'}$ (for some $m' \leq m$) obtained by erasing columns from A. A subset of columns of A with some property is the subset of columns obtained by erasing the columns of A not satisfying that property, and keeping all the others.

Definition 2.3.6. *We associate with $A(t)$ a graph $G(t)$ with n vertices such that the subset of columns of weight two of $A(t)$ is the incidence matrix of $G(t)$.*

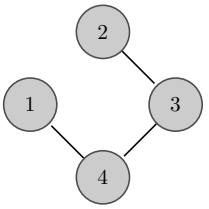


Figure 2.6: Graph $G(3)$ associated with $A(3)$ for the automaton from Example 2.2.5.

Example 2.3.7. *For the automaton from Example 2.2.5, at $t = 3$, the graph $G(3)$ associated with $A(3)$ is shown in Fig. 2.6.*

In the graph $G(t)$ associated with $A(t)$, we call a *singleton* a vertex which is disconnected from the rest of the graph, a *pair* two vertices which are connected to each other and disconnected from the rest of the graph, and a *cycle* a set of vertices forming a cycle[9] and disconnected from the rest of the graph. We call a cycle *odd* (resp. *even*) if it contains an odd (resp. even) number of vertices.

We also use the reverse correspondence. Given a graph G on n vertices, each vertex v_j is represented by e_j^T, and an edge (v_i, v_j) is represented by the vector $(e_i + e_j)^T$. We notice that, if $1 < t < T_3$, since $A(t)$ contains at least one column that is not in $A(t-1)$, $G(t)$ has at least one more edge than $G(t-1)$.

The strategy that we use to bound the T_3 is the following: first, based on this matrix-graph approach, we study the values that $k(t)$ can take with $t < T_3$ (Lemma 2.3.10). Then, we bound the maximal dimension that the polytope P_t of solutions of Program (2.3) can take for each value $k(t)$ (Theorem 2.3.11). We conclude by using the fact that this maximum strictly decreases if $k(t)$ stays constant.

To do so, we start from the set of reachable vectors $A(t)$. We prove that it is possible to find a subset of columns $A_c(t)$ of $A(t)$ satisfying the following properties; the matrix $A_c(t)$ is such that its associated graph $G_c(t)$ is composed of disjoint singletons, pairs and odd cycles, and is such that the optimal objective value for Program (2.3) and Program (2.4) is the same if $A(t)$ is replaced by $A_c(t)$ (Lemma 2.3.8). This structure allows us to easily compute the value $k(t)$ and the dimension of the corresponding polytope of optimal solutions. Note that when replacing $A(t)$ with $A_c(t)$, the dimension of P_t is non-decreasing

[9] c vertices are forming a *cycle* if we can number them from 1 to c in such a way that vertex 1 is only connected to vertices 2 and c, each vertex $1 < i < c$ is only connected to vertices $i-1$ and $i+1$, and the vertex c is only connected to vertices $c-1$ and 1.

with respect to t, as every optimal solution of Program (2.3) with $A(t)$ is also an optimal solution with $A_c(t)$.

We call *support* of the strategy q of Program (2.4) the set of columns in $A(t)$ corresponding to non-zero entries in q. We call a column of $A(t)$ *critical* if there exists an optimal solution q with a non-zero entry corresponding to this column.

In the proof of Lemma 2.3.8 hereunder, we split the program into two subprograms and introduce notation to get to our arguments. The reader may refer to Example 2.3.9 to have an illustration of the steps.

Lemma 2.3.8. *If $t < T_3$, there exists an optimal solution q for Program (2.4) such that its support is associated with a graph composed of disjoint singletons, pairs, and odd cycles.*

Proof. We first present the structure of the proof, then we demonstrate the technical elements.

Structure of the proof. We proceed by induction on the number of rows of matrix $A(t)$, which is also the number of vertices of $G(t)$. If there is only one or two vertices, the lemma is trivially true. Otherwise, we use the fact that the graph associated with a subset of columns of $A(t)$ can either be connected or disconnected.

If it is disconnected, we can define two subprograms with the structure of Program (2.4) with less variables, for which we know by induction that there exist optimal solutions satisfying the lemma. From these solutions, we can build an optimal solution of the original program whose support is also composed of disjoint singletons, pairs, and odd cycles.

If it is connected, we can either find an optimal solution with a support associated with an odd cycle including all the vertices, or find a solution with a support associated with a disconnected graph. In this latter case, we are again able to split the program as in the disconnected case.

Demonstration. Let us write $Prog_A$ for the original Program (2.4), and $Prog_N$ for Program (2.4) in which matrix A is replaced by a matrix N, with the corresponding dimensions for vectors q and e. Let us write k_A and k_N the optimal objective value of $Prog_A$ and $Prog_N$.

We suppose by induction that the Lemma holds if $A(t)$ is associated with a graph with $n-1$ vertices or less (so if $A(t)$ has $n-1$ rows or less). Let us now consider a matrix $A(t)$ associated with a graph $G(t)$ with n vertices. Let A_{min} be a *minimal subset* of columns of $A(t)$ (that is, such that $k_A = k_{A_{min}}$, and $k'_A < k_A$ for any strict subset of columns A' of A_{min}) and with not more

than n columns (such a matrix exists, see [Jun12, Proposition 4]). Let G_{min} be the graph associated with A_{min}. We prove that either G_{min} is already composed of disjoint singletons, pairs and odd cycles, or that we can build an optimal solution q_c to $Prog_{A_{min}}$ such that the graph associated with its support is composed of disjoint singletons, pairs and odd cycles. Since the program $Prog_{A_{min}}$ has the same optimal objective value as $Prog_A$, there is a vector q which is an optimal solution of $Prog_{A_{min}}$ (such that $A_{min}q \geq k_A e_{n\times 1}$).

Two cases can occur: either G_{min} is connected, or it is disconnected.

Disconnected case. If G_{min} is disconnected, then the graph can be split into two separate graphs G_{min1}, G_{min2} with respectively n_1 and n_2 vertices. For these two graphs, let us consider the associated matrices A_{min1} and A_{min2} (with n_1 and n_2 rows respectively). We call $Prog_{A_{min1}}$ and $Prog_{A_{min2}}$ *subprograms* of $Prog_{A_{min}}$.
Now define q_1 and q_2 the subvectors of q corresponding to the graphs G_{min1} and G_{min2} respectively. That is, the entries of q_1 are the same as the entries of q corresponding to columns from $Prog_{A_{min1}}$, and the entries of q_2 are the same as the entries of q corresponding to columns from $Prog_{A_{min2}}$. As q is optimal, we have that $A_{min1}q_1 \geq k_A e_{n_1\times 1}$ and $A_{min2}q_2 \geq k_A e_{n_2\times 1}$ (these inequalities and the following ones are entrywise).
Then define $w_1 = 1/(eq_1)$, $w_2 = 1/(eq_2)$, the inverse of the weight associated with each subprogram in the strategy q.
We have that q_1w_1 and q_2w_2 are feasible solutions for $Prog_{A_{min1}}$ and for $Prog_{A_{min2}}$ respectively, so

$$A_{min1}q_1w_1 \geq w_1k_A e_{n_1\times 1}$$

and

$$A_{min2}q_2w_2 \geq w_2k_A e_{n_2\times 1}.$$

Therefore, $k_{A_{min1}} \geq w_1k_A$ and $k_{A_{min2}} \geq w_2k_A$.
By the induction hypothesis, for $Prog_{A_{min1}}$ and for $Prog_{A_{min2}}$, there are vectors r_1 and r_2 which are optimal solutions and have a support associated with a graph composed of singletons, pairs and odd cycles. Therefore for these vectors we have

$$A_{min1}r_1 \geq k_{A_{min1}}e_{n_1\times 1} \geq w_1k_A e_{n_1\times 1}$$
$$A_{min2}r_2 \geq k_{A_{min2}}e_{n_2\times 1} \geq w_2k_A e_{n_2\times 1}.$$

Let us now extend r_1 and r_2 to vectors q_{c1} and q_{c2} of the same length as q, with the entries corresponding to the related subprogram equal to the ones of r_1 and r_2, and the other entries equal to zero. For these strategies, we have that $Aq'_1 \geq w_1 k_A e$ and $Aq'_2 \geq w_2 k_A e$ from the argument above. Now defining $q_c = q_{c1}/w_1 + q_{c2}/w_2$, we have $e^T q_c = 1$ and $Aq_c \geq k_A$, which implies that q_c is an admissible optimal solution for $Prog_A$.
Since the union of two graphs composed of disjoint singletons, pairs and odd cycles is also composed of disjoint singletons, pairs and odd cycles, the graph G_c associated with the support of the strategy q_c is composed of disjoint singletons, pairs and odd cycles, as wanted.

Connected case. For the connected case, if G_{min} is connected and $n \geq 3$, we can either show that G_{min} has no vertex of degree one, or build an optimal solution such that its support is disconnected, which leads to the previous case. Indeed, consider an edge (v_1, v_2) with the vertex v_2 of degree one. If v_1 is of degree one, then (v_1, v_2) is a disconnected pair and G_{min} is not connected. If v_1 has other adjacent edges (v_1, v_i), with $i > 2$, one can obtain a new solution in which the value corresponding to these other edges is equal to zero, without changing k_A and without adding value to edges that are not in the support of the current solution.

For the description of this construction, let us denote by q_{v_α} the entry of q corresponding to the vertex v_α, and by $q_{(v_\beta, v_\gamma)}$ the entry of q corresponding to the edge (v_β, v_γ), with $1 \leq \alpha \leq n$ and $1 \leq \beta \neq \gamma \leq n$. From the solution q, we build a vector q' of the same dimension as follows:

$$\begin{aligned} q'_{v_1} = q'_{v_2} &= 0, && (2.5)\\ q'_{(v_1,v_2)} &= q_{v_1} + q_{v_2} + q_{(v_1,v_2)}, \\ q'_{(v_1,v_i)} &= 0 && \text{for } i > 2 \text{ and } (v_1, v_i) \in G_{min}, \\ q'_{v_i} &= q_{(v_1,v_i)} + q_{v_i} && \text{for } i > 2 \text{ and } (v_1, v_i) \in G_{min}, \end{aligned}$$

and all the other entries of q' equal to the entries of q.

With this construction, we have the following inequalities:

$$q'_{v_1} + \sum_{j \neq 1,\ (v_1,v_j) \in G_{min}} q'_{(v_1,v_j)} = q_{v_1} + q_{v_2} + q_{(v_1,v_2)} \geq k_A,$$

$$q'_{v_2} + \sum_{j \neq 2,\ (v_2,v_j) \in G_{min}} q'_{(v_2,v_j)} = q_{v_1} + q_{v_2} + q_{(v_1,v_2)} \geq k_A,$$

as $q_{v_2} + q_{(v_1,v_2)} \geq k_A$, because q is an optimal solution. Moreover, for every other $i > 2$,

$$q'_{v_i} + \sum_{j\neq i,\ (v_i,v_j)\in G_{min}} q'_{(v_i,v_j)} = q_{v_i} + \sum_{j\neq i,\ (v_i,v_j)\in G_{min}} q_{(v_i,v_j)} \geq k_A.$$

In addition, it is straightforward from (2.5) that $\sum_{i=1}^{m(t)} q'_i = \sum_{i=1}^{m(t)} q_i = 1$. Therefore q' and k_A satisfy the conditions of Program (2.4), and q' is an optimal strategy.

The value associated with the edges (v_1, v_i), $i > 2$, in q' is 0, so these edges are not in the support of q', and the pair (v_1, v_2) is disconnected. Therefore either G_{min} has no vertex of degree one, or we can build a solution with a disconnected support.

Now, notice that a connected graph with not more than n edges and no vertex of degree one cannot have vertices of degree larger than two (because the sum of the degrees is equal to twice the number of edges). Therefore, all the vertices are in the same cycle. If n is odd, we have our result. If n is even, let us label its vertices $v_1, \ldots, v_n$ in the cyclic order and define the vector q' of the same dimension as q as follows:

$$\begin{aligned}
q'_{v_i} &= 0, && \text{for } 1 \leq i \leq n,\\
q'_{(v_{2j-1},v_{2j})} &= q_{v_{2j-1}} + q_{v_{2j}} + q_{(v_{2j-1},v_{2j})} + q_{(v_{2j},v_{2j+1})} && \text{for } 1 \leq j \leq n/2-1,\\
q'_{(v_{n-1},v_n)} &= q_{v_{n-1}} + q_{v_n} + q_{(v_{n-1},v_n)} + q_{(v_n,v_1)},\\
q'_{(v_{2l},v_{2l+1})} &= 0 && \text{for } 1 \leq l \leq n/2-1,\\
q'_{(v_n,v_1)} &= 0.
\end{aligned}$$

We can verify that q' is a feasible solution reaching the objective value k_A. Since the edges (v_{2l}, v_{2l+1}) for $1 \leq l \leq n/2$ and the edge (v_n, v_1) are not in its support, it leads again to a disconnected case, and the proof is done. □

Example 2.3.9. *The purpose of this example is to illustrate the steps of Lemma 2.3.8 in the disconnected case.*

We start from Example 2.2.5. We have

$$A(3) = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0\\ 0 & 1 & 0 & 0 & 0 & 0 & 1\\ 0 & 0 & 1 & 0 & 0 & 1 & 1\\ 0 & 0 & 0 & 1 & 1 & 1 & 0 \end{pmatrix}.$$

A possible minimal subset of columns of $A(3)$ *is:*

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

which is the support of the solution of the original program

$$q = (0, 0, 0, 0, 1/2, 0, 1/2)^T.$$

The associated graph G_{min} *is represented in Fig. 2.7. It is composed of two pairs. It is already composed of singletons, pairs, and odd cycles, however we continue the analysis to illustrate the ideas.*

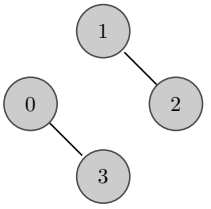


Figure 2.7: Graph G_{min} associated with A_{min}.

As the graph is disconnected, we separate it into G_{min1} *and* G_{min2}*, being the subgraphs induced by respectively* $\{0, 3\}$ *and* $\{1, 2\}$*. This gives us the incidence matrices* $A_{min1} = (1, 1)^T$ *and* $A_{min2} = (1, 1)^T$*. We now consider the programs* $Prog_{A_{min1}}$ *and* $Prog_{A_{min2}}$*. Optimal solutions of these subprograms are* $r_1 = (1)$ *and* $r_2 = (1)$*, which each have a support forming a single pair (in a general setting we have to use the induction argument to prove the existence of solutions with support associated to a graph composed of singletons, pairs and odd cycles).*

From the strategy q*, we obtain* $w_1 = 2$ *for* $Prog_{A_{min1}}$ *and* $w_2 = 2$ *for* $Prog_{A_{min2}}$*. Expanding these solutions gives us* $q_{c1} = (1, 0)^T$*,* $q_{c2} = (0, 1)^T$*. This leads to the solution of the original program* $q_c = q_{c1}/w_1 + q_{c2}/w_2 = (1/2, 1/2)^T$*, which has a support composed of two pairs.*

It turns out that, given a set $A(t)$, the knowledge of a subset of its columns which is the support of an optimal solution of Program (2.4) and is composed of disjoint odd cycles, pairs and singletons allows one to easily compute the

SPF $k(t)$. Moreover, it provides an upper bound on the dimension of the set of solutions P_t:

Lemma 2.3.10. *If the graph $G(t)$ associated with $A(t)$ is composed of disjoint odd cycles, pairs and singletons, then the optimum of Program (2.4) is given by $2/(n+n_1)$, where n_1 is the number of singletons. Moreover, the dimension of P_t is the number of pairs.*

Proof. To make notations concise, we define $K = n + n_1$. Our claim is that $k(t) = 2/K$. We provide an admissible solution for the primal Program (2.3), as well as for the dual Program (2.4), with the same objective value. Therefore this value is optimal.

A solution of Program (2.3) can be built as follows:

$$p_i = \begin{cases} 2/K & \text{if state } i \text{ corresponds to a singleton vertex,} \\ 1/K & \text{otherwise.} \end{cases} \tag{2.6}$$

The sum of the coefficients is $\sum_{i=1}^{n} p_i = (n-n_1)/K + 2n_1/K = 1$, and p is a feasible solution for Program (2.3) with an objective value of $2/K$.

For Program (2.4), a solution can be built as follows:

$$q_i = \begin{cases} 2/K & \text{if column } i \text{ corresponds to a pair,} \\ 1/K & \text{if column } i \text{ corresponds to an edge of an odd cycle,} \\ 2/K & \text{if column } i \text{ corresponds to a singleton.} \end{cases} \tag{2.7}$$

The sum of the coefficients is $\sum_{i=1}^{m(t)} q_i = n_1 \times 2/K + (n-n_1) \times 1/K = 1$, and q is a feasible solution for Program (2.4) with objective value of $2/K$. Summarizing, Equation (2.6) describes a solution for Program (2.3), and Equation (2.7) describes a solution for Program (2.4), achieving the same objective value. As the programs are dual, this implies that both strategies are optimal, and $k(t) = 2/K$.

We can now give an explicit expression for P_t. Reordering the vertices such that the first f indices correspond to singletons, the next g indices correspond to vertices in pairs, grouped by pair (two vertices in the same pair have indices $f+2j-1$ and $f+2j$, $1 \leq j \leq g/2$), and the last h indices correspond to vertices in odd cycles, we have the following set P_t of optimal solutions for

Program (2.3):

$$\left\{ (p_1, ..., p_{f+g+h}) \;\middle|\; \begin{array}{ll} p_i = 2/K, & \text{with } 1 \leq i \leq f, \\ p_{f+2j-1} = 1/K + x_j, & \\ p_{f+2j} = 1/K - x_j, & \text{with } 1 \leq j \leq g/2 \text{ and} \\ & -1/K \leq x_j \leq 1/K, \\ p_k = 1/K, & \text{with } f+g+1 \leq k, \\ & k \leq f+g+h \end{array} \right\} \quad (2.8)$$

One can check that the vectors p described in (2.8) are feasible solutions for Program (2.3). Each element of the set P_t is uniquely defined by $g/2$ independent parameters x_j that can vary between $-1/K$ and $1/K$, so the dimension of P_t is $g/2$. □

We now combine these lemmas to obtain a universal upper bound on the triple rendezvous time for synchronizing automata. The main steps of the reasoning are as follows: starting from any original program obtained from an automaton and a value t, due to Lemma 2.3.8, one can find a subset of columns $A_c(t)$ of $A(t)$ such that $Prog_{A_c(t)}$ has the same objective value as $Prog_{A(t)}$ and such that the associated graph satisfies the conditions of Lemma 2.3.10. For this program, we can easily compute the value of the SPF and an upper bound on the dimension of P_t. Then, using dimensional arguments, we obtain a lower bound on the SPF growing rate with respect to t, for $t < T_3$:

Theorem 2.3.11. *If $t < T_3$, then $k(t)$ can only take the values $2/(n+s)$, $0 \leq s \leq n-1$, and this value cannot be optimal at more than $\lfloor (n-s)/2 \rfloor + 1$ consecutive values of t.*

Proof. Let us fix $t < T_3$. By Lemma 2.3.8, one can find a matrix A_c which is a subset of the columns of $A(t)$, such that the associated graph G_c is composed of singletons, pairs and odd cycles, and such that $k_A(t) = k_{A_c}$. Let R_t be the set of optimal solutions of $Prog_{A_c}$. From Lemma 2.3.10, the dimension of R_t is the number of pairs in G_c. Let s be the number of singletons. There are $n-s$ vertices of G_c which are either in pairs or in odd cycles, so there are at most $(n-s)/2$ pairs. The set of optimal solutions R_t of $Prog_{A_c}$ has a dimension not smaller than the set of optimal solutions P_t of $Prog_A$, because optimal solutions of $Prog_A$ are also optimal solutions of $Prog_{A_c}$. Therefore we have that $\dim(P_t) \leq \dim(R_t) \leq (n-s)/2$.

We claim that, if $k(t+1) = k(t)$, then $\dim(R_{t+1}) < \dim(R_t)$. Equation (2.7) gives the set of optimal solutions, in which a positive value is assigned to each of

the columns of $A_c(t)$, so it is a set of critical columns. We use a similar argument as in the proof of [Jun12, Theorem 3] to obtain that $\dim(R_{t+1}) < \dim(R_t)$.

Hence we suppose that $k(t+1) = k(t)$. First recall that, from Lemma 2.2.11,

$$P_{t+1} \subset P_t.$$

We define

$$A'(t+1) = A_c(t) \cup \{MA_c(t) : M \in \Sigma\}$$

and P'_{t+1} as the set of solutions $p \geq 0, e^T p = 1$, such that $p^T A'(t+1) = ke$. Note that $P_{t+1} \subset P'_{t+1}$, since the columns in $A'(t+1)$ are a subset of the critical columns in $A(t+1)$. Also, $P'_{t+1} \subset R(t)$ since the columns in $A_c(t)$ are a subset of columns in $A'(t+1)$.

We first show that $P'_{t+1} \neq R_t$. Indeed, since $A'(t+1)$ contains all the columns of $A_c(t)$ multiplied by a matrix in Σ, it is clear that

$$\forall M \in \Sigma, \forall p \in P'_{t+1}, M^T p \in R_t.$$

Supposing $P'_{t+1} = R_t$, the above equation implies that $B^T R_t \subset R_t$ for all $B \in \Sigma^s, s \geq 1$, which implies that Σ is not synchronizing. Indeed, this implies that for all $p \in R_t$, for all $B \in \Sigma^s, p^T B \leq ke^T$.

So, there must be a matrix $M \in \Sigma$ and a column a_j of $A_c(t)$ such that $Ma_j \notin A_c(t)$. Again, since $MA_c(t)q \geq ke, Ma_j$ is a new critical column.

Now, by Theorem 2.2.8, the new critical column Ma_j is such that $p^T Ma_j = k$ for all $p \in P'_{t+1}$. Let H be the hyperplane represented by this constraint. Since $R_t \cap H \neq R_t$, and $R_{t+1} \subset R_t \cap H$, it follows that $\dim(R_{t+1}) < \dim(R_t)$, which proves the claim.

Therefore, as $\dim(R_t) \leq \lfloor (n-s)/2 \rfloor$, if $k(t) = k(t + \lfloor (n-s)/2 \rfloor + 1)$, the dimension of $(R_{t+\lfloor (n-s)/2 \rfloor + 1})$ would have to be negative, which is impossible. This implies that $k(t + \lfloor (n-s)/2 \rfloor + 1) > k(t)$. □

With Theorem 2.3.11, we can now obtain the bound:

Theorem 2.3.12. *In a synchronizing automaton Σ with n states,*

$$T_{3,\Sigma} \leq \frac{n(n+4)}{4} - \frac{n \bmod 2}{4}.$$

Proof. By Theorem 2.3.11, if $t < T_3$, $k(t)$ can only take the values $2/(n+s)$, with $0 \leq s \leq n-1$. Moreover, if $k(t) = 2/(n+s)$, then $k(t + \lfloor (n-s)/2 \rfloor + 1) >$

$k(t)$. Summing over all possible values for $k(t)$ if $t < T_3$, one gets

$$\sum_{s=0}^{n-1} (\lfloor (n-s)/2 \rfloor + 1) = \sum_{s=1}^{n} (\lfloor s/2 \rfloor + 1) = \frac{n(n+4)}{4} - \frac{n \bmod 2}{4}. \tag{2.9}$$

□

2.3.2 Two-Sided Approach of the Triple Rendezvous Time

We now present another technique that provides an alternative upper bound on the triple rendezvous time. It is based on the observation that, if the synchronizing probability function at $t = n$ is small, the T_3 is also small. Then, coupling this with Theorem 2.3.11, we obtain a further improvement of the bound.

Lemma 2.3.13. *For $1 \leq s \leq n/2$, if $k(n) < \frac{1}{n-s}$, then for any word $W \in \Sigma^{\leq n}$, there are less than s zero entries in eW.*

Proof. Suppose that there is a word $L \in \Sigma^{\leq n}$ such that eL has at least s zero entries, and therefore at most $n - s$ non-zero entries. Let p be an optimal solution for Program (2.3). Applying the word L to the automaton, the final probability distribution on the states is pL. Entrywise, $pL \leq eL$, therefore pL has at most $n - s$ non-zero entries. As the sum of the entries of pL is equal to one, the average of its non-zero entries is at least $1/(n-s)$. Therefore, at least one of the entries is higher than or equal to $1/(n-s)$. As p is an optimal solution, it implies that $k(n) \geq \frac{1}{n-s}$. Therefore we have the contrapositive: if $k(n) < \frac{1}{n-s}$, for any word $W \in \Sigma^{\leq n}$ there are less than s zero entries in eW. □

Now, based on Lemma 2.3.13, we have the following result on the triple rendezvous time:

Lemma 2.3.14. *For any strongly connected synchronizing automaton with n states, and for any integer $1 \leq s \leq n/2$, either*

$$k(n) \geq \frac{1}{n-s}$$

or $T_3 \leq n(s+2)/2$.

Proof. Let us fix s as in the statement of Lemma 2.3.13. We suppose that $k(n) < \frac{1}{n-s}$, and our goal is to prove that $T_3 \leq n(s+2)/2$. Let t_s be the

minimal t such that $G(t_s)$ has a vertex of degree larger than s, and let v be such a vertex. As there is at least one more edge in $G(t+1)$ than in $G(t)$, and there are n vertices, we have that $t_s \leq sn/2$. This means that there exists a state q corresponding to v, s states q_i corresponding to the vertices connected to v in $A(sn/2)$, and words $W_i \in \Sigma^{\leq sn/2}$, with $1 \leq i \leq s$, such that $qW_i = q_iW_i$.

We now build a word with a column of weight three. As the graph is strongly connected, starting with a letter having a column of weight two, there exists a word W_1 of length at most n and two states q_1 and q_2 such that $q_1W_1 = q_2W_1 = q$. From Lemma 2.3.13, there are less than s zeros in any product of length n. So, there exists one state q_3 such that its corresponding vertex is connected to v in $G(sn/2)$, and such that its corresponding entry in eW_1 is at least one. Let us call W_2 the word of length at most $sn/2$ such that $qW_2 = q_3W_2$. The word W_1W_2 is such that three states are mapped to a single state. Therefore either $T_3 \leq sn/2 + n$ or $k(n) \geq \frac{1}{n-s}$. □

Coupling Theorem 2.3.11 and Lemma 2.3.14, we obtain the following upper bound:

Proposition 2.3.15. *For any strongly connected synchronizing automaton with n states, for any $1 \leq s \leq n/2$,*

$$T_3 \leq \max\{n(s+2)/2, (n(n+4) - (2s-1)(2s+3) + 1)/4\} \tag{2.10}$$

Proof. If

$$k(n) < \frac{1}{n-s},$$

we apply Lemma 2.3.14.

Otherwise, from Theorem 2.3.11, the values that can be taken by the synchronizing probability function for t between n and T_3 are of the shape

$$k(t) = \frac{2}{n+r},$$

with $0 \leq r \leq n-2s$, and if $k(t) = 2/(n+r)$, then $k(t + \lfloor (n-r)/2 \rfloor + 1) > k(t)$.

Summing over all possible values for $k(t)$, one gets

$$\begin{aligned}
\sum_{r=0}^{n-2s} \lfloor (n-r)/2 \rfloor + 1 &= \sum_{r=2s}^{n} \lfloor (r)/2 \rfloor + 1 \\
&= \frac{(n)(n+4)}{4} - \frac{n \bmod 2}{4} - \frac{(2s-1)(2s+3)}{4} + \frac{1}{4} \\
&\leq (n(n+4) - (2s-1)(2s+3) + 1)/4.
\end{aligned}$$

□

This inequality on T_3 holds for all s. Therefore, minimizing the bound in (2.10) over s, we obtain:

Theorem 2.3.16. *In a strongly connected synchronizing automaton Σ with n states,*

$$T_3 \leq \frac{1}{8}\left(\sqrt{5n^2+4n-12}-n+6\right)n.$$

Proof. In Equation (2.10), $n(s+2)/2$ is an increasing function of s, while $(n(n+4)-(2s-1)(2s+3)+1)$ is a decreasing function of s. Therefore, the minimum of (2.10) is reached at the intersection of the two functions. The value of s that minimizes the expression in (2.10) is the solution of

$$n(s+2)/2 = (n(n+4)-(2s-1)(2s+3)+1)/4.$$

This is equivalent to:

$$s^2 + s(n+2)/2 + (1-n^2/4) = 0,$$

which has the solution

$$s = (\sqrt{5n^2+4n-12}-(n+2))/4.$$

This solution is positive and lower than $n/2$ as long as $n > 3$. Plugging this value into Equation (2.10) we obtain

$$T_3 \leq n(\sqrt{5n^2+4n-12}-n+6)/8.$$

□

Therefore, we have that $T_3 < n^2/(6.4...)$ for n sufficiently large. This bound strictly improves on our previous one (2.9).

2.4 The Growth of the SPF

In this section, motivated by Conjecture 2.3.3, we analyze the growth of the SPF. We first focus on the bound obtained in Theorem 2.3.11, which is obtained by analysing the growth of the SPF if $t < T_3$ based on the structure of the

Linear Program (2.3). We show that this bound is tight if we consider only the constraints of Program (2.3). Second, we provide a counterexample to Conjecture 2.3.3, which is also a counterexample to Conjecture 2.3.4.

2.4.1 Tight Bound on the Relaxed SPF Growth Before T_3

The question we tackle here was asked by Jungers [Jun14] as a research question in order to better understand the SPF:

Problem 2.4.1. *We consider the following process. Starting from $A(0)$ equal to the $n \times n$ identity matrix, $A(t+1)$ is obtained by adding binary columns of weight two to $A(t)$ with the following constraint:*

- *either the solution $k(t+1)$ and $k(t)$ of Program (2.3) obtained with $A(t+1)$ and $A(t)$ respectively are such that $k(t+1) > k(t)$*
- *or the condition of Lemma 2.2.11 is satisfied.*

If for some $A(t)$ there does not exist such matrix $A(t+1)$, then the process stops, and t is called the length of the process.

Theorem 2.3.12 provides the upper bound

$$\frac{n(n+4)}{4} - \frac{n \bmod 2}{4} - 1.$$

Is it possible to improve this bound?

Improving this bound would lead to a improvement of Theorem 2.3.12. We notice that this problem does not take into account the fact that $A(t)$ was initially defined from an automaton. We answer Problem 2.4.1 by providing a construction reaching this bound, therefore answering the question negatively.

As in the automaton setting, let us associate with $A(t)$ a graph $G(t)$ with n vertices such that the subset of columns of weight two of $A(t)$ is the incidence matrix of $G(t)$.

In terms of the graph $G(t)$, Program (2.3) is the following. There is a set of n nodes on a graph. Each of these nodes is associated to a value p_i such that $p_1 + p_2 + ... + p_n = 1$. We have to define a value k such that $0 \leq p_i \leq k$. Moreover, each column of weight 2 of $A(t)$ is an edge on this graph implying that the sum of the values of the connected nodes can't be higher than k. The aim is to minimize k.

In order to easier illustrate our construction, we consider the following problem:

$$max(V(t) = X_1 + X_2 + ... + X_n)$$

subject to $0 \leq X_i \leq 1$ and such that each column of weight 2 of $A(t)$ is an edge on this graph implying that the sum of the values of the connected nodes can't be higher than 1.

We notice that this problem is equivalent to the original one. Indeed, the solution $V(t) = 1/k(t)$ and $X_i = p_i/k(t)$ is optimal for any solution set $\{p_1, \ldots, p_k\}$ with objective function $k(t)$ of the original problem, and vice-versa (the whole problem has been divided by $k(t)$ in order to work with sums equal to 1 instead of $k(t)$).

We consider in $\mathbb{R}^n$ the set $P(t)$ of points achieving this value. We denote by $d(t)$ the dimension of this polyhedron.

Example 2.4.2. *Let us consider a graph on 5 states, with an edge connecting node 1 and node 2. This graph is represented in Fig. 2.8*

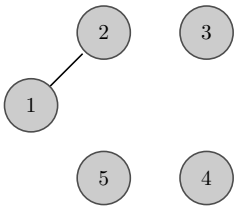


Figure 2.8: Graph G.

This situation corresponds to the the following Matrix A in the linear program (2.3):

$$A = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

The maximal value of the objective function is 4, and it is obtained with a weight of 1 on node 3, 4 and 5, and the sum of the weights of node 1 and node 2 equal to one. More precisely, the set of optimal solutions is:

$$P = \{X_1, X_2 = 1 - X_1, X_3 = 1, X_4 = 1, X_5 = 1\}$$

and the corresponding value is $V(3) = 4$*, or* $k(t) = 1/4$*. We have then that* $d(t) = 1$*.*

We now express Problem 2.4.1 in our setting. We start with a graph with n nodes that are all disjoint (therefore, initially, the maximisation has the optimal solution

$$X_1 = 1, X_2 = 1, \ldots, X_n = 1,$$

which is a zero dimension polyhedron in $\mathbb{R}^n$, and so $V(0) = n$ and $d(0) = 0$). At each step s, an edge is added to this graph with the constraint that either $V(s-1) > V(s)$, or $p(s-1) > p(s)$. This means that either the solution of the problem has decreased, or the dimension of the polyhedron of optimal solutions has decreased. If it is not possible to satisfy one of the two properties, the process stops.

The question is: how many such edges can we add at most?

Construction attaining the bound

In this section we provide a recursive construction satisfying the conditions and attaining the upper bound of Problem 2.4.1.

We consider a graph with $n > 3$ nodes. We notice that the construction adds recursively nodes to a strongly connected component. At each step, we give the two nodes that we connect with the new edge.

Initialisation For the first three nodes, we add the following edges in this order:

$$(1,2), (1,3), (2,3).$$

This gives us $V(3) = n - 1.5$, and 0 degrees of freedom.

Then, we add the fourth node and the edges in this order:

$$(1,4), (2,4), (3,4).$$

This gives us $V(6) = n - 2$, and 0 degrees of freedom.

First steps after the initialisation From now on, the process always keeps the same defined structure. Let us first enumerate the steps for the 5th, 6th and 7th nodes, before defining the generic process.

To add the 5th node, we add the following edges:

$$(1, 5), (2, 5).$$

This gives us $V(8) = n - 2.5$, and 0 degrees of freedom.

To add the 6th node, we add the following edges:

$$(1, 6), (3, 5), (4, 5), (2, 6).$$

This gives us $V(12) = n - 3$, and 0 degrees of freedom.

To add the 7th node, we add the following edges:

$$(1, 7), (3, 6), (2, 7).$$

This gives us $V(15) = n - 3.5$, and 0 degrees of freedom.

Recursive step The recursive step is different depending on the parity of the label of the node we add.

To add a node of the form $2m + 1$, we add the following edges:

$$(1, 2m + 1), (m, m + 3), (m - 1, m + 4), \ldots (2, 2m + 1).$$

This is a total of m steps, ending with 0 degrees of freedom and $V = n - (2m + 1)/2$.

To add a node of the form $2m$, we add the following edges:

$$(1, 2m), (m, m + 2), (m + 1, m + 2), (m - 1, m + 3), (m - 2, m + 4), \ldots, (2, 2m).$$

This is a total of $m+1$ steps, ending with 0 degrees of freedom and $V = n - m$.

We illustrate this process in Fig. 2.9, where we consider $n = 8$, and represented the edges, labelled by the order in which they are added. We can now claim our result:

Theorem 2.4.3. *The recursive construction presented attains the bound of Problem 2.4.1 and satisfies its constraints.*

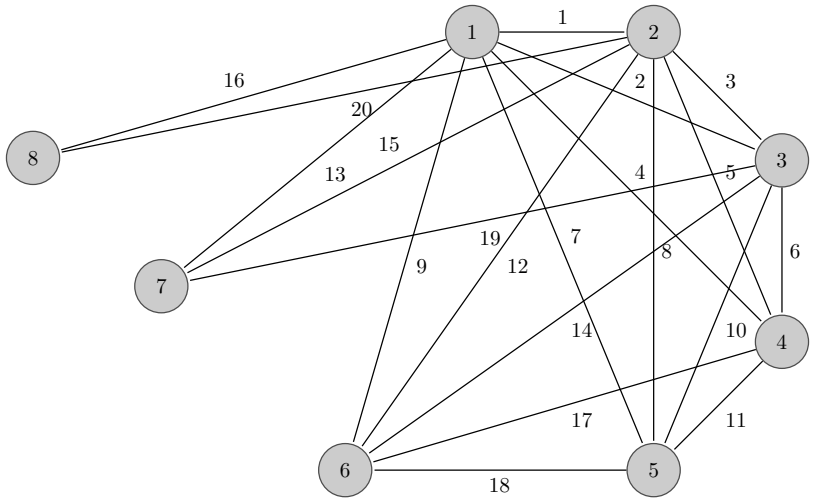


Figure 2.9: Maximal process for $n = 8$

Proof. We first show that the constraints of Problem 2.4.1 are satisfied. We proceed by recurrence, and in order to do so we first have to analyze explicitly the process up to $n = 4$.

For $n = 2$, we have one edge and keep one degree of freedom, but we can not add any other edge.

For $n = 3$, we first add the edge $(1, 3)$, and we have the solution $X_2 = X_3 = 1 - X_1$, which has one degree of freedom. Then we add the edge $(2, 3)$, and the only solution is $X_1 = X_2 = X_3 = 1/2$.

For $n = 4$, we first add the edge $(1, 4)$ and we have the solution $X_1 = 1 - X_4$, $X_2 = 1 - X_3$ with the constraint $X_1 < X_2 \leq 1/2$, which has two degrees of freedom. Then we add the edge $(2, 4)$, and the only solution becomes $X_1 = X_2 = 1 - X_3 = 1 - X_4$, which has one degree of freedom. Finally we add $(3, 4)$, and the only solution remaining is $X_1 = X_2 = X_3 = X_4$, with no degree of freedom.

(i) Before doing the recursive step, we notice that, in the described process, after adding a node n and all the edges of this step, the subgraph composed of states $1, \ldots, \lfloor n/2 \rfloor + 2$ is a clique, state n is connected to the first two states, state $n - 1$ is connected to the first three states, ... up to state $\lfloor n/2 \rfloor + 3$.

For any $n > 4$, we notice that the first edge added is $(1, n)$. After adding this edge, we can assign $X_n = 1 - \epsilon_n$ and $X_1 = \epsilon_n$, with ϵ_n a small value. In that setting, none of the edges connecting 1 to states $2, \ldots, n - 1$ represents

a constraint any more, since the value assigned to 1 is defined as very small. Therefore, states $2, \ldots, n-1$ can have their values assigned independently of X_1 and X_n.

Now, we continue the process of adding edge between nodes $2, \ldots, n-1$ up to when we don't have any degree of freedom on these nodes any more (where we have $X_2 = \cdots = X_{n-1} = 1/2$, while still having $X_n = 1 - \epsilon_n$ and $X_1 = \epsilon_n$). We notice that, when we don't consider state 1 and n, the states states $2, \ldots, n-1$ are connected between each other exactly in the same way as $1, \ldots, n-2$ were after the first step of the process that was done for $n-2$ (see (i) above). Therefore, it has exactly the same number of degrees of freedom, and we can add the edges in the same order as for this step, just by adding 1 on the indices.

Finally, we add the edge $(2, n)$, which forces that $X_1 = \cdots = X_n = 1/2$.

We notice that connecting a new vertex to a strongly connected graph where the only solution is $X_1 = \cdots = X_n = 1/2$ implies a decrease of the value assigned to this new vertex. Therefore in our process, when increasing n, the first step (adding edge $(1, n)$) satisfies the constraint on the decreasing value of $V(t)$.

Now, we show that we reach the bound of Problem 2.4.1. Let us call $S(n)$ the number of steps we are doing with this procedure when n nodes are in the strongly connected component. We have the following values:

- $S(1) = 0$
- $S(2) = 1$
- $S(3) = 3$
- $S(4) = 6$
- $S(5) = 8$
- $S(6) = 12$
- $S(7) = 15$
- $S(2m) = S(2m-1) + m + 1$
- $S(2m+1) = S(2m) + m$.

The two last terms of the regression gives us that

$$S(2m) = m(m+1)$$

and

$$S(2m+1) = m(m+1) + m = m(m+2).$$

And so, if n is even we have

$$S(n) = \frac{n(n+2)}{4}$$

and if n is odd we have

$$S(n) = \frac{(n-1)(n+3)}{4},$$

which is equal to the upper bound obtained in [GJ16]. □

The hypercube formulation

We notice that it is possible to formulate this problem as a geometrical problem in a n-dimensional space. The problem is the following: we consider an hypercube of side 1 on n dimensions, situated between 0 and 1 in each of its coordinates, in the positive orthant.

From this starting polyhedron, we consider the set of points of the polyhedron that are the farthest from the origin, in the sense of norm one.

At each step, we cut the polyhedron with the hyperplane defined by $X_{i_a} + X_{j_a} = 1$, with i_a and j_a that can be chosen. We then keep the part containing the origin and remove the other part of the polyhedron.

The hyperplane used can be chosen as we want, but it must fulfil the following condition: either all the points that are the farthest of the origin are removed, or the dimension of the polyhedron of farthest points has at least decreased. With this setting, we are searching for the maximal number of cuts that can be done, while satisfying the constraint.

2.4.2 Counterexample to a Conjecture on the SPF

We present an infinite family of automata which are counterexamples to Conjecture 2.3.3 and Conjecture 2.3.4. This family provides us with a lower bound on the maximum value of the triple rendezvous time for automata with n states for every odd integer $n \geq 9$. Let us name the automaton of the family with

n states $\mathscr{TR}_n$. The automaton $\mathscr{TR}_9$ is shown in Fig. 2.10. It has 9 states, labelled from q_1 to q_9, and two letters a and b defined as (zeros are replaced by dots):

$$a = \begin{pmatrix} . & . & . & . & . & . & 1 & . & . \\ . & . & . & 1 & . & . & . & . & . \\ . & . & 1 & . & . & . & . & . & . \\ . & 1 & . & . & . & . & . & . & . \\ . & . & 1 & . & . & . & . & . & . \\ . & . & . & . & . & . & . & 1 & . \\ 1 & . & . & . & . & . & . & . & . \\ . & . & . & . & . & 1 & . & . & . \\ . & . & . & . & . & . & . & . & 1 \end{pmatrix}, b = \begin{pmatrix} . & 1 & . & . & . & . & . & . & . \\ . & . & 1 & . & . & . & . & . & . \\ 1 & . & . & . & . & . & . & . & . \\ . & . & . & . & 1 & . & . & . & . \\ . & . & . & . & . & 1 & . & . & . \\ . & . & . & 1 & . & . & . & . & . \\ . & . & . & . & . & . & . & . & 1 \\ . & . & . & . & . & . & . & 1 & . \\ . & . & . & . & . & . & 1 & . & . \end{pmatrix}.$$

.

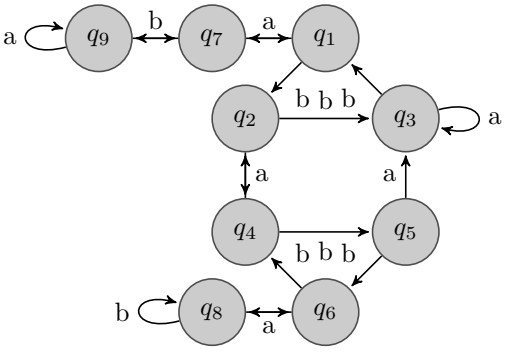


Figure 2.10: The automaton $\mathscr{TR}_9$.

With the parameter j in Conjecture 2.3.3 fixed to 2, this conjecture implies that for any synchronizing automaton with 9 states, $k(11) \geq 2/8$. Conjecture 2.3.3 is stronger than Conjecture 2.3.4. This latter implies, for the same automaton, that $T_3 \leq 11$. However, we have that $T_{3,\mathscr{TR}_9} = 12$, thus disproving both conjectures:

Proposition 2.4.4. *The T_3 of the automaton $\mathscr{TR}_9$ is equal to 12.*

Proof. On the one hand, the set of reachable vectors at $t = 11$, in which we separated the columns by blocks to show the evolution of $A(t)$ with respect to t, is equal to:

$$A_{tr_9}(11) = \left(\begin{array}{*{9}{c}|*{9}{c}|*{9}{c}|*{9}{c}|*{9}{c}|*{9}{c}|*{9}{c}|*{9}{c}|*{9}{c}} 1 & . & . & . & . & . & . & . & . & . & . & 1 & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\ . & 1 & . & . & . & . & . & . & . & . & 1 & . & . & . & . & . & . & 1 & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\ . & . & 1 & . & . & . & . & . & . & 1 & . & . & . & . & . & . & . & . & . & . & . & . & . & . & 1 & 1 & . & . & . & . & . & . \\ . & . & . & 1 & . & . & . & . & . & . & 1 & . & . & . & . & . & 1 & . & . & . & . & . & . & . & . & . & . & . & . & . & 1 \\ . & . & . & . & 1 & . & . & . & . & 1 & . & . & . & . & . & 1 & . & . & . & . & . & 1 & . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & 1 & . & . & . & . & 1 & . & . & 1 & . & . & . & 1 & . & . & . & . & . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & 1 & . & . & . & . & 1 & . & . & 1 & . & . & 1 & 1 & . & . & . & . & . & . & . & . & . & . & . & 1 \\ . & . & . & . & . & . & . & 1 & . & . & . & . & 1 & 1 & . & . & . & . & . & 1 & . & . & 1 & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & 1 & . & . & . & . & 1 & 1 & . & 1 & 1 & . & . & 1 & . & 1 & . & . & . & . & . & . & . & . \end{array}\right).$$

and contains only columns of weight one or two. On the other hand, the word *abbabbababba*, which is twelve letters long, maps states q_3, q_5 and q_9 on state q_3. □

Figure 2.11 represents the SPF of $\mathscr{TR}_9$, compared with the SPF of the automaton of the Černý family with 9 states. For $t = 11$, the SPF of $\mathscr{C}_9$ is larger than the SPF of $\mathscr{TR}_9$.

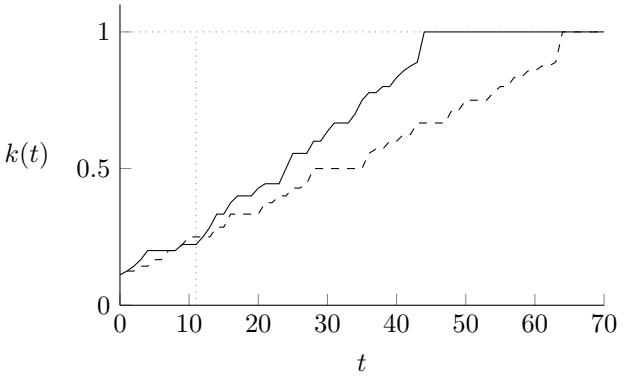


Figure 2.11: The synchronizing probability function of $\mathscr{TR}_9$ (solid curve), and of the automaton of the Černý family with 9 states (dashed curve). We have $k_{\mathscr{TR}_9}(11) = 2/9$.

We can extend $\mathscr{TR}_9$ to an infinite family of automata with an odd number of states. Starting from $\mathscr{TR}_n$, let l_1 be the letter such that $q_n l_1 = q_n$ and l_2 the other one. The automaton $\mathscr{TR}_{n+2}$ has $n+2$ states $q_1, \ldots, q_{n+2}$, the two letters l_1 and l_2, and the effects $q_n l_1 = q_{n+2}$, $q_{n+2} l_1 = q_n$, $q_{n-1} l_2 = q_{n+1}$, $q_{n+1} l_2 = q_{n-1}$, $q_{n+2} l_2 = q_{n+2}$ and $q_{n+1} l_1 = q_{n+1}$. All the others effects of the letters are the same as in $\mathscr{TR}_n$. Figure 2.12 show the automata $\mathscr{TR}_{11}$ and

$\mathscr{TR}_{13}$, and Fig. 2.13 shows $\mathscr{TR}_{2k+7}$, with k odd.

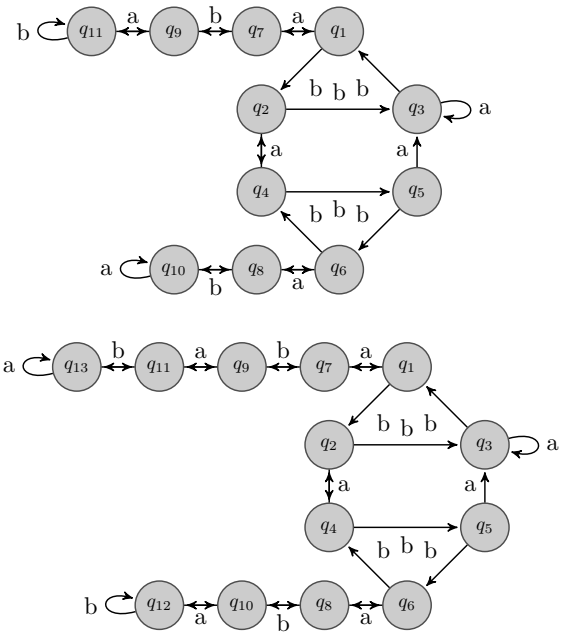


Figure 2.12: The automata $\mathscr{TR}_{11}$ and $\mathscr{TR}_{13}$.

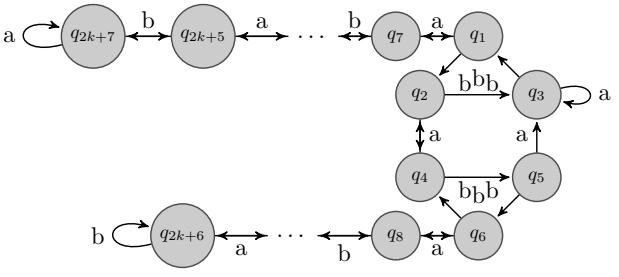


Figure 2.13: The automaton $\mathscr{TR}_{2k+7}$, with k odd.

We now prove the T_3 value for $\mathscr{TR}_{2k+7}$. In order to do so, we first provide with a set of lemmas. Let us denote by (q_i, q_j) the vector $(e_i+e_j)^T$. We analyze the evolution of the matrix $A(t)$ of $\mathscr{TR}_{2k+7}$ with respect to t. Notice that in order to obtain $A(t)$ from $A(t-1)$, we multiply the letters by the vectors of $A(t)$ from the right. We say that a vector c_{old} *can induce* a vector c_{new} if there

is a letter l such that $lc_{old} = c_{new}$. We say that a vector c_{new} of $A(t)$ *is induced* by a vector c_{old} of $A(t-1)$ at t if c_{new} is not in $A(t-1)$ and there is a letter l such that $lc_{old} = c_{new}$. As there are two letters in $\mathscr{TR}_n$, each vector can induce at most two vectors. We first notice that:

Lemma 2.4.5. *The reachable vectors of $A(t)$ with $t > 1$ that are induced by vectors of $A(t-1)$ at t, are induced by vectors of $A(t-1)$ that were themselves induced at $t-1$.*

Proof. A vector c_{new} which is in $A(t)$ and not in $A(t-1)$ is equal to lc_{old}, for a letter l and some vector c_{old} of $A(t-1)$. If c_{old} was in $A(t-2)$, then $lc_{old} = c_{new}$ would be in $A(t-1)$, therefore c_{old} was induced at $t-1$. □

Therefore, in order to analyze the evolution of $A(t)$ up to $t = T_3$, for each value of t, we only have to consider the columns induced at $t-1$.

For the ease of notation, we write (q_i, q_j) for the vector which has ones in position i and j and zeros in other positions. The structure of $\mathscr{TR}_n$ provides us with the following property for pairs of states of indices higher than 7:

Lemma 2.4.6. *The vectors that can induce (q_{2i}, q_{2j-1}), with $i, j > 3$, are also the only vectors that (q_{2i}, q_{2j-1}) can induce.*

Proof. For any vector $c = (q_{2i}, q_{2j-1})$, with $i, j > 3$, we have that $a^2c = b^2c = c$, and if for some column c', we have $ac' = c$ or $bc' = c$, then $a^2c' = b^2c' = c'$. Therefore a vector that can be induced by c can induce c' and vice versa. □

Corollary 2.4.7. *If a vector (q_{2i}, q_{2j-1}), with $i, j > 3$ is induced at t, then it can induce at most one vector at $t+1$.*

Proof. Any vector can induce at most two vectors. Since a vector (q_{2i}, q_{2j-1}), with $i, j > 3$ can only be induced by vectors that it can induce, one of the two possible vectors that it can induce must already be in $A(t)$. □

We can now prove the result:

Proposition 2.4.8. *The triple rendezvous time of $\mathscr{TR}_n$, with $n = 2k+7$ and $k \in \mathbb{N}$, is equal to $n+3$.*

Proof. We consider the evolution of $A(t)$ with respect to t.

The n first columns of $A(t)$ are the identity matrix for $t \geq 0$.

At $t = 1$, the column (q_3, q_5) is induced by e_3.

At $t = 2$, the column (q_2, q_4) is induced by (q_3, q_5).

At $t = 3$, the column (q_1, q_6) is induced by (q_2, q_4).

At $t = 4$, the column (q_7, q_8) is induced by (q_1, q_6).

The column (q_7, q_8) is such that Corollary 2.4.7 applies, and therefore induces only one column. For $4 \leq t \leq 2k + 4$, Corollary 2.4.7 applies at every step and the induced column is straightforward. At $t = 2k + 4$, the column (q_6, q_9) is induced.

At $t = 2k + 5$, the column (q_5, q_7) is induced by (q_6, q_9).

At $t = 2k + 6$, the column (q_4, q_9) is induced by (q_5, q_7).

At $t = 2k + 7$, the column (q_2, q_{11}) and the column (q_6, q_7) are induced by (q_4, q_9)[10].

At $t = 2k + 8$, the column (q_1, q_{13}) is induced by (q_2, q_{11}), the columns (q_1, q_8) and (q_5, q_9) are induced by (q_6, q_7).

At $t = 2k + 9$, the column (q_3, q_{10}) is induced by (q_1, q_8). The columns (q_7, q_{15}) and (q_3, q_{11}) are induced by (q_1, q_{13}), the column (q_4, q_7) is induced by (q_5, q_9).

At $t = 2k + 10$, the product of letter a with both the column (q_3, q_{11}) or the column (q_3, q_{10}) provides with a column of weight three.

Therefore, the T_3 is equal to $2k+10 = n+3$, which concludes the proof. □

The graph in Fig. 2.14 presents the synchronizing probability function of these automata with 9, 11 and 13 states.

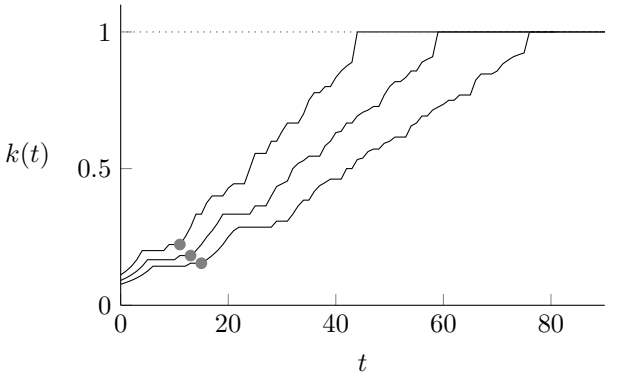


Figure 2.14: The synchronizing probability function of $\mathscr{TR}_9$, $\mathscr{TR}_{11}$ and $\mathscr{TR}_{13}$ (from the left to the right), and the points at which Conjecture 2.3.4 is not satisfied.

[10]Starting at $t = 2k+7$, the vectors induced are not the same for $\mathscr{TR}_9$, $\mathscr{TR}_{11}$ and $\mathscr{TR}_{13}$, as the vectors including states q_n or q_{n-1} do not induce vectors with higher indices. However it does not change the result.

The family $\mathscr{TR}_k$ was obtained through a random search on the space of automata composed of one permutation and a letter of rank $n-1$. Computing T_3 requires to compute the new columns generated in the matrix $A(t)$ at each time t. Since there are only a quadratic number of columns of weight two, computing T_3 requires a computational time of quadratic order, which is considered as fast. This supports the fact that the triple rendezvous time is an efficient indicator for testing large numbers of random automata.

2.5 Conclusion

In this Chapter, we first formalised the concept of triple rendezvous time as an intermediate step towards the Černý conjecture. Using the synchronizing probability function, a tool which allows one to represent the synchronization process within an automaton, we were able to prove a non-trivial upper bound on the triple rendezvous time.

Then, by providing an infinite family of automata for which $T_3 = n+3$ (with n being the number of states of the automaton), we refuted Conjecture 2.3.3, formulated in [Jun12]. Conjecture 2.3.3 was stated as a tentative roadmap towards a proof of the Černý conjecture with the help of the synchronizing probability function, and in that sense our conterexample is a negative result towards that direction.

A natural continuation of this research would be to find non-trivial bounds for T_l, with $3 < l \leq n$ (i.e., the smallest number such that the set of reachable vectors includes a column of weight at least l). Another research question is how to narrow the gap between $n+3$ and $n^2/(6.4...)$ for the triple rendezvous time.

Chapter 3

2-Transitivity of Permutation Sets and Square Graph of Automata

Abstract

In this chapter we study the diameter of the pair digraph of permutation automata. We first construct a n-state permutation automaton with diameter $\frac{n^2}{4} + o(n^2)$. Second, we show that the diameter of the square graph of a set of permutation is lower than $n(n-1)/2$. Third, we present an application of these results to the the joint spectral radius of matrices.[1]

3.1 Overview

In this chapter, we study the diameter of the square graph, also called *pair digraph*, of sets of permutations. Although synchronizing automata are not sets of permutations, as they must contain a letter of lower rank, they usually involve permutation letters. Understanding these permutations can already provide a lot of information on the automata itself, as we will see below.

Let A be a set of permutations from S_n. The pair digraph $P(A)$ consists of pairs $\{i, j\}$ as the set of vertices and the edges $(\{i, j\}, \{ia, ja\})$ for all i, j and $a \in A$. The *diameter* of $P(A)$, denoted $\operatorname{diam} P(A)$, is the maximum among

[1] Most results presented in this chapter have been published in [GGG+17].

the lengths of shortest (directed) paths between any two vertices. We study the behavior of diam $P(A)$ in terms of n. The problem comes from analysis of certain aspects of Markov chains and group theory [FJR$^+$98], but our interest in it is mainly motivated by its importance for the theory of synchronizing automata. Indeed, every synchronizing automaton $\mathscr{A}$ must have a letter a, say, whose action merges a pair of states. Thus, one can construct a reset word for $\mathscr{A}$ by successively moving pairs of states to a pair merged by a. If $\mathscr{A}$ possesses sufficiently many letters acting as permutations (as automata with the full transition monoid do), one can move pairs by these permutations, and hence, upper bounds on the diameter of the corresponding pair digraph induce upper bounds on $\mathrm{rt}(\mathscr{A})$.

In Section 3.2 we establish the lower bound $\frac{n^2}{4} + o(n^2)$ on the maximum of diam $P(A)$ by presenting a series of examples with two generators for every odd n. In Section 3.3, we show that a complete numbering of the pairs is impossible. This implies that diam $P(A) \leq \frac{n(n-1)}{2}$ for all $A \subseteq S_n$. In section 3.4 we provide an application of these results to the Joint Spectral Radius of sets of matrices.

Related Work. The diameters of groups and semigroups constitute a relatively well studied topic. A general discussion on diameters and growth rates of groups can be found in [Hel15]. Various results about the diameter of T_n and its submonoids are described in [Sal03, Pan15]. The length of the shortest representation of a constant[2] (including the case of partially defined transformations) is typically studied in the framework of synchronizing automata, see [Vol08, Vor14, AGV13].

3.2 Bounds on the Diameter of the Pair Digraph

In this section we present a lower bound on the largest diameter of the pair digraph $P(A)$ for $A \subseteq S_n$. We proceed by presenting subsets $A \subseteq S_n$ for every odd n such that the diameter of $P(A)$ is $\frac{n^2}{4} + o(n^2)$. In order to simplify the presentation, we mostly use automata terminology and describe the corresponding examples as a family of automata $\mathscr{F}_n = \langle Q_n, A, \delta \rangle$ (the input letters of $\mathscr{F}_n$ form the subset A). We let $Q_n = \{q_1, \ldots, q_n\}$ and denote pairs of states such as $\{q_i, q_j\}$ simply by $q_i q_j$.

[2] A constant is a transformation of rank 1.

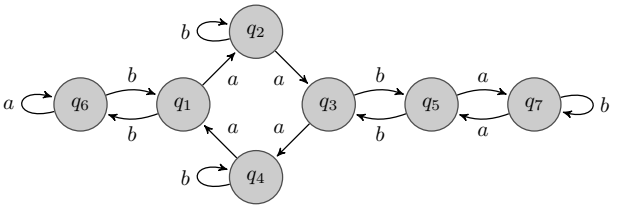


Figure 3.1: The automaton $\mathscr{F}_7$.

The automaton $\mathscr{F}_7$ shown in Fig. 3.1 is the first of the family $\mathscr{F}_n$. The digraph of pairs of its states is shown in Fig. 3.2. One can verify that the shortest word mapping q_2q_4 to q_4q_7 has length 15.

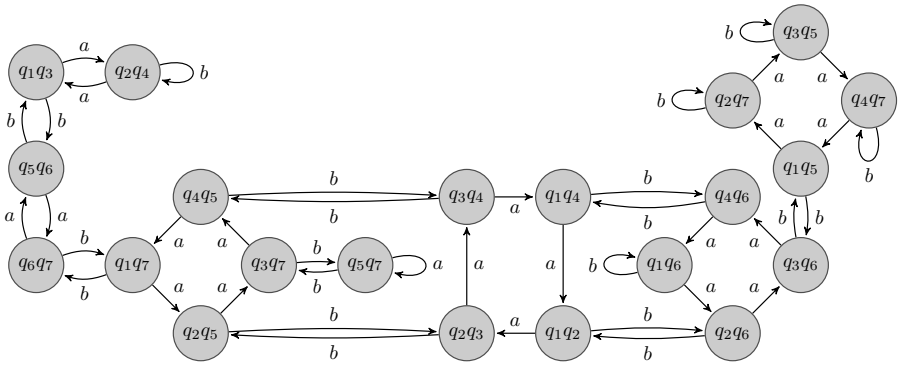


Figure 3.2: The pair digraph of $\mathscr{F}_7$.

The automata of the family are obtained recursively, starting with $\mathscr{F}_7$. From $\mathscr{F}_n$, we construct $\mathscr{F}_{n+2}$. The effect of the letters is the same for the states $q_1, \ldots, q_{n-2}$ in $\mathscr{F}_n$ and $\mathscr{F}_{n+2}$. The effect of the letters a and b at the states q_{n-1}, q_n, q_{n+1} and q_{n+2} is defined as follows: the letters mapping q_{n-1} and q_n to themselves in $\mathscr{F}_n$ exchange q_{n-1} with q_{n+1} and q_n with q_{n+2} respectively in $\mathscr{F}_{n+2}$. The other letter maps q_{n+1} and q_{n+2} to themselves and q_{n-1}, q_n to q_{n-3} and respectively q_{n-2}. The result is shown in Fig. 3.3 (for $n \equiv 3 \pmod 4$), in which k stands for $\frac{n-5}{2}$.

Theorem 3.2.1. *For odd $n \geq 7$, the diameter of the pair digraph of the automaton $\mathscr{F}_n$ is at least $\frac{n^2}{4} + o(n^2)$.*

Proof sketch. For the automaton $\mathscr{F}_n$ (with $n \equiv 3 \pmod 4$ and $n > 7$; for $n = 7$, the most distant pairs are q_2q_4 and q_4q_7, as shown in Fig. 3.2), we

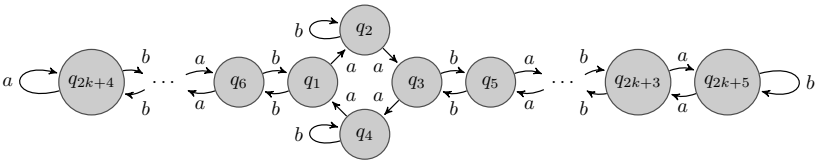


Figure 3.3: The automaton $\mathscr{F}_{2k+5}$, with k odd.

claim that any word mapping q_2q_4 to $q_{k+2}q_{k+4}$ with $k = \frac{n-5}{2}$ has length at least $\frac{n^2}{4} + \frac{5n}{4} - 7$. For this, we define a function N which associates a non-negative integer $N(q_iq_j)$ to each pair q_iq_j, $i < j$. This function is such that if a pair q_iq_j is mapped by a or b to a pair $q_{i'}q_{j'}$, then $N(q_{i'}q_{j'}) \geq N(q_iq_j) - 1$. This implies that if $(q_iq_j)\cdot w = q_sq_t$ for some word w, then the length of w is at least $N(q_iq_j) - N(q_sq_t)$. The number assigned to $q_{k+2}q_{k+4}$ is 0, while the number given to q_2q_4 is equal to $\frac{n^2}{4} + \frac{5n}{4} - 7$, thus, the claim holds.

In addition, we describe a word of length $\frac{n^2}{4} + \frac{5n}{4} - 7$ that maps q_2q_4 to $q_{k+2}q_{k+4}$. Therefore $\frac{n^2}{4} + \frac{5n}{4} - 7$ is the exact value of the "distance" between these two particular pairs.

A similar argument holds for $n \equiv 1 \pmod 4$, with the distance between two particular pairs of states being at least $\frac{n^2}{4} + \frac{5n}{4} - 7.5$.

Proof. Recall that we aim to define a function N which associates an integer $N(q_iq_j) \geq 0$ to each pair q_iq_j, $i < j$, and has the following property:

$$\text{if } q_iq_j \text{ is mapped by } a \text{ or } b \text{ to a pair } q_{i'}q_{j'}, \text{ then } N(q_{i'}q_{j'}) \geq N(q_iq_j) - 1. \tag{3.1}$$

For an illustration, see Fig. 3.4 which presents the pair digraph of the automaton $\mathscr{F}_7$ with the values of the corresponding function N shown at each vertex.

For $n > 7$, $n \equiv 3 \pmod 4$, the values of the function N are provided in Appendix A.1. It can be routinely verified that the function N defined this way satisfies (3.1). Therefore the shortest word mapping q_2q_4 to $q_{k+2}q_{k+4}$ is at least $N(q_2q_4) - N(q_{k+2}q_{k+4})$ letters long. Unfolding the definition, we obtain

$$N(q_{k+2}q_{k+4}) = 0$$

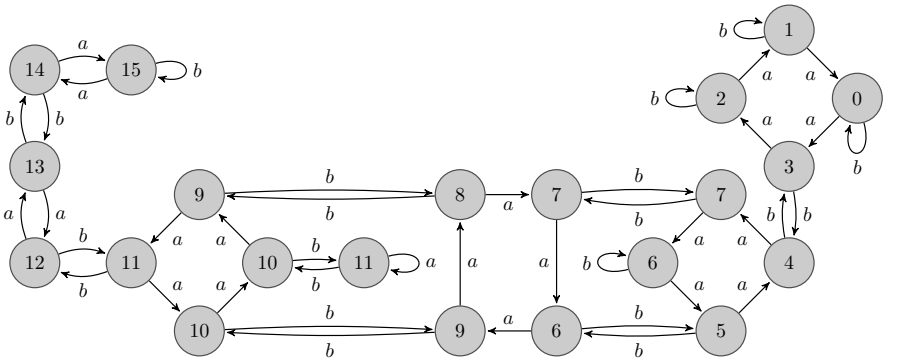


Figure 3.4: The pair digraph of $\mathscr{F}_7$, with function values.

and

$$N(q_2q_4) = N_1 + N_2 + 4k + 8 = \frac{k+3}{2} + (k+4)(k-1) + 4k + 8$$
$$= k^2 + \frac{15k}{2} + \frac{11}{2} = \frac{n^2}{4} + \frac{5n}{4} - 7.$$

In addition, Table 3.2 provides the construction for a word of length $\frac{n^2}{4} + \frac{5n}{4} - 7$ which maps q_2q_4 to $q_{k+2}q_{k+4}$. The word is composed of several factors being labels of a certain segments of the directed path from q_2q_4 to $q_{k+2}q_{k+4}$ in the pair digraph of the automaton $\mathscr{F}_n$. For each segment we give its start and end pairs as well as its label and length. There are 4 "starting" factors of total length $4k + 8$, followed by $2(k-1)$ "inner" factors, forming $\frac{k-1}{2}$ words of total length $2k + 8$ each, and 2 "finishing" factors of total length $\frac{k+3}{2}$. Altogether they indeed form a word of length

$$4k + 8 + k^2 + 3k - 4 + \frac{k+3}{2} = k^2 + \frac{15k}{2} + \frac{11}{2} = \frac{n^2}{4} + \frac{5n}{4} - 7.$$

□

Our numerical experiments confirm that $\frac{n^2}{4} + \frac{5n}{4} - 7$ is indeed the diameter of the pair graph of the automaton $\mathscr{F}_n$ for $n \equiv 3 \pmod 4$ from $n = 11$ to $n = 31$ while $\frac{n^2}{4} + \frac{5n}{4} - 7.5$ is the exact value of the diameter for $n \equiv 1 \pmod 4$ from $n = 13$ to $n = 29$.

We have computed the largest diameter of the pair digraph $P(A)$ for all $A \subseteq S_n$ with $|A| = 2$ and $n = 5, 7, 9$ and performed a number of random sampling

Start pair	End pair	Factor	Length of the factor	Sum of lengths of factors within the group
q_2q_4	q_1q_3	a	1	
q_1q_3	q_1q_7	$(ba)^kb$	$2k+1$	
q_1q_7	q_3q_8	aba^3ba	7	
q_3q_8	q_1q_{11}	$(ba)^{k-1}b$	$2k-1$	
				$4k+8$
q_1q_{11}	q_1q_5	a^3ba	5	
q_1q_5	q_3q_6	b	1	
q_3q_6	q_3q_{12}	a^3ba	5	
q_3q_{12}	q_1q_{15}	$(ba)^{k-2}b$	$2k-3$	
q_1q_{15}	q_1q_9	a^3ba	5	
q_1q_9	q_3q_{10}	bab	3	
q_3q_{10}	q_3q_{16}	a^3ba	5	
q_3q_{16}	q_1q_{19}	$(ba)^{k-3}b$	$2k-5$	
$\dots$	$\dots$	$\dots$	$\dots$	
q_1q_{2k+1}	q_1q_{2k-5}	a^3ba	5	
q_1q_{2k-5}	q_3q_{2k-4}	$(ba)^{\frac{k-5}{2}}b$	$k-4$	
q_3q_{2k-4}	q_3q_{2k+2}	a^3ba	5	
q_3q_{2k+2}	q_1q_{2k+5}	$(ba)^{\frac{k+1}{2}}b$	$k+2$	
q_1q_{2k+5}	q_1q_{2k-1}	a^3ba	5	
q_1q_{2k-1}	q_3q_{2k}	$(ba)^{\frac{k-3}{2}}b$	$k-2$	
q_3q_{2k}	q_3q_{2k+4}	a^3ba	5	
q_3q_{2k+4}	q_1q_{2k+3}	$(ba)^{\frac{k-1}{2}}b$	k	
				$(2k+8)\frac{k-1}{2}$ $= k^2+3k-4$
q_1q_{2k+3}	q_3q_{2k+3}	a^2	2	
q_3q_{2k+3}	$q_{k+2}q_{k+4}$	$(ba)^{\frac{k-3}{4}}b$	$\frac{k-1}{2}$	
				$\frac{k+3}{2}$

Table 3.2: Construction of a word bringing q_2q_4 to $q_{k+2}q_{k+4}$.

experiments with two permutations for larger values of n. The experimental results suggest that the pair digraph of the automaton $\mathscr{F}_n$ has the largest diameter among all possible pair digraphs. Regarding the size of the alphabet, we notice that any letter added to the automaton creates a large number of edges in the square graph. More precisely, if a letter has an impact on a state,

then it has an impact on all the pairs including this state in the square graph. From our experiments, these additional edges create shortcuts in the square graph, lowering its diameter. Therefore, we believe that the largest diameter is obtained with two letters. Moreover, structure analysis as performed in Appendix A.1 shows that, if the square graph is strongly connected, multiple disjoint paths from a pair to an other are possible. This implies that the longest path between two pairs can at most include half of the pairs. Thus, we formulate the following conjecture:

Conjecture 3.2.2. *The diameter of the pair digraph for a subset of S_n is bounded above by $\frac{n^2}{4} + o(n^2)$.*

3.3 The Pair Numbering Method

In this section, we analyze the structure of the square graph based on the knowledge of the structure of the base graph. Using this method, we first show that the diameter of the square graph of a 2-transitive set of permutations on more than 12 states can never reach the bound $n(n-1)/2-1$. Second, we show that only a restricted class of automata could reach this bound.

The notion that guides our approach is the *complete numbering* of a square graph. It is defined as follows.

Definition 3.3.1. *Let $\mathscr{A} = (Q; \Sigma; \delta)$ be an n-state automaton. We call complete numbering N of $SG(\mathscr{A})$ a function $N : Q^2 \to \mathbb{N}$ associating an integer to each pair with the following conditions:*

1. *if $q_i, q_j \neq q_k, q_l$, then $N(q_iq_j) \neq N(q_kq_l)$*
2. *if there is a letter $a \in \Sigma$ such that $\{q_i, q_j\}a = \{q_k, q_l\}$, for any $i \neq j$, $k \neq l$ then $N(q_iq_j) \leq N(q_kq_l) + 1$,*
3. *if there is a letter $a \in \Sigma$ such that $q_ia = q_ja$ with $q_i \neq q_j$, then $N(q_iq_j) = 1$.*

In the case of a set of permutations, a complete numbering only has to satisfy the two first conditions. For a strongly connected square graph, a complete numbering is possible if and only if its diameter is equal to $n(n-1)/2-1$. Indeed a path from a pair with the maximal number to a pair with the minimal number cannot be shorter than $n(n-1)/2-1$, as each pair in the path can only decrease the value by one.

Our analysis is based on the following procedure. We first assume a pattern in the graph of $\mathscr{A}$ (which we call *base graph*). From this pattern, we deduce the pattern in the square graph $SG(\mathscr{A})$ and show that this pattern implies that *a complete numbering* is impossible. To illustrate the method, Example 3.3.2 shows a situation where a complete numbering is impossible.

Example 3.3.2. *Let us consider the two permutations in Fig. 3.5, the square graph is shown in Fig. 3.6. We observe that the longest shortest path between two pairs is of length 4 (between q_1q_2 and q_3q_4). This implies that a complete numbering is impossible, since with a complete numbering the diameter of the square graph would be equal to 5.*

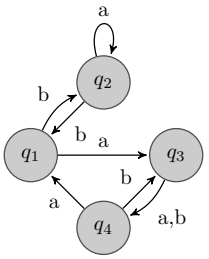


Figure 3.5: Two permutations.

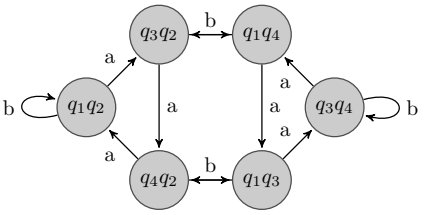


Figure 3.6: The square graph of the permutations in Fig. 3.5.

In Subsection 3.3.1, we consider complete numbering for sets of permutations. In Subsection 3.3.2 we consider complete numbering for synchronizing automata.

3.3.1 Strongly Connected Permutations

In this section, we consider a set of permutations such that both the base graph and the square graph are strongly connected.

In order to prove that a complete numbering is impossible, we consider the largest cycle identified by one single letter in the base graph. We define a cycle as follows:

Definition 3.3.3. *We call a* cycle *C of length l a set (V, E) of vertices $v_1, \ldots, v_l$ and oriented edges $e_1, \ldots, e_l$, such that $v_1 e_1 = v_2$, $\ldots$,$v_l e_l = v_1$. We say that a vertex v_i is* preceding *a vertex v_j in the cycle, and that v_j is* following *v_i, if $v_i e_i = v_j$, with $v_i, v_j \in V$ and $e_i \in E$.*

From now on, we use C and D to designate cycles of single states in the base graph, and use P_1 and P_2 to designate cycles of pairs in the square graph.

Since we assume that the base graph is strongly connected, there always exists a cycle of highest length labelled by a single letter. The proof that a complete numbering is not possible depends on the length of this cycle. The first subsection provides a set of general lemmas. The following subsections cover all the possible size of the largest cycle, and leads to the conclusion that in any case, a complete numbering is impossible.

Key lemmas

We provide here the main lemmas that we use in this section. The first one is about cycles in the square graph, and is illustrated in Example 3.3.5.

Lemma 3.3.4. *If a complete numbering of the square graph is possible, then in any cycle of the square graph the number associated to consecutive pairs is decreasing by one unit, except for one single increase of magnitude equal to the size of the cycle minus one.*

Proof. Since we consider a complete numbering, there should be k different numbers, with k the number of pairs in the cycle. Moreover each pair is reachable from any other pair by a path of length at most $k - 1$, so the numbers must be consecutive. Then, starting from the pair with highest number, the only possible numbering for the next pair is this number minus one. The next must be this number minus two, etc. Continuing this process up to the end of the cycle gives the result. □

We call the numbers of the pairs in the cycle the *range* of this cycle, and we call a *jump* the transition between a lowest and a highest number in a cycle.

Example 3.3.5. *Figure 3.7 shows a complete numbering of a cycle in the square graph.*

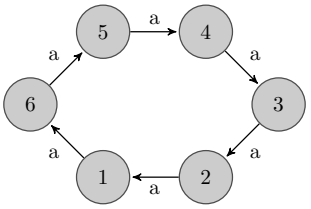


Figure 3.7: Example of complete numbering of a cycle.

This implies that, in a complete numbering, each pair in a cycle of the square graph has either the highest number of the cycle, the lowest number of the cycle, or a number N such that $N + 1$ and $N - 1$ are attributed to pairs preceding it and following it in the cycle. In this latter case, we call this pair a *middle pair* of the numbering. With these concepts, we can now consider overlapping cycles in the square graph. This situation is illustrated in Example 3.3.7.

Lemma 3.3.6. *If a complete numbering of the square graph is possible, if P_1 and P_2 are two cycles of the square graph such that some pairs are in P_1 and not in P_2, some pairs are in P_2 and not in P_1, and some pairs are in both, then the numbers of the pairs which are in both cycle are the lowest numbers of one of the cycles, and the highest numbers of the other cycle.*

Proof. Since a cycle has all the numbers in some range exactly once, and since the numbers of the pairs that are only in one of the cycles cannot be equal, one of the cycles has a higher range than the other, and the common pairs have the numbers that are in both ranges. □

Example 3.3.7. *Figure 3.8 gives an example of complete numbering of overlapping cycles in the square graph. Letter a labels a cycle of length 5 that has values* 6, 5, 4, 3 *and* 2. *Letter b labels a cycle of length 3 that has values* 3, 2 *and* 1. *The pairs with values* 3 *and* 2 *are common to both cycles. Therefore in any complete numbering the numbers of these two common pairs are the highest of one cycle and the lowest of the other cycle.*

This implies that two cycles in the square graph can not have multiple intersections:

Corollary 3.3.8. *If a complete numbering of the square graph is possible, then if two cycles of the square graph have some common pairs, all these pairs are consecutive in both cycles.*

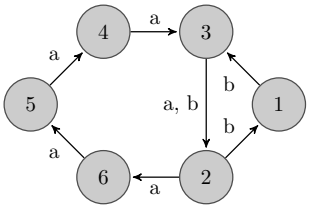


Figure 3.8: Example of complete numbering of overlapping cycles.

We can now state a simple lemma which allows us to discard many possible patterns:

Lemma 3.3.9. *If a complete numbering of the square graph is possible, then if a cycle of the square graph is labelled by a single letter, there cannot be three outgoing edges from pairs of this cycle labelled by an other single letter.*

Proof. Let us call the cycle P_1 and the second letter b. We consider the cycles that letter b labels in the square graph. If the letter b labels three distinct cycles, each starting from one of the pair that are mapped outside of P_1, then one of these cycles has common states with P_1 which are not the highest numbers or the lowest numbers of that cycle, and Lemma 3.3.6 implies that no complete numbering is possible.

Otherwise two of the outgoing edges are in the same cycle labelled by b. Therefore this cycle has a common pair with P_1, then some pairs not in C, then again some states in P_1, and then again some states not in P_1. In that case, we are in the situation of Corollary 3.3.8, and a complete numbering is impossible. □

Lemma 3.3.10. *If a complete numbering of the square graph is possible and if a cycle P_1 of the square graph is labelled by a single letter a and has exactly two outgoing edges labelled b from pairs not following each other in P_1, then one of them has the lowest number in P_1 and the other one is a middle pair. Let us call the the middle pair p_i and p_j the other one, then the pairs $p_j a$, ..., p_i are in the cycle $p_i b^k$, $k \in \mathbb{N}$ in the same order as in P_1.*

Proof. We first notice that only one of the two outgoing edges labelled by b can be a jump, otherwise a complete numbering would be impossible. Therefore, the other outgoing edge has to be a decrease by 1 in the cycle labelled by b, and can only start from the pair with lowest number in P_1. Since the lowest and highest number in P_1 are adjacent, the other pair has to be a middle pair.

As p_i is a middle pair, we must have that $N(p_ib) > N(p_i)$, and therefore p_i has the lowest number in the cycle p_ib^k. Therefore, as a complete numbering is possible, we must have that $p_ia^{-1} = p_ib^{-1}$, because both have number equal to $N(p_i) + 1$. Moreover, for each element p_ia^{-k} such that $N(p_ia^{-k}) > N(p_i)$, we must have that $p_ia^{-k} = p_ib^{-k}$. Therefore, this stops when $N(p_ia^{-k}) < N(p_i)$, ie. when we reached the lowest number in the cycle, which must be p_j. This means that p_ja is the first common pair of both cycle P_1 and cycle p_ib^k. □

We now have a last general lemma applying on the full partition of pairs:

Lemma 3.3.11. *If a complete numbering of the square graph is possible, then if there is a partition of all the pairs into cycles labelled by a single letter, one of these cycles has less than two outgoing edges labelled by an other letter.*

Proof. Indeed, to obtain a complete numbering, each cycle must have a range of numbers. Therefore, there must be an ordering in the ranges of numbers of the cycles. In a complete numbering, the cycle with the highest range cannot have two different states mapped to states in an other range, otherwise there is a decrease of at least two. Therefore, if all cycles have at least two pairs mapped on pairs of other cycles, none can be the one with the highest range in a complete numbering. If all the pairs are in that partition, then a complete numbering of the square graph is impossible. □

With these tools in hand, we can now analyze the possible cases. We start with a largest cycle including all the states and the case of a largest cycle larger or equal to five. We then analyze the cases of letters labelling only cycles of lower size. As the proofs need a detailed systematic analysis of particular cases, they have been placed in Appendix A.2. The summary of the analysis is presented in Table 3.3. The main argument used for each case and the reference to the corresponding proof is given. Notice that the table is ordered in decreasing order of size of cycles, while the appendix follows the proof order, because some situations need to be excluded before analyzing other situations.[3]

Each result in Table 3.3 is stated as a proposition in Appendix A.2. Combining these propositions, we can formulate the general result:

Theorem 3.3.12. *Let consider a set of permutations on $n > 12$ elements. Then, either the pair digraph is disconnected, or its diameter is strictly smaller than $n(n-1)/2 - 1$.*

[3]For example, the case with both cycles of length 4 and self-loops needs to be excluded before analyzing the case of cycles of length 4 and 3.

Cycle size	Main argument	Proof
Cycle of length n	Lemma 3.3.11	A.2.1.
Cycle of length larger than or equal to 5	Lemma 3.3.9 and Lemma 3.3.10	A.2.2
Both cycles of length 4 and 3	Lemma 3.3.9	A.2.4
Both cycles of length 4 and swaps	Structure analysis	A.2.5
Both cycles of length 4 and self-loops	Lemma 3.3.9	A.2.3
Only cycles of length 4	Structure analysis	A.2.7
Both cycles of length 3 and swaps	Lemma 3.3.9	A.2.6
Both cycles of length 3 and self-loops	Lemma 3.3.9	A.2.3
Only cycles of length 3	Structure analysis	A.2.8
Only swaps and self-loops	Lemma 3.3.9	A.2.9

Table 3.3: Possible cycle situations and the proof reference.

3.3.2 Application to Synchronizing Automata

In this section, we show how the pair numbering method can be applied to synchronizing automata. We notice that if it is possible to prove that a complete numbering is impossible for an automaton, then the last step of the compression method is shorter than $n(n-1)/2$. Proving that it is true for all synchronizing automata for which Conjecture 1.2.2 is not proven would improve the bounds obtained by Pin, Szykuła and Shitov.

First of all, we notice that Conjecture 1.2.2 can be restricted to strongly connected automata, and is proven for circular automata [Dub98]. We notice in Example 3.3.13 that, unlike for permutations, a complete numbering of the square graph of synchronizing automata is not impossible in general.

Example 3.3.13. *Figure 3.9 shows the automaton $\mathscr{C}_6$, and Fig. 3.10 shows $SG(\mathscr{C}_6)$. We observe that in this case, a complete numbering of the square graph is possible, as shown in Fig. 3.11.*

The two following lemmas allow one to restrict the set of automata in which a complete numbering is possible.

Lemma 3.3.14. *Let $\mathscr{A} = (Q; \Sigma; \delta)$ be an n-state synchronizing automaton. If $|Qa| \leq n-2$ for some $a \in \Sigma$, then a complete numbering of $SG(\mathscr{A})$ is impossible.*

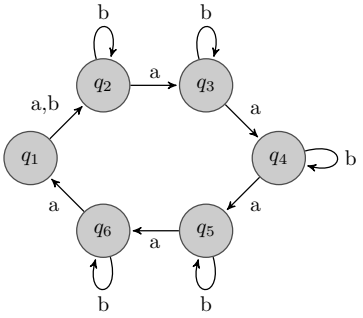


Figure 3.9: The automaton $\mathscr{C}_6$.

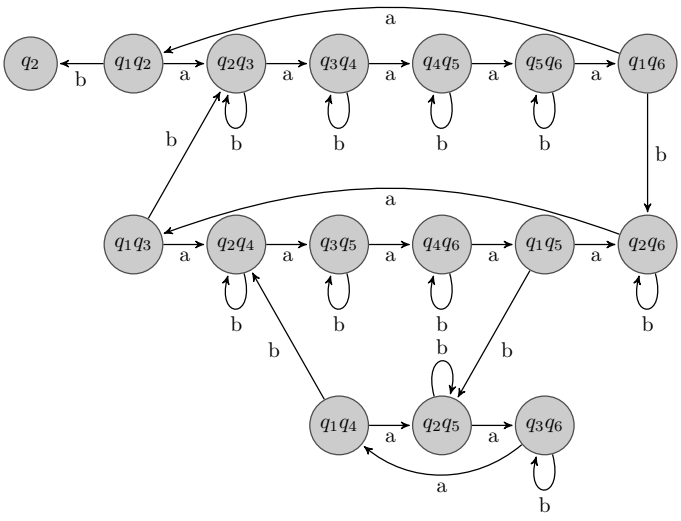


Figure 3.10: The square graph $SG(\mathscr{C}_6)$.

Proof. We have that Qa contains at most $n-2$ states, therefore at least two pairs are synchronized (or one triple). Therefore at least two pairs have number one. □

Lemma 3.3.15. *Let $\mathscr{A} = (Q; \Sigma; \delta)$ be an n-state synchronizing automaton. If $a, b \in \Sigma$ are synchronizing two different pairs, then a complete numbering of $SG(\mathscr{A})$ is impossible.*

Proof. The pair synchronized by a and the pair synchronized by b must both

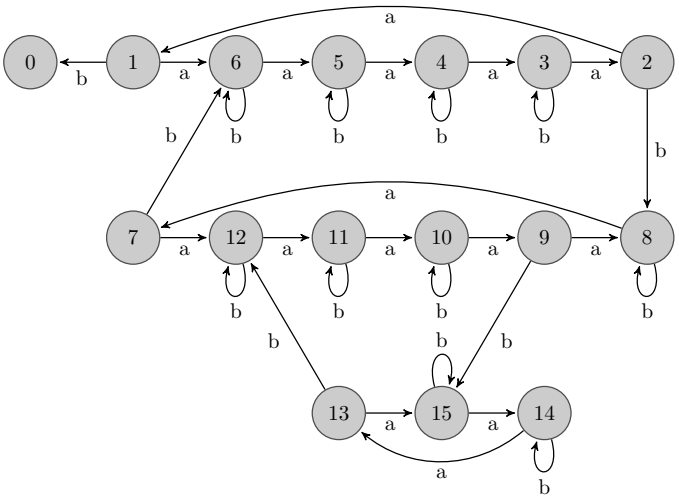


Figure 3.11: A complete numbering of the square graph $SG(\mathscr{C}_6)$.

have number one. □

Now, the only cases for which we have not proven that a complete numbering is impossible are automata composed of permutation letters and letters of rank $n-1$ that all synchronize the same pair. Some of the proofs used in Section 3.3.1 can be directly adapted to this setting. However, the assumption that some states had a preimage, which is not always the case in sychronizing automata, is often a key part of the proofs, and therefore we have not yet completely transposed the result for synchronizing automata. We leave this as an open question:

Problem 3.3.16. *Let $\mathscr{A} = (Q; \Sigma; \delta)$ be an n-state synchronizing automaton. Let all letters $l \in \Sigma$ be either non-circular permutations letters or letters of rank $n-1$ synchronizing the same pair. Is a complete numbering of $SG(\mathscr{A})$ possible?*

Answering negatively to this question would improve the last step of the compression method, and improve the general bound on the reset threshold for synchronizing automata.

3.4 An Application to the Joint Spectral Radius

The *joint spectral radius*(JSR) of a set of matrices is the highest growth rate of products of these matrices. For linear swaping systems in discrete time, it characterizes the maximal asymptotic growth rate of the system. The book [Jun09] provides a deep analysis of the JSR properties.

The formal definition of the JSR is the following. For a set Σ of $\mathbb{R}^{n\times n}$ matrices, a submultiplicative matrix norm $\|.\|$, and $t \in \mathbb{N}$ the quantity $\hat{\rho}_t(\Sigma)$ is defined as:

$$\hat{\rho}_t(\Sigma) = \sup\{\|A\|^{1/t} : A \in \Sigma^t\}.$$

The joint spectral radius is then defined as its limit towards infinity:

$$\rho(\Sigma) = \lim_{t\to\infty}\hat{\rho}_t(\Sigma, \|.\|).$$

We notice that the submultiplicativity of the norm implies that the JSR of a set is lower than or equal to the highest norm of its elements.

Example 3.4.1. *Let us consider the following set of matrices:*

$$\Sigma = \left\{ \begin{pmatrix} 0 & 2 \\ 0 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 0 \\ 2 & 0 \end{pmatrix} \right\}.$$

Both matrices have a spectral radius equal to zero. However by multiplying these matrix we obtain the matrix

$$A = \begin{pmatrix} 4 & 0 \\ 0 & 0 \end{pmatrix},$$

which has a spectral radius of 4. Therefore we have that

$$\rho(\Sigma) \geq \sqrt{\rho(A)} = 2.$$

Since the norm of the two matrices in Σ is also equal to 2 and we consider submultiplicative norm, the JSR can not be greater than 2, and we therefore have $\rho(\Sigma) = 2$.

The problem on which we can apply our results comes from the following statement:

Corollary 3.4.2 (Corollary 3.2 in [Jun09])**.** *Let Σ be a finite set of nonnegative integer matrices of dimension n. If $\rho(\Sigma) > 1$, then $\rho(\Sigma) \geq 2^{1/n^2}$.*

This corollary comes from the following one:

Corollary 3.4.3. *For any finite set of nonnegative integer matrices Σ, we have $\rho(\Sigma) > 1$ if and only if there is a product $A \in \Sigma^*$ such that $A_{i,i} \geq 2$ for some i.*

As explained in [Jun09], and entry $A_{i,i} \geq 2$ can only be obtained in two different ways: either there is a cycle containing an edge of weight two in the graph of the set, or there is a cycle in the graph of pairs containing a node (i, i) and a node (p, q).

The first case implies a cycle of length at most n, and therefore a joint spectral radius higher than or equal to $2^{1/n}$. The second case implies a cycle of length at most n^2 in the square graph, and therefore a joint spectral radius higher than or equal to $2^{1/n^2}$. In this section, we first prove a tight bound of $\rho(\Sigma) \geq 2^{2/n(n+1)}$. Second, we restrict ourselves to the case of three matrices, and we build a set with joint spectral radius $\rho(\Sigma) = 2^{4/n^2}$, which is so far the lowest example for sets with a bounded number of matrices.

3.4.1 General Case

First of all, we notice that Corolary 3.4.2 can be refined to

Theorem 3.4.4. *Let Σ be a finite set of nonnegative integer matrices of dimension n. If $\rho(\Sigma) > 1$, then $\rho(\Sigma) \geq 2^{2/n(n+1)}$.*

Proof. Indeed, the demonstration of Corollary 3.4.2 provided in [Jun09] is based on the fact that a shortest path in the graph of pairs is at most n^2 long. However, the pairs don't have to be ordered. There are only $n(n-1)/2$ different unordered pairs of two different states, and we can add the n pairs composed of equal states. The longest possible cycle of weight two is therefore equal to $n(n-1)/2 + n$, i.e. the number of pairs plus the number of single states. □

Moreover, we notice that this bound is tight for any n. Indeed, we can build a set Σ composed of $n(n-1)/2 + 1$ matrices in the following way. All the non specified entries are zero:

$$\begin{cases} A_e & \text{sends } n \text{ to 1 and 2} \\ A_l & \text{sends } n-1 \text{ and } n \text{ to 1} \\ A_{i,j} & \text{with } 0 < i < j < n \text{ sends } i \text{ to } i \text{ and } j \text{ to } j+1 \\ A_{i,n} & \text{with } 0 < i < n-1 \text{ sends } i \text{ to } i+1 \text{ and } n \text{ to } i+2 \end{cases}$$

We notice that these matrices describe a unique path in the graph of pairs from q_1q_2 to q_2q_3 of length $n(n-1)/2$. For any weight distributed in the graph, the only matrix increasing the total weight is A_l, sending q_n to q_1 and q_2. The only matrix which sum weights in multiple states is A_l, which regroups the weights of q_{n-1} and q_n in state q_1. The shortest path from q_1 to q_n is of length $n-1$.

In order to obtain a path of weight two, starting with a weight 1 on state k, we first have to bring it to q_n, which takes $n-k-1$ steps. Then we have to use matrix A_e which double the weight, but put it into two different states. In any situation of a pair with equal weight, there is only one matrix which is not decreasing the total weight, and it is the one following the path in the square graph. After reaching q_1q_n in $n(n-1)/2-1$ steps, the matrix A_l allows one to get a weight of 2 on state q_1, which can then be send to state k in $k-1$ steps, leading to a length of $n(n-1)/2+n$.

Example 3.4.5. *For matrices of size three, we have the following set:*

$$\Sigma = \left\{ A_e = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 1 & 0 \end{pmatrix}, A_l = \begin{pmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix}, A_{1,2} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix}, A_{1,3} = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} \right\},$$

which is represented in Fig. 3.12. The shortest way to have a 2 appearing in

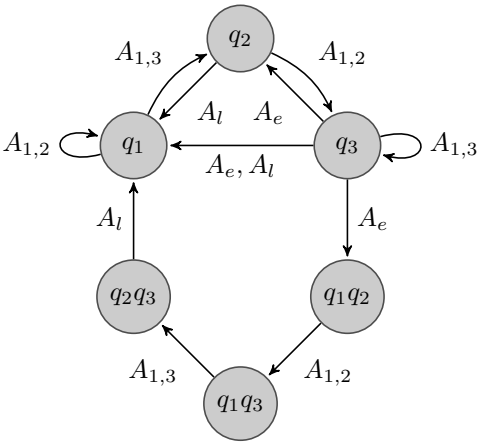


Figure 3.12: Effect on pairs of states.

the diagonal of the matrix product is to follow the path:

$$q_3 \to q_1q_2 \to q_1q_3 \to q_2q_3 \to q_1 \to q_2 \to q_3,$$

which corresponds to the matrix product:

$$A_eA_{1,2}A_{1,3}A_lA_{1,3}A_{1,2} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 2 \end{pmatrix},$$

which is of length 6.

3.4.2 Three Matrices Case

We now restrict the possible sets to a fixed number of matrices.

Our goal is to find a graph of pairs with a shortest cycle containing a single state and a pair of two different states as long as possible. One natural way to tackle this question is to start by searching for graphs of pairs with a high diameter. This question has been studied in the paper [GGG$^+$17]. In this paper, it is conjectured that for a set of two permutations, the diameter of the square graph is at most of the order $n^2/4$, and a set of permutation reaching this bound is given.

In this section, we build a set of matrices with JSR equal to $2^{\frac{1}{(n-1)^2/4+5(n-1)/4-5}}$. The automaton of [GGG$^+$17] is represented in Fig. 3.3, and is such that the shortest path from q_2q_4 to $q_{k+2}q_{k+4}$ is of length $n^2/4 + 5n/4 - 7$.

To build our set of matrices, we add an additional state q_0 and a new letter c. c sends q_0 to q_2 and q_4, and deletes other elements. We also add edges from q_{k+2} and q_{k+4} to q_0 labelled by letter a, as represented in Fig. 3.13.

With this setting, starting from a single base vector, the only way to increase the weight of our vector is to send it to q_0, to double the weight with letter c, to follow the only path in the graph of pairs, leading to $q_{k+2}q_{k+4}$, and to go back to q_0 with letter a. This path is $n^2/4 + 5n/4 - 5$ letter long, for $n + 1$ states, which is the claimed result.

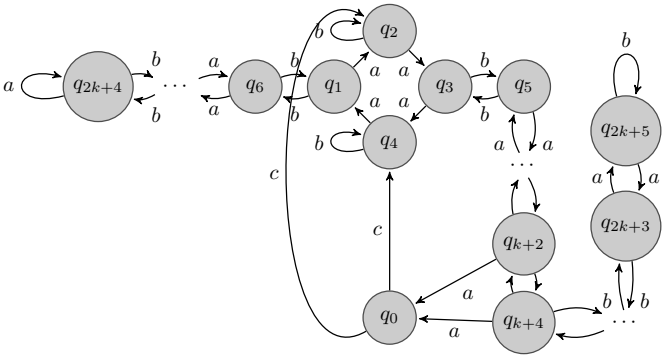


Figure 3.13: Graph representation of the set of three matrices reaching the new bound on the JSR.

3.5 Conclusion

We studied the diameter of the square graph of automata and of permutation sets. We first described a series of n-state automata with diameter of the pair digraph equal to $\frac{n^2}{4} + o(n^2)$, thus lower bounding the maximal diameter of the square graph of a set of permutations. Second, we showed that the diameter of the square graph of a set of permutation was lower than $n(n-1)/2 - 1$, which is a new upper bound. Third, we used these results to improve a theorem on the JSR of nonnegative integer matrices.

Chapter 4

Short Words for Subset Reachability

Abstract

In this chapter, we first study the length of shortest words reaching subsets of states in synchronizing automata. We provide an automata family with subsets that cannot be reached by words shorter than $2^n/n$, thus disproving a recent conjecture of Don. We then analyze relaxed versions of this conjecture. Second, we focus on the converse problem: ensuring that a state is not in the image of some word. We provide a counterexample to a lemma used in a recent tentative improvement of the Pin-Frankl bound for synchronizing automata. This example naturally leads us to formulate an open question, whose answer could fix the line of the proof, and improve the Pin-Frankl bound.[1]

4.1 Overview

In this chapter, we focus on the length of shortest words reaching subsets of states, or such that some states are not in their image. Both problems are natural control problems. Indeed, one could want to ensure that, after applying a sequence of commands of short length, the system is guaranteed to be in a precise set of states or avoids some states. For example, in formal verification, one tries to avoid failure states or ensure a valid terminal state [CW96].

[1] Most results presented in this chapter have been published in [GGG+17, GJ18, GJT15]

In Section 4.2, we study the length of shortest words reaching some subsets of states. In [Don16], Don showed that for any reachable subset of k states of any automaton of a specific class, there exists a word of length $n(n-k)$ reaching this subset. He also conjectured that it is true for any synchronizing automaton. First, we show that this conjecture does not hold by providing a family of counterexamples where the length of the shortest word reaching a set of $n-2$ states is quadratic. Second, we analyze alternative versions of the conjecture and the questions they raise. In particular, we build a family of automata such that a set of $\lfloor \frac{n}{2} \rfloor$ states cannot be reached by a word shorter than $2^n/n$.

In Section 4.3, we provide an automaton with a state which is in the image of any word of length lower than or equal to $n+1$. This automaton is a counterexample to a lemma used in a recent tentative improvement of the Pin-Frankl bound for synchronizing automata [Tra11]. It naturally leads us to formulate an open question, whose answer could fix the line of the proof, and improve the Pin-Frankl bound.

4.2 Subset Reachability

In this section, we provide an answer to Don's conjecture on subset reachability and analyze questions arising from it.

Conjecture 4.2.1 (Conjecture 18 in [Don16])**.** *Let $\mathscr{A} = (Q; \Sigma; \delta)$ be an n-state automaton. If $S \subset Q$ is a set of size k and there exists a word w such that $Qw = S$, then there exists a word with this property of length at most $n(n-k)$.*

This conjecture would imply that Conjecture 1.2.2 is also true [Don16, Proof of Theorem 12]. Indeed, any pair would be reachable with a word of length $n(n-2)$. Since in any synchronizing automaton there is also a reachable pair which can be synchronized with a single letter, reaching this pair with a word of length $n(n-2)$ and then synchronizing it with a single letter leads to a synchronizing word of length $n(n-2)+1 = (n-1)^2$.

Complexity Implication of the Conjecture Conjecture 4.2.1 would not only imply Conjecture 1.2.2 but also the collapse of polynomial hierarchy[2]. Indeed, it is shown in [BV16] that the following problem is PSPACE-complete:

[2]This was kindly communicated to us by an anonymous reviewer of the conference version of this work.

Problem 4.2.2. *Given a DFA $\mathscr{A} = (Q, \Sigma, \delta)$ and a non-empty subset $S \subseteq Q$, is it true that S is reachable in $\mathscr{A}$?*

Would Conjecture 4.2.1 hold true, then, whenever S is reachable, one could guess a word w of length at most $n(n-k)$ where $n = |Q|$ and $k = |S|$ and then check in polynomial time that $Qw = S$. Thus, the above problem would be in NP whence NP = PSPACE. This implies that the conjecture 4.2.1 was too strong from a theoretical perspective.

First Counterexample In order to analyze Conjecture 4.2.1, we need the following lemma:

Lemma 4.2.3. *Let $\mathscr{A} = (Q; \Sigma; \delta)$ be an automaton. Let $S \subset Q$ be a reachable subset, and $w \in \Sigma^*$ be a shortest word such that $Qw = S$. Let us write $w = l_1 \dots l_k$, with k the length of w and $l_i \in \Sigma$ the letters of w. Let the subsets $S_i = Ql_1 \dots l_i$ with $0 < i < k$ be the subsets reached by prefixes of w.*

Then, all the subsets S_i are different, and they all have a cardinality larger than or equal to the cardinality of S.

Proof. If two subsets $S_i = Ql_1 \dots l_i$ and $S_j = Ql_1 \dots l_j$, with $i < j$ are equal, then the word $l_1 \dots l_i l_{j+1} \dots l_k$, which is shorter than w, also reaches S, which is a contradiction.

In a complete deterministic automaton, each state has exactly one image. Therefore for any word w and and set S, we have $|Sw| \leq |S|$. □

This lemma implies that any reachable set of size $n-1$ can be reached with a word of length n. Indeed, there are only $n+1$ different subsets of cardinality higher than or equal to $n-1$. Therefore, we search for a counterexample to Conjecture 4.2.1 with a set of size $n-2$ which cannot be reached with a word of length lower than or equal to $2n$.

One way to build such automata is to define the letters in the following way. First choose a set of permutation letters such that the square graph of the automaton limited to these letters has a large diameter. Second, choose a pair (q_i, q_j) such that the shortest path to an other pair (q_k, q_l) in the square graph has a large length D. Then, add a 2-deficient letter l such that $Q \backslash Ql = \{q_i, q_j\}$. With this construction, the set $Q \backslash \{q_k, q_l\}$ cannot be reached by words shorter than $D + 1$.

In Fig. 4.1, we present the automaton $\mathscr{P}_{2,n}$, with n congruent to 3 modulo 4 which is built in that way.

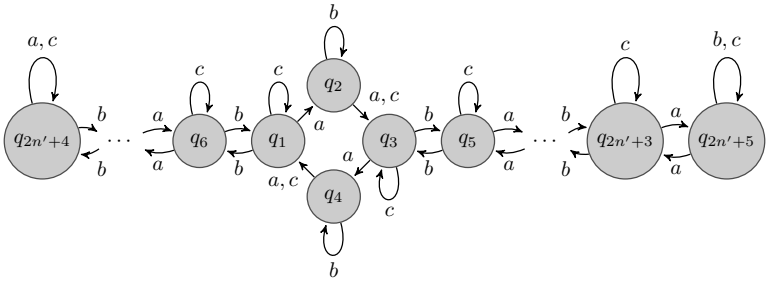


Figure 4.1: The automaton $\mathscr{P}_{2,n}$.

Proposition 4.2.4. *The shortest word reaching the set $Q\setminus\{q_{n'+2}, q_{n'+4}\}$ with $n' = (n-5)/2$ in the automaton in $\mathscr{P}_{2,n}$ is of length $n^2/4 + 5n/4 - 6$.*

Proof. We first notice that letters a and b of the automaton $\mathscr{P}_{2,n}$ are a set of permutation letters studied in [GGG+17] and that we have seen earlier in this thesis such that the shortest path from the pair q_2q_4 to $q_{n'+2}q_{n'+4}$ is of length $n^2/4 + 5n/4 - 7$. As a and b are permutations, we have $Qa = Q$ and $Qb = Q$, so the first letter of any shortest word reaching a set S should be c.

Second, we notice that letter c maps q_2 to q_3 and q_4 to q_1, and the other states to themselves, so $Qc = Q \setminus \{q_2, q_4\}$ and is of cardinality $n-2$. Moreover, if letter c is applied to any set containing $n-2$ states, it either sends it to $Q \setminus \{q_2, q_4\}$, or to a set of lower cardinality. Therefore, due to Lemma 4.2.3, in any shortest word reaching $Q \setminus \{q_{n'+2}, q_{n'+4}\}$, the letter c can only be at the first position.

Third, as a and b are permutations, the image of a set S by a word $w \in \{a, b\}^*$ is equal to the complement of the image of the complement of S by w, i.e. $Sw = Q \setminus ((Q \setminus S)w)$. In our case, applying letters a and b to a set containing $n-2$ states is equivalent to applying it to the complement pair and taking the complement of the image as result. Therefore, the shortest word $w \in \{a, b\}^*$ mapping the set $Q\setminus\{q_2, q_4\}$ to $Q\setminus\{q_{n'+2}, q_{n'+4}\}$ is also the shortest word mapping $\{q_2, q_4\}$ to $\{q_{n'+2}, q_{n'+4}\}$, which is of length $n^2/4 + 5n/4 - 7$. Therefore, cw is the shortest word reaching $Q \setminus \{q_{n'+2}, q_{n'+4}\}$, and is of length $n^2/4 + 5n/4 - 6$. □

The automaton of $\mathscr{P}_{2,n}$ is a generic counterexample to Conjecture 4.2.1, which claims that the distance from Q to any set of size $n-2$ should be $2n$.

We notice that the reachable sets of $\mathscr{P}_{2,n}$ can be reached with words of

polynomial length. Therefore, one can wonder if it is a general property:

Problem 4.2.5. *Let $\mathscr{A} = (Q; \Sigma; \delta)$ be an n-state automaton. If there exists a word w such that $Qw = S$, does there always exist a word of polynomial length with respect to $|S|$ with this property?*

Answer to Problem 4.2.5: Exponential Length Shortest Words For any n, we can build an automaton such that it has a set $S \subset Q$ containing $\lfloor \frac{n}{2} \rfloor$ states which cannot be reached with a word shorter than $L = \binom{n-1}{\lfloor \frac{n}{2} \rfloor}$. This is shown in Definition 4.2.6 and Proposition 4.2.7 below.

Definition 4.2.6. *We define the automaton $\mathscr{P}_{3,n}$ as follows. It has n states $q_1 \dots q_n$, a letter a and letters l_i. First, we define letter a as follows:*

$q_1 a = q_2$

$q_i a = q_i$ *for* $1 < i \leq \lfloor \frac{n}{2} \rfloor + 1$

$q_i a = q_2$ *for* $\lfloor \frac{n}{2} \rfloor + 1 < i \leq n$.

Then, we take all subsets $S \subset (Q \setminus q_1)$ with $|S| = \lfloor \frac{n}{2} \rfloor$ and enumerate them as $S_1, \dots, S_L$, with $L = \binom{n-1}{\lfloor \frac{n}{2} \rfloor}$. We define letters l_i such that $S_i l_i = S_{i+1}$ and $(Q \setminus S_i) l_i = q_1$.

In Fig. 4.2, we show a representation of $\mathscr{P}_{3,n}$ (we do not represent the effect of $l_1, \dots, l_{L-1}$ on other states than q_1 for the sake of clarity). We show that the set S_L cannot be reached with a word shorter than the size of the alphabet. Indeed, in this automaton, the only way to exclude q_1 from a subset is to use letter a. Moreover, we have that $|Qa| = |S_1| = |S_L|$. Therefore, in order to reach S_L, after using an a, letters reducing the cardinality of the current set cannot be used anymore. This implies that there is only one possible letter following any prefix of a shortest word reaching S_L.

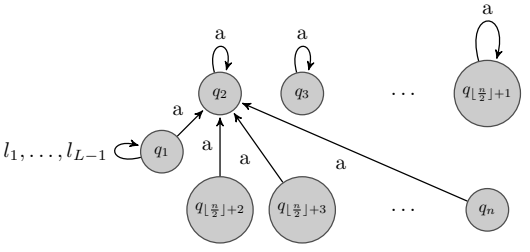


Figure 4.2: Part of the automaton $\mathscr{P}_{3,n}$.

Proposition 4.2.7. *The shortest word reaching the set*

$$S_L = \{q_{n-\lfloor \frac{n}{2} \rfloor+1}, \ldots, q_n\}$$

in the automaton $\mathscr{P}_{3,n}$ *is* $\binom{n-1}{\lfloor \frac{n}{2} \rfloor}$ *letters long, which is larger than* $2^n/n$ *(for* $n > 6$*).*

Proof. From the definition of letters l_i, it is clear that the word $al_1l_2 \ldots l_{L-1}$ maps Q on S_L.

Moreover, it is the shortest of such words. Indeed, let w be a word such that $Qw = S_L$. We notice that the only letter which maps q_1 on another state is a. Therefore, as S_L does not include q_1 we have that: first, w must contain at least one letter a; second, every letter after the last a does not map any state of the set reached by the prefix of w preceding this letter on q_1.

Now, if we consider a set S_i of $\lfloor \frac{n}{2} \rfloor$ states from $q_2, \ldots, q_n$, the only letter l which does not map any states of S_i on q_1 is l_i. Therefore, if a subset S_i was reached with a prefix of w containing the last a of w, the only possible letter l such that $q_1 \notin S_i l$ is l_i. This implies that the letter following the prefix is l_i and that the next subset is $S_i l_i = S_{i+1}$. Since $Qa = \{q_2, \ldots, q_{\lfloor \frac{n}{2} \rfloor+1}\}$ is already of cardinality $\lfloor \frac{n}{2} \rfloor$, the first subset after the last a must be $\{q_2, \ldots, q_{\lfloor \frac{n}{2} \rfloor+1}\} = S_1$. Then, by induction, the letters following the last a in w are $l_1, \ldots, l_{L-1}$, in that order. Therefore, any word w reaching S_l has a suffix $al_1l_2 \ldots l_{L-1}$, which is $\binom{n-1}{\lfloor \frac{n}{2} \rfloor}$ letters long. Since this suffix is itself a word reaching S_L, it is also the shortest word reaching S_L. □

The number of letters of the automaton $\mathscr{P}_{3,n}$ is exponential with respect to the number of states. We notice that it is possible to build families of automata with a fixed number K of letters, such that the shortest word reaching a particular subset has exponential length (as a function of n). Indeed, a construction of this kind with 32 letters for any n can be obtained from the construction given in [FK17, Theorem 20].

In the families $\mathscr{P}_{2,n}$ and $\mathscr{P}_{3,n}$, shortest words reaching a defined subset were such that a first letter was used to lower the cardinality of the set, and the following letters were only permutations. Applying the first letter again would indeed have reduced the number of states below the cardinality of this set or reached a set already reached, which is forbidden by Lemma 4.2.3. Moreover, it turns out that these automata have a set S which cannot be reached by a word of polynomial length, but such that some subsets of S can be reached with such word. This leads to the following question:

Problem 4.2.8. *Let $\mathscr{A} = (Q; \Sigma; \delta)$ be an n-state automaton. If $S \in Q$ is a set of size k and there exists a word w such that $Qw \subseteq S$, does there exist a word with this property of length at most $n(n-k)$?*

Answer to Problem 4.2.8: Included Subsets We notice that, although this question is weaker than Conjecture 4.2.1, a positive answer would still imply that Conjecture 1.2.2 is true.

However, even for this weaker version, the answer is negative. Indeed, the automaton $\mathscr{E}$ in Fig. 4.3, which was introduced in [GJT15], is such that the set $S = \{1, 2, 3\}$ or any of its subset cannot be reached with a word of length lower than 6. This can be verified by a breadth first search algorithm on the power automaton $P(\mathscr{E})$, presented in Fig. 4.4.

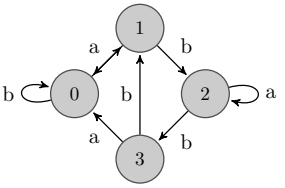


Figure 4.3: The automaton $\mathscr{E}$.

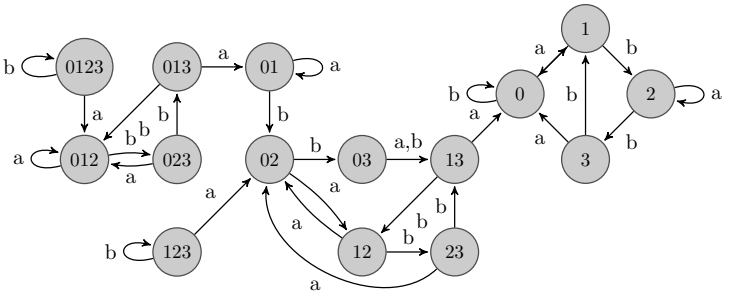


Figure 4.4: The power automaton $P(\mathscr{E})$.

4.3 The Excluding State Approach

For more than 30 years, the best proven bound for Conjecture 1.2.2 has been $(n^3-n)/6-1$, obtained in [Fra82] and [Pin83], and rediscovered independently

in [KRS87]. Recently, a tentative improvement to $n(7n^2 + 6n - 16)/48$ has been proposed in [Tra11]. However, as mentioned later by the author on ArXiv, there is a flaw in the proof. Nevertheless, since the publication of [Tra11], many new papers are citing this result, and no publication clearly confirms that the proof is not valid. In this section, we make this point clear by providing a counterexample to Lemma 3 in [Tra11]. The lemma is the following:

Lemma 4.3.1 (Lemma 3 in [Tra11])**.** *Let Q be the set of states of a synchronizing strongly connected n-state DFA. Then for any state q there exists a word w of length not greater than n such that $q \notin Qw$. For any $k < n$ there are at least k states $q_1, ..., q_k$ and words $w_1, ..., w_k$ of length not greater than k such that $q_i \notin Qw_i$, $1 \leq i \leq k$.*

We refute the lemma by exhibiting an automaton such that, for one state q_0, there is no word w of length smaller than or equal to the number of states with the property that $q_0 \notin Qw$.

Counterexample 4.3.2. *The automaton represented in Fig. 4.5 (named $\mathscr{E}$ in the present thesis) is a synchronizing automaton, as the word abbababba is a synchronizing word. However, the shortest word w such that $q_0 \notin Qw$ is $w = abbaba$. Since the automaton has only 4 states and w is 6 letters long, this contradicts Lemma 4.3.1.*

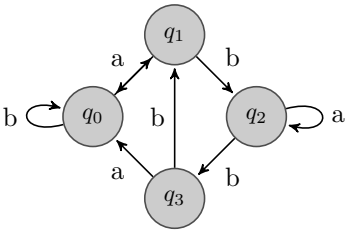


Figure 4.5: A counterexample to Lemma 4.3.1.

Lemma 4.3.1 was a key step in the improvement on the maximal length of a shortest synchronizing word. We observe that a weaker version of Lemma 1 could still improve the Pin-Frankl bound. In fact, any value proportional to the number of states of the automaton would lead to an improvement of the bound. This motivates us to raise the following open question:

Open question 4.3.3. *Let Q be the set of states of a synchronizing strongly connected $n-$state DFA.*

Is there a constant c such that, for any state $q \in Q$, there exists a word w of length not greater than cn such that $q \notin Qw$?

4.4 Conclusion

In this chapter, we focused on subset reachability in synchronizing automata. The first problem considered was the maximal length of the shortest words reaching some subsets of states. This was motivated by [Don16, Conjecture 18], which stated that this length should be lower than or equal to $n(n-k)$, with k the number of states in the subset.

We started by presenting a family of automata built from a set of permutations with a large square graph diameter. This family has subsets of $n-2$ states which cannot be reached by words shorter than a quadratic value, and therefore is a counterexample to Conjecture 4.2.1. Then, we analyzed modified versions of Conjecture 4.2.1. First, we asked if the length of the shortest word reaching a subset could be bounded by a polynomial value in n. Second, we asked if the bound of the conjecture would be true if we considered *subsets included* in the target subset instead of the target subset itself. Both cases led to negative answers.

We finished by presenting an automaton with a state which cannot be excluded by a word shorter than $n+2$, contradicting a statement by Trahtman [Tra11]. We proposed an alternative version of this statement which would still improve the Pin-Frankl bound as open problem.

Chapter 5

Completely Reachable Automata and Automata Generating T_n

Abstract

In this chapter, we first analyze the Γ_1-graph construction. The Γ_1-graph is derived from 1-deficient words, and is a key tool for studying completely reachable automata. We introduce the concept of roots of 1-deficient words, which allows to state explicit concatenation rules for these words. Based on these results, we provide a polynomial-time algorithm for constructing the Γ_1-graph. Then, we disprove a conjecture by Bondar and Volkov linking the strong connectivity of this graph and the concatenation of 1-deficient words of completely reachable automata. Finally, we prove an alternative version of this conjecture.

Second, motivated by the Babai conjecture and the Černý conjecture, we study the reset thresholds of automata with the transition monoid equal to the full monoid of transformations of the state set. For automata with n states in this class, we prove that the reset threshold is upper-bounded by $2n^2 - 6n + 5$ and can attain the value $\frac{n(n-1)}{2}$. [1]

[1] Most results presented in this chapter have been published in [GGG+17, GJ18]

5.1 Overview

Completely reachable automata are deterministic finite automata in which every non-empty subset of the state set occurs as the image of the whole state set under the action of a suitable input word. These appeared in the study of descriptional complexity of formal languages [Mas12] and in relation to the Černý conjecture [Don16].

The first focus of this chapter is the Γ_1-graph of completely reachable automata. The Γ_1-graph is built from *1-deficient words*, i.e. words which have an image of cardinality $n-1$. It turns out that this graph can be used as a key tool in the extension method. Indeed, if the Γ_1-graph is strongly connected, then any subset of states can be extended with a 1-deficient word and conversely, the automaton is completely reachable. This property was used by Rystsov to prove a quadratic reset threshold for automata with simple idempotent [Rys00], and is studied in [Don16, BV16]. In [BV18], the Γ_1-graph is extended to the Γ-graph and allows to fully characterize completely reachable automata. More precisely, the Γ-graph is the union of the Γ_k-graphs, which are defined from letters synchronizing k states (see section 3 of [BV18] for the explicit description). Extending the proofs obtained with the Γ_1-graph using the extension method to the Γ-graph could lead to proving quadratic bounds for more classes of automata, and is therefore very promising. In [BV16], Bondar and Volkov conjectured that the Γ_1-graph is strongly connected if any subset is reachable by a concatenation of 1-deficient words. They also discuss the question of building algorithmically the Γ_1-graph and show that straightforward approaches computing all the 1-deficient words would have an exponential complexity. This complexity question is then let open.

In [BV16], the emphasis is on automata in this class with minimal transition monoid size. In the second part of this chapter, we focus on automata being in a sense the extreme opposites of those studied in [BV16], namely, on automata of maximal transition monoid size. In other words, we consider automata *with full transition monoid*, i.e., transition monoid equal to the full monoid of transformations of the state set; clearly, automata with this property are completely reachable. There are several reasons justifying special attention to automata with full transition monoid. First, as observed in [BV16], the membership problem for this class of automata is decidable in polynomial time (of the size of the input automaton) while the complexity of membership in the class of all completely reachable automata still remains unknown. Second, this class contains automata that correspond to Brzozowski's most complex regular

languages [Brz13] and to other regular languages that play a distinguished role in descriptive complexity analysis. Finally, and most importantly from our viewpoint, automata with full transition monoid are synchronizing and their synchronization issues constitute a sort of meeting point for two famous open problems—the *Babai conjecture* and the *Černý conjecture*. Next, we recall these conjectures and outline our contributions in view of these problems.

The Babai Conjecture. Let A be a set of generators of a finite group G. The *Cayley graph* $\Gamma(G, A)$ consists of G as the set of vertices and the edges $\{g, ga\}$ for all $g \in G$, $a \in A$. The *diameter* of $\Gamma(G, A)$ is the maximum among the lengths of shortest paths between any two vertices. In group theory terms, the diameter of $\Gamma(G, A)$ is the smallest ℓ such that every $g \in G$ can be represented as $g = a_1^{\varepsilon_1} a_2^{\varepsilon_2} \cdots a_\ell^{\varepsilon_\ell}$, where $\varepsilon_i \in \{1, -1\}$ and $a_i \in A$ for all $i = 1, \ldots, \ell$. The *diameter* $\mathrm{diam}(G)$ of G is the maximal diameter of $\Gamma(G, A)$ among all generating sets A of G. The notion of group diameter is related to the growth rate in groups, expander graphs, random walks on groups and their mixing times, see, e.g., [SC04, Hel15]. Recently, the following conjecture received significant attention:

Conjecture 5.1.1 (Babai [BS92])**.** *The diameter of each non-abelian finite simple group G does not exceed $(\log |G|)^{O(1)}$, where the implied constant is absolute.*

Note that for the case of the symmetric group S_n, this conjecture readily implies $\mathrm{diam}(S_n) \leq n^{O(1)}$. (The group S_n is not simple but for $n \geq 5$ it contains a non-abelian simple subgroup of index 2.)

The Babai conjecture was proved for various classes of groups, but despite intensive research effort it remains open, see [HS14] for an overview. In the case of S_n, a recent breakthrough gives only a quasipolynomial upper bound, namely, $\exp(O((\log n)^4 \log\log n)$, and it relies on the Classification of Finite Simple Groups [HS14]. It is even more astonishing if we compare it to the best known lower bound in this case: for the classical set of generators consisting of the transposition $(1, 2)$ and the full cycle $(1, 2, \ldots, n)$, every permutation in S_n can be expressed as a product of at most $\sim\frac{3n^2}{4}$ (asymptotically) generators [Zub98].

In order to make the relationship between the Černý and the Babai conjectures more visible, we borrow from [AV04] the idea of restating the former in terms similar to those used in the formulation of the latter. Let T_n be the full transformation monoid of an n-element set Q. A transformation $t \in T_n$

is a *constant* if there exists $f \in Q$ such that for all $q \in Q$ we have $t(q) = f$. We can state the Černý conjecture as follows: for every set of transformations $A \subseteq T_n$, if the submonoid generated by A contains a constant, then there exists a constant g such that $g = a_1 a_2 \cdots a_\ell$, where $\ell \leq (n-1)^2$ and $a_i \in A$ for all $i = 1, \ldots, \ell$. It is easy to see that this formulation is equivalent to the original one by treating the letters of an automaton as the transformations of its state set since reset words precisely correspond to constant transformations.

Our Contributions. Section 5.2 is devoted to the construction of the Γ_1-graph and its link with *completely reachable automata*. In this section, we first introduce the concept of *root states* of a 1-deficient word. Together with the already well known concepts of *excluded state* and *duplicate state*, it allows us to state explicitly the concatenation rules for 1-deficient words. Second, using these rules, we provide a polynomial time algorithm building the Γ_1-graph. Third, we provide a completely reachable automaton which does not fulfil Bondar and Volkov's conjecture. Finally, we prove a slightly modified version of the conjecture.

Section 5.3 is devoted to the following *hybrid Babai–Černý problem*: given a set of generators A of the full transformation monoid T_n, what is the length $\ell(A)$ of the shortest product $a_1 a_2 \cdots a_\ell$ with $a_i \in A$ which is equal to a constant? Namely, we are interested in the bounds on $\ell(A)$ that depend only on n. The hybrid Babai–Černý problem is a special case of the Černý problem. Indeed, it is a restriction to the class of DFAs with the *transition monoid*, i.e., the transformation monoid generated by the actions of letters, equal to T_n. Of course, the general cubic upper bound is valid, but not the lower bound, since the Černý automata $\mathscr{C}_n$ do not belong to this class (even though they are completely reachable, see [BV16]). In Section 5.3 we establish that the growth rate of $\ell(n)$ is $\Theta(n^2)$, more precisely, we show that $\frac{n(n-1)}{2} \leq \ell(n) \leq 2n^2 - 6n + 5$. We also present the exact values of $\ell(n)$ for small values of n resulting from our computational experiments. Our contribution can be also seen as a progress towards resolution of Conjecture 3 from [Sal03].

5.2 Γ_1-graph of Completely Reachable Automata

First, let recall that the *excluded* state of a 1-deficient word w is the state which is not in its image. Conversely, the *duplicate* state of w is the state which is the image of two states by w. From these two concepts, the Γ_1-

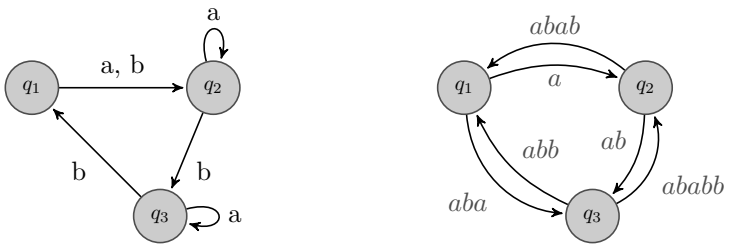


Figure 5.1: The automaton $\mathscr{C}_3$ and its Γ_1-graph.

graph is defined as the unlabelled graph with the same nodes as the automaton and edges corresponding to all directed pairs $(excl(w), dupl(w))$ of 1-deficient words w. Figure 5.1 shows the automaton of the the Černý family $\mathscr{C}_3$ and the corresponding Γ_1-graph. For example, the word a induces the edge (q_1, q_2) in Γ_1 because q_1 is not in the image of a, so $excl(a) = q_1$ and q_2 is the image of both q_1 and q_2 by a, so $dupl(a) = q_2$. The edges have been labelled in gray for illustration purpose, and the words that yield the same edge as shorter words are not explicitly mentioned. E.g. ba or aa gives the same edge as a.

It is shown by Don in [Don16] that if the Γ_1-graph is cyclic, then the automaton is completely reachable. In the work of Bondar and Volkov [BV16], which study completely reachable automata, it is noticed that complete reachability holds under the weaker assumption of strong connectivity of the Γ_1-graph. However, they also notice that the converse is not true: there are completely reachable automata for which the Γ_1-graph is not strongly connected. In order to show this, they provide an example using 2-deficient letters. These letters do not contribute to the Γ_1-graph but allow to reach subsets which are not reachable with 1-deficient words. The restriction of this statement to 1-deficient words was formulated as the following conjecture:

Conjecture 5.2.1 (Conjecture 3 in [BV16])**.** *If for every proper non-empty subset S of the state set of an n-state DFA $\mathscr{A} = (Q; \Sigma; \delta)$ there is a product w of 1-deficient words such that $S = Qw$, the graph $\Gamma_1(\mathscr{A})$ is strongly connected.*

The paper [BV16] also leaves open the algorithmic aspects of building the Γ_1-graph. The authors point out that building the Γ_1-graph is non trivial, since the number of transformations of rank $n-1$ can reach $n!C_2^n$. Moreover, a fast algorithm building the Γ_1-graph would provide a fast positive criterion for complete reachability of an automaton and could be used to experimentally investigate properties of the Γ_1-graph.

In order to formalize the link between strong connectivity of the graph Γ_1 and subset reachability, we need the following definition of a strongly connected graph:

Definition 5.2.2. *A directed graph $\Gamma = (N, E)$ with nodes N and edges E is strongly connected iff for any set $S \subset N$, there exists an edge $e = (n_1, n_2) \in E$ with $n_1 \notin S$ and $n_2 \in S$.*

We say that the edge $e = (n_1, n_2) \in E$ with $n_1 \notin S$ and $n_2 \in S$ is *intersecting* the set S.

If Γ_1 is strongly connected, then the automaton is completely reachable. Indeed, if any set $S \subset Q$ is intersected by an edge of Γ_1, there exists a 1-deficient word w with $dupl(w) \in S$ and $excl(w) \notin S$. This first implies that each state in S has a preimage by w. Second, as $dupl(w) \in S$, it implies that there is a state in S which has two preimages. Therefore $|Sw^{-1}| > S$, and w is extending S. As this holds for every subset S, the automaton is completely reachable.

5.2.1 Algorithmic Construction of Γ_1

In order to analyze the effect of word concatenation, we cannot restrict ourselves to the concepts of excluded state and duplicate state of a word. Indeed, for two 1-deficient letters a and b, the knowledge of these states is not enough to determine whether ab or ba are of deficiency one or two. To analyze this question, we need an additional concept: the *roots* of a 1-deficient word. The roots are the states which are sent on the duplicate state. More formally, for a 1-deficient word w, $root(w) = \{q_i | q_i w = dupl(w)\}$. We notice that applying a 1-deficient word w to a set $S \in Q$ does not decrease the cardinality of the set if at least one of the roots of w is not in S. This leads to the following concatenation rule:

Lemma 5.2.3. *Let $\mathscr{A} = (Q; \Sigma; \delta)$ be an n-state DFA. Let $w_1, w_2 \in \Sigma^*$ be two 1-deficient words. If $excl(w_1) \notin root(w_2)$, then $w_1 w_2$ is 2-deficient. If $excl(w_1) \in root(w_2)$ then $w_1 w_2$ is 1-deficient. In that case,*

$$excl(w_1 w_2) = excl(w_2),\ dupl(w_1 w_2) = dupl(w_1) w_2,\ root(w_1 w_2) = root(w_1).$$

Proof. If $excl(w_1) \notin root(w_2)$, then $root(w_2) \subset Qw_1$, and the two states in $root(w_2)$ are synchronized by w_1 while the other states are permuted, so we have $|Qw_1 w_2| = |Qw_1| - 1 = |Q| - 2$.

Moreover, if $excl(w_1) \in root(w_2)$, then all the states in Qw_1 have a different image by w_2 and $|Qw_1w_2| = n - 1$. In that case, we have the following properties.

The state $dupl(w_1w_2)$ is the image of $dupl(w_1)$ by word w_2, i.e. $dupl(w_1)w_2$.

The states $root(w_1w_2)$ are equal to $root(w_1)$, since for these states we have that $q_rw_1 \in dupl(w_1)$ and therefore $q_rw_1w_2 \in dupl(w_1)w_2$.

The state $excl(w_1w_2)$ is $excl(w_2)$, since $excl(w_2) \notin Qw_2$ and $Qw_1w_2 \subseteq Qw_2$. □

Of course, the concatenation of a 1-deficient word with a permutation also provides a 1-deficient word. In that case, the resulting word has the following properties:

Lemma 5.2.4. *Let $\mathscr{A} = (Q; \Sigma; \delta)$ be an n-state DFA. Let $w \in \Sigma^*$ be a 1-deficient word and $p \in \Sigma^*$ be a permutation. Then both pw and wp are 1-deficient words. Moreover,*

$$excl(wp) = excl(w)p,\ excl(pw) = excl(w),$$

$$dupl(wp) = dupl(w)p,\ dupl(pw) = dupl(w),$$

$$root(wp) = root(w),\ root(pw) = root(w)p^{-1}.$$

These two lemmas allow to build the Γ_1-graph algorithmically. Indeed, we can identify which words can be concatenated into other 1-deficient words, which induce edges in Γ_1. If two words combine into a 2-deficient word, then the concatenation does not contribute to Γ_1. The Algorithm 1 provides a pseudo code implementing the effect of Lemmas 5.2.3 and 5.2.4. The algorithm computes all 4-tuples composed of the two root states, the excluded state and the duplicate state of 1-deficient letters of the automaton, then computes the pairs of excluded and duplicate states for all 1-deficient words of the automaton. Line 1 initiates the list for letters of deficiency 1. Line 2 initiates the lists in which the results are stored. In lines 3 to 11 the 4-tuples are computed for single letters. Lines 12 to 33 describe a loop corresponding to increasing the length of words tested by one letter if new results were obtained at the previous step. Inside of this, lines 12 to 30 test each pair generated at the previous step. For each of them, lines 14 to 21 test the concatenation with a permutation letter and lines 22 to 29 test the concatenation with another 1-deficient letter.

In order to prove the properties of the algorithm, we need the following properties about 4-tuples obtained from two different words:

Algorithm 1 Construction algorithm for the Γ_1-graph

INPUT*: An automaton (Q, Σ, δ). We consider $|Q| = n$. The alphabet is separated between permutation letters and 1-deficient letters as follows: $\Sigma = \Sigma_0 \cup \Sigma_1$, with m_0 permutation letters $p_1, \ldots, p_{m_0}$ in Σ_0 and m_1 1-deficient letters $l_1, \ldots, l_{m_1}$ in Σ_1.*

OUTPUT*: The exhaustive list of pairs composed of $excl(w)$ and $dupl(w)$ for any 1-deficient word w of the automaton.*

Initiate empty list L_0
▷ List of 4-tuples of single letters
Initiate empty lists L_1, L_2, L_3
▷ List of pairs. L_1 for the final output, L_2 for the previous step of the algorithm, L_3 for the current step of the algorithm
▷ Single letters computation
for $i = 1, \ldots, m_1$ **do**
    Compute $excl(l_i)$, $dupl(l_i)$, $root1(l_i)$, $root2(l_i)$
    if $[excl(l_i), dupl(l_i), root1(l_i), root2(l_i)]$ is not in L_0 **then**
        append $[excl(l_i), dupl(l_i), root1(l_i), root2(l_i)]$ to L_0
    end if
    if $[excl(l_i), dupl(l_i)]$ is not in L_1 **then**
        append $[excl(l_i), dupl(l_i)]$ to L_1, L_2 and L_3
    end if
end for
▷ Words computation
while L_3 is not empty **do** ▷ If the algorithm did not stagnate
    Empty the list L_3 ▷ Reset current step
    for $i = 1, \ldots, length(L_2)$ **do**
        $[V_1, V_2] = L_2(i)$ ▷ Test each pair added in the previous step
        for $j = 1, \ldots, m_0$ **do** ▷ With each possible permutation
            $TestPair = [V_1 p_j, V_2 p_j]$
            if $TestPair$ is not in L_1 **then** ▷ If the result is new, store it
                append $TestPair$ to L_1 and L_3
            end if
        end for
        for $j = 1, \ldots, m_1$ **do** ▷ With each possible 1-deficient letter
            ▷ Test if the letter can be concatenated at right
            if V_1 equal $root1(l_j)$ or $root2(l_j)$ **then**
                $TestPair = [excl(l_j), V_2 l_j]$
                if $TestPair$ is not in L_1 **then** ▷ If the result is new, store it
                    append $TestPair$ to L_1 and L_3
                end if
            end if
        end for
    end for
    Empty the list L_2 ▷ Reset previous step
    $L_2 = L_3$ ▷ Current step becomes previous step
end while

Lemma 5.2.5. *Let w_1 and w_2 be two 1-deficient words of an n-state DFA $\mathscr{A} = (Q; \Sigma; \delta)$ such that*

$$root(w_1) = root(w_2),\ excl(w_1) = excl(w_2),\ dupl(w_1) = dupl(w_2).$$

1. *If l is a 1-deficient letter such that $w_1 l$ is 1-deficient, then $w_2 l$ is also 1-deficient and*

$$root(w_1 l) = root(w_2 l),\ excl(w_1 l) = excl(w_2 l),\ dupl(w_1 l) = dupl(w_2 l).$$

2. *If p is a permutation letter, then pw_1, $w_1 p$, pw_2 and $w_2 p$ are 1-deficient words and*

$$root(pw_1) = root(pw_2),\ excl(pw_1) = excl(pw_2),\ dupl(pw_1) = dupl(pw_2),$$

$$root(w_1 p) = root(w_2 p),\ excl(w_1 p) = excl(w_2 p),\ dupl(w_1 p) = dupl(w_2 p).$$

Proof. This is a direct application of Lemma 5.2.3 and Lemma 5.2.4.

1. If l is a 1-deficient letter such that $w_1 l$ is 1-deficient, then from Lemma 5.2.3 we have that $excl(w_1) \in root(l)$. As $excl(w_1) = excl(w_2)$, we have $excl(w_2) \in root(l)$ and by Lemma 5.2.3 $w_2 l$ is 1-deficient. Then, again from Lemma 5.2.3,

$$root(w_1 l) = root(w_1) = root(w_2) = root(w_2 l),$$

$$excl(w_1 l) = excl(l) = excl(w_2 l),$$

and

$$dupl(w_1 l) = dupl(w_1) l = dupl(w_2) l = dupl(w_2 l).$$

2. If p is a permutation letter, then Lemma 5.2.4 implies

$$root(pw_1) = root(w_1) p^{-1} = root(w_2) p^{-1} = root(pw_2),$$

$$excl(pw_1) = excl(w_1) = excl(w_2) = excl(pw_2),$$

$$dupl(pw_1) = dupl(w_1) = dupl(w_2) = dupl(pw_2),$$

$$root(w_1 p) = root(w_1) = root(w_2) = root(w_2 p),$$

$$excl(w_1 p) = excl(w_1) p = excl(w_2) p = excl(w_2 p),$$

and

$$dupl(w_1p) = dupl(w_1)p = dupl(w_2)p = dupl(w_2p).$$

□

Corollary 5.2.6. *Let w_1 and w_2 be two 1-deficient words of an n-state DFA $\mathscr{A} = (Q; \Sigma; \delta)$ such that*

$$root(w_1) = root(w_2),\ excl(w_1) = excl(w_2),\ dupl(w_1) = dupl(w_2).$$

Then for any 1-deficient word w_3 or permutation word p, the following pairs of words (i) w_1w_3 and w_2w_3, (ii) pw_1 and pw_2, (iii) w_1p and w_2p are such that the first one is 1-deficient if and only if the second one is. In that case both words have the same corresponding 4-tuples composed of the root states, excluded state and duplicate state.

This implies that if, in a greedy exhaustive search for pairs composed of an excluded and a duplicate state, two words w_1 and w_2 have the same pair, then w_2 can be ignored as all the results obtained from concatenating a letter to w_1 replicates the pair obtained by concatenating the same letter to w_2.

Now, we notice that if no new pair is found with words of length $k+1$ compared to pairs obtained with words of length k or lower, then all pairs have been found. More generally, this result is true for 4-tuples corresponding to words:

Lemma 5.2.7. *Let $\mathscr{A} = (Q; \Sigma; \delta)$ be an n-state DFA. If for any 1-deficient word $w_1 \in \Sigma^{k+1}$ there exists a 1-deficient word $w_2 \in \Sigma^{\leq k}$ such that*

$$root(w_1) = root(w_2),\ excl(w_1) = excl(w_2),\ dupl(w_1) = dupl(w_2),$$

then for any 1-deficient word $w_3 \in \Sigma^{\geq k+1}$ there exists a 1-deficient word $w_4 \in \Sigma^{\leq k}$ such that

$$root(w_3) = root(w_4),\ excl(w_3) = excl(w_4),\ dupl(w_3) = dupl(w_4).$$

Similarly, if for any 1-deficient word $w_1 \in \Sigma^{k+1}$ there exists a 1-deficient word $w_2 \in \Sigma^{\leq k}$ such that

$$root(w_1) = root(w_2),\ excl(w_1) = excl(w_2),$$

then for any 1-deficient word $w_3 \in \Sigma^{\geq k+1}$ *there exists a 1-deficient word* $w_4 \in \Sigma^{\leq k}$ *such that*

$$root(w_3) = root(w_4),\ excl(w_3) = excl(w_4).$$

Proof. We proceed by induction.

First, by definition, any 1-deficient word $w_3 \in \Sigma^{k+1}$ is such that there exists a 1-deficient word $w_4 \in \Sigma^{\leq k}$ such that

$$root(w_3) = root(w_4),\ excl(w_3) = excl(w_4),\ dupl(w_3) = dupl(w_4).$$

Second, we assume that for any 1-deficient word $w_1 \in \Sigma^{k+1\leq,\leq k+i}$, $i > 1$, there exists a 1-deficient word $w_2 \in \Sigma^{\leq k}$ such that

$$root(w_1) = root(w_2),\ excl(w_1) = excl(w_2),\ dupl(w_1) = dupl(w_2).$$

Then we have that any 1-deficient word $w_3 \in \Sigma^{\leq k+i+1}$ can be written as:

1. $w_3'p$, if the last letter p is a permutation and the prefix w_3' is 1-deficient;
2. $w_3'l$, if the last letter l is 1-deficient and the prefix w_3' is 1-deficient;
3. or pw_3', if the last letter is 1-deficient and it turns out that the prefix preceding it is a permutation.

We notice that in the three cases above, w_3' is a 1-deficient word of length $k + i$. Therefore, from the assumption above, there exists a 1-deficient word $w_4' \in \Sigma^{\leq k}$ such that

$$root(w_3') = root(w_4'),\ excl(w_3') = excl(w_4'),\ dupl(w_3') = dupl(w_4').$$

Then, the 4-tuple corresponding to pw_3', $w_3'p$ or $w_3'l$ is the same as the one corresponding to pw_4', $w_4'p$ or $w_4'l$, which is of length at most $k + 1$ and 1-deficient. Again from the assumption, any 4-tuple corresponding to pw_4', $w_4'p$ or $w_4'l$ also corresponds to a word $w_4 \in \Sigma^{\leq k}$. Therefore we have that for any 1-deficient word $w_3 \in \Sigma^{\leq k+i+1}$, there exists a 1-deficient word $w_4 \in \Sigma^{\leq n}$ such that

$$root(w_3) = root(w_4),\ excl(w_3) = excl(w_4),\ dupl(w_3) = dupl(w_4).$$

This implies that this property is true for any $i \geq 1$, and therefore the lemma is true.

The proof of the restriction to the excluded state and the duplicate state is the same, except that case (3) is directly solved as the roots are not taken into account, and therefore in that case the word w'_3 already satisfies the condition. □

Theorem 5.2.8. *Algorithm 1 provides the exhaustive list of pairs which can be obtained from 1-deficient words and has a computational complexity in $O(mn^2)$, with m the number of letters and n the number of states of the automaton.*

Proof. First, we notice that any pair obtained through the algorithm corresponds to the application of Lemma 5.2.3 and 5.2.4 to a concatenation of letters, and therefore is a valid pair corresponding to a word.

Second, any pair corresponding to a 1-deficient word is found by the algorithm. We again proceed by induction:

1. first, any pair corresponding to a 1-deficient word w of length 1 (single letter) is in the list L_1 due to lines 3 to 11.

2. Second, assume that all pairs corresponding to any 1-deficient word w of length k are in the list. Then, any word w_{k+1} can be written as

   (a) $w'_k p$, if the last letter p is a permutation and the prefix w'_k is 1-deficient

   (b) $w'_k l$, if the last letter l is 1-deficient and the prefix w'_k is 1-deficient

   (c) or pw'_k, if the last letter is 1-deficient and it turns out that the prefix preceding it is a permutation.

   In all three cases, w'_k is of length k, and therefore the corresponding pair is in the list L_1 (the case pw'_k is therefore directly solved). Therefore, it was also added in this list at some step, at which it was in the list L_2. At that step, from line 12 of Algorithm 1 the pair corresponding to the words $w'_k p$, $w'_k l$ were tested for any 1-deficient letter or permutations. Therefore the pair corresponding to w_{k+1} was either already in the list, or was added in the list at that step.

Now, we notice that the final list L_1 contains at most n^2 pairs. Each of them is added exactly once to the table and tested exactly once, when it is in the list L_2, and if no pair is added in list L_2 after line 32, then due to Lemma 5.2.7 the

algorithm is stopped and all pairs have been found. Once a pair is added, the concatenation with all the letters is tested, so each new pair induces at most m comparisons, which corresponds to line 18 and 25. We notice that, if the list L_1 is defined as a binary vector of length n^2 indicating if each pair is in the list or not, then verifying that a pair is in the list takes $O(1)$ operation, while the naive implementation comparing each new pair with all pairs already in the list would take $O(n^2)$. As there are at most n^2 pair, this loop can be executed at most n^2 times, leading to n^2m operations at most, which provides the algorithmic complexity of $O(mn^2)$. □

5.2.2 A Completely Reachable Automaton with Weakly Connected Γ_1-graph

We now define in Fig. 5.2 the automaton $\mathscr{G}$, which has six states and six letters of rank 5. Using Lemma 5.2.3, we prove that it is a counterexample to Conjecture 5.2.1.

Proposition 5.2.9. *The automaton $\mathscr{G}$ is completely reachable, and the graph $\Gamma_1(\mathscr{G})$ is not strongly connected.*

Proof. Figure 5.4 lists all the words of rank 5 of this automaton, with their duplicate states, excluded states and root states. In the first six lines are listed the single letters. We notice that the roots of any letter are states 1, 5 and 6, and the only letters with 1, 5 or 6 as excluded states are a, e and f. Therefore, due to Lemma 5.2.3 in words of rank 5, letter a can only be preceded by letters a or f, and all the other letters, including e and f, can only be preceded by letters e or f. This leads to the last six lines of the table, listing all words of rank 5. The duplicate states are found by applying the combination rule of Lemma 5.2.3.

The edges of the graph Γ_1 are defined by the excluded state and duplicate state of words in Fig. 5.4, i.e. the second and third columns. Since state 1 does not appear in the duplicate state column, there are no edges of Γ_1 intersecting the set $\{1\}$, and therefore the graph Γ_1 is not strongly connected.

We notice however that any set of $\mathscr{G}$ is reachable. The edges of Γ_1 obtained with single letters form a cycle with a tail containing only state 1, as shown in Fig. 5.3. In this setting, any subset of Q except for $\{1\}$ has an intersecting edge. Therefore, all sets of states except for $\{1\}$ are extendable by 1-deficient words, which implies by induction on the number of states that they are also reachable. The set $\{1\}$ itself can also be reached from set $\{4\}$ by applying

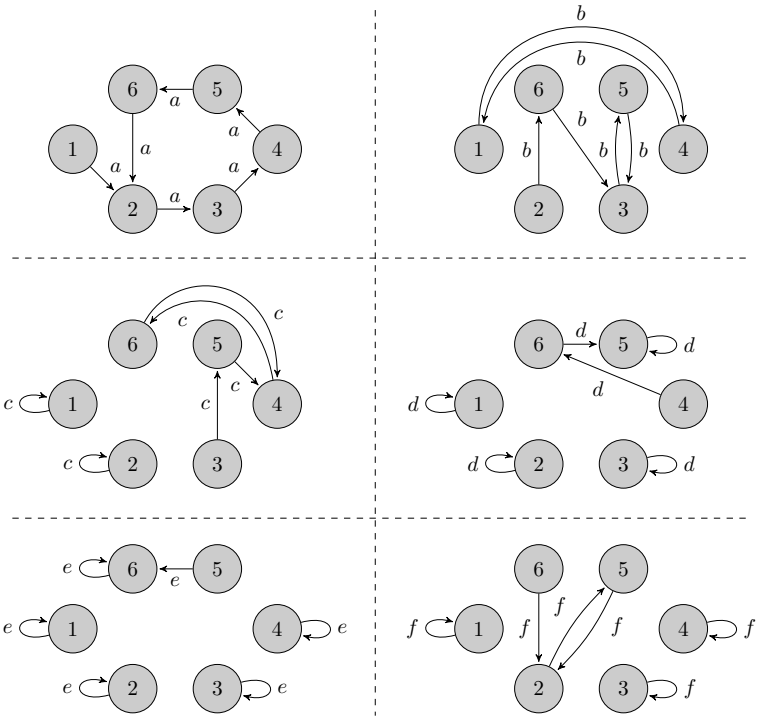


Figure 5.2: The 6 letters of the automaton $\mathscr{G}$.

letter b. Therefore, since $\mathscr{G}$ is completely reachable and $\Gamma_1(\mathscr{G})$ is not strongly connected, it is a counterexample to Conjecture 5.2.1. □

The automaton $\mathscr{G}$ was built as follows: the starting point was the target Γ_1-graph. From the Γ_1-graph, a list of excluded, duplicate and root states was designed. Finally, the letters were tailored in order to achieve these specifications.

Thus, Conjecture 5.2.1 turns out to be false. However, we notice that, in order to reach the set $\{1\}$ in the automaton $\mathscr{G}$, the last letter used was 1-deficient, but was acting as a permutation on the remaining states. This leads to the following observation: if a set S is such that $Qw = S$, and such that $w = w_1w_2$, with w_2 a 1-deficient word, two possible cases arise. Either $|Qw_1| = |Qw|$, or $|Qw_1| = |Qw| + 1$. In the first case, the word w_2 acts as a permutation on the set Qw_1. In the latter case, w_2 synchronizes two states of Qw_1, and we observe the following:

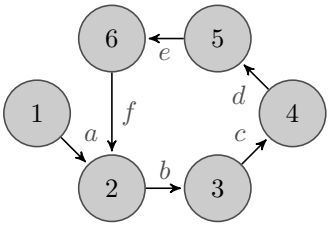


Figure 5.3: The graph $\Gamma_1(\mathscr{G})$ restricted to single letters.

word	excluded state	duplicate state	root states
a	1	2	1, 6
b	2	3	5, 6
c	3	4	5, 6
d	4	5	5, 6
e	5	6	5, 6
f	6	2	5, 6
a^*	1	2, 3, 4, 5 or 6	1, 6
$\{e,f\}^*$	5 or 6	2, 5 or 6	5, 6
$\{e,f\}^*fa^*$	1	2, 3, 4, 5 or 6	5, 6
$\{e,f\}^*b$	2	3 or 6	5, 6
$\{e,f\}^*c$	3	2 or 4	5, 6
$\{e,f\}^*d$	4	2 or 5	5, 6

Figure 5.4: Exhaustive list of words of rank 5.

Lemma 5.2.10. *Let $\mathscr{A} = \{Q, \Sigma, \delta\}$ be an n-state DFA. Let $S \subset Q$, $w_1, w_2 \in \Sigma^*$, and w_2 be a 1-deficient word. If $Qw_1w_2 = S$ and $|Qw_1| = |Qw_1w_2| + 1$, then there is an edge intersecting S in $\Gamma_1(\mathscr{A})$.*

Proof. The edge $(excl(w_2), dupl(w_2))$ is an edge of Γ_1 which intersects S. Indeed, we first have $excl(w_2) \notin S$ because $S = Qw_1w_2 \subseteq Qw_2$, and $excl(w_2) \notin Qw_2$. Second, we have $root(w_2) \subset Qw_1$, because otherwise $|Qw_1| = |Qw_1w_2|$, which contradicts the assumption that $|Qw_1| = |Qw_1w_2| + 1$. As $dupl(w_2)$ is the image of $root(w_2)$ by w_2, it implies that $dupl(w_2)$ is in $Qw_1w_2 = S$. □

Based on this observation, we prove the following modified version of Conjecture 5.2.1:

Theorem 5.2.11. *Let $\mathscr{A} = \{Q, \Sigma, \delta\}$ be an n-state DFA. If for every proper non-empty subset $S \subset Q$ there is a word $w = w_1w_2 \in \Sigma^*$ with $S = Qw$, such that w_2 is 1-deficient and $|Qw_1| = |Qw| + 1$, then the graph $\Gamma_1(\mathscr{A})$ is strongly connected.*

Proof. It is a direct application of Lemma 5.2.10 on any set of states $S \subset Q$ of the automaton. Indeed, it implies that any set has an intersecting edge in Γ_1, which is the definition of a strongly connected graph. □

In particular, this theorem applies to automata with a simple idempotent [Rys00], and for automata with letters generating the full transformation group T_n.

5.3 Automata with Full Transition Monoid

2.1. Naïve Construction. Recall that, on the one hand, the Černý automata $\mathscr{C}_n$ from [Černý64] have two letters of which one acts as a cyclic permutation and the other fixes all states, except one, which is mapped to the next element in the cyclic order defined by the cyclic permutation. On the other hand, the extremal case of the Babai conjecture for S_n is composed of a cyclic permutation and the transformation which fixes all elements except two, which are neighbors in the cyclic order defined by the cyclic permutation. Therefore, one could wonder if a combination of these transformations could result in a DFA with both large reset threshold and full transition monoid.

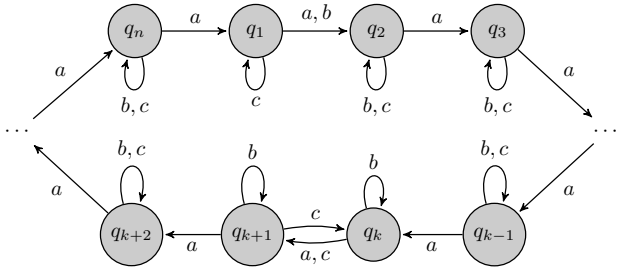


Figure 5.5: The automaton $\mathscr{CB}_{n,k}$.

The construction is defined as follows. There are n states $q_1, \ldots, q_n$ and three letters a, b, and c. The letter a acts as a cyclic permutation on the states, following their indices. The letter b fixes all states, except q_1, which is mapped to q_2 by b. The letter c fixes all states, except q_k and q_{k+1}, for some k, which are swapped by c. The resulting automaton $\mathscr{CB}_{n,k}$ is shown in Fig. 5.5. We notice that if we remove the letter c, we obtain the automaton $\mathscr{C}_n$ from the Černý family providing the largest currently existing lower bound in the Černý problem, and if we remove the letter b, we obtain a generating set of

the group S_n providing the largest currently existing lower bound in the Babai problem for S_n. Also observe that in the case where $k = 2$, our automaton is nothing but Brzozowski's "Universal Witness" [Brz13] recognizing the most complex regular language, i.e., the language witnessing at once practically all tight lower bounds found for the state complexity of various operations with regular languages, see [Brz13, Theorem 6]. The next result shows that, however, the reset threshold of the automaton $\mathscr{CB}_{n,k}$ is upper-bounded by $O(n \log n)$, while, as we show later, among automata with full transition monoid there exist ones whose reset threshold is a quadratic function of their state number.

Theorem 5.3.1. *The automaton $\mathscr{CB}_{n,k}$ has a reset word of length at most $4n\lceil \log_2 n \rceil$.*

Proof. Recall that we aim to show that, for each k, the automaton $\mathscr{CB}_{n,k}$ has a reset word of length at most $4n\lceil \log_2 n \rceil$. It is easy to see that the word $b(cab)^{n-2}$ of length $3n - 5 < 4n\lceil \log_2 n \rceil$ resets the automaton $\mathscr{CB}_{n,1}$, so that we assume that $k > 1$ in the rest of the proof.

We construct a word w letter-by-letter in several rounds, starting with the empty word. The main parameter in our construction is the current image of the state set of $\mathscr{CB}_{n,k}$ under the action of the word constructed so far; let S stand for this image. (Thus, we have $S = \{q_1, \ldots, q_n\}$ at the beginning, and S becomes a singleton at the end of the process.) It is quite helpful to visualize S as the set whose states bear certain tokens. If one colors states covered by tokens light-gray, then Fig. 5.5 represents the initial position while Fig. 5.6 shows some intermediate situation. When a letter $x \in \{a, b, c\}$ is applied to

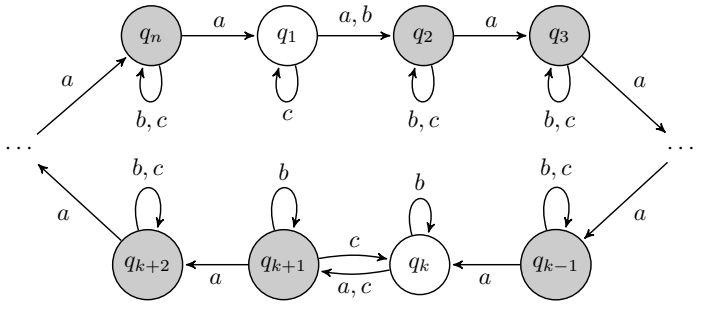


Figure 5.6: Tokens mark a subset in $\mathscr{CB}_{n,k}$.

S, the token that covers q_i, say, moves to the state $q_i \cdot x$; in more visual terms, the token "slides" along the arrow representing the transition $q_i \to q_i \cdot x$. If two

tokens arrive at the same state, which happens whenever both q_1 and q_2 bear tokens and the letter b is applied, we remove one of the tokens.

In the course of our construction, rounds of two sorts alternate: *merging*, in which only a's and b's are applied to S, and *pairing*, in which only a's and c's are applied. We call a state from S *isolated* if both its neighbor states (with respect to the cyclic order defined by a) are not in S (see Fig. 5.7). A merging round starts whenever $|S| > 1$ and S has at most one isolated state, and it lasts while S contains non-isolated states; a pairing round starts whenever $|S| > 1$ and all states in S are isolated, and it lasts while S contains more than one isolated state. Every round consists of a number of steps, in each of which we choose a letter, append the chosen letter to the word w and update the set S by applying the letter to it. The choice is done according to one of the two following rules (M) and (P) used during merging and pairing rounds, respectively:

(M) b is chosen whenever $q_1, q_2 \in S$; otherwise a is chosen;

(P) c is chosen whenever $q_{k+1} \in S$, but $q_k, q_{k+2} \notin S$ (so that q_{k+1} is isolated); otherwise a is chosen.

Clearly, at the beginning no state is isolated, and hence, the first round of our construction must be merging. It amounts to an immediate calculation to see that by the end of the first round, we have $w = b(a^2b)^{\lfloor \frac{n-1}{2} \rfloor}$ and $S = \{q_2, q_4, \dots q_{2\lfloor \frac{n}{2} \rfloor}\}$.

Now we are going to verify two claims.

Claim 1. If $|S| = m$ before any of the next merging rounds, then $|S| = \lceil \frac{m}{2} \rceil$ at the end of the round.

We say that two neighbor states $q_\ell, q_\ell{\cdot}a \in S$ form an *isolated couple* if each of these states has exactly one neighbor in S (see Fig. 5.7).

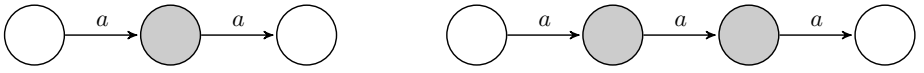


Figure 5.7: An isolated state and an isolated couple.

Claim 2. If $|S| = m$ before a pairing round, then at the end of the round S is partitioned in either $\frac{m}{2}$ isolated couples (if m is even) or $\frac{m-1}{2}$ isolated couples and one isolated state (if m is odd).

First we prove Claim 2. The *distance from* q_i *to* q_j is $\min\{d \in \mathbb{N} \mid q_i{\cdot}a^d = q_j\}$. We order tokens that cover the states in S according to the distance from

their states to the state q_{k+1}: the i-th token is the one that covers the state with the distance d_i to q_{k+1}, where $0 \le d_1 < d_2 < \cdots < d_m < n$. Now consider the evolution of the set S under the choice of letters according to the rule (P). Clearly, the first d_1 choices are all a's. After that the first token reaches q_{k+1}. Since the action of a translates the set S, without affecting distances between its states, all states in S remain isolated at this point. In particular, q_{k+1} is isolated, and hence, (P) forces the letter c to be applied. This moves the first token "backwards" to the state q_k while all other tokens keep their positions. The next letter to be applied is a, and its application moves all tokens one step "forwards" so that the token from q_k returns to q_{k+1}. Clearly, the distance from the state that holds the second token to q_{k+1} becomes $d_2 - d_1 - 1$ after these two moves. If the state q_{k+1} remains isolated, another application of c is invoked, followed by another application of a, and this results in a further decrement of the distance from the state that holds the second token to q_{k+1}. Eventually, after the suffix $a^{d_1}(ca)^{d_2-d_1}$ is appended to w, the second token reaches the state q_k. At this moment, the third token (if it exists) covers a state with distance $d_3 - d_2 > 1$ to q_k whence q_{k+1}, q_k form an isolated couple in S. The two tokens covering these states will then remain adjacent till the end of the round.

If $m = 2$ or $m = 3$, we are done. If $m > 3$, we proceed in the same way. Namely, the next $d_3 - d_2$ choices are all a's. After that the third token reaches q_{k+1}. Except the first two, all other tokens remain isolated. Now (P) forces c and a to be alternatively chosen $d_4 - d_3$ times each. This makes the third token shuffle between q_{k+1} and q_k, while the fourth and the next tokens move $d_4 - d_3$ steps "forwards". After that q_{k+1}, q_k form yet another isolated couple in S, etc.

We have shown that at the end of the round, the set S indeed consists of either $\frac{m}{2}$ isolated couples (if m is even) or $\frac{m-1}{2}$ isolated couples and one isolated state (if m is odd). Moreover, the suffix appended to w during the round is of the form

$$a^{d_1}(ca)^{d_2-d_1}a^{d_3-d_2}(ca)^{d_4-d_3}a^{d_5-d_4}(ca)^{d_6-d_5}\cdots . \tag{5.1}$$

The letter a occurs in this suffix d_m times if m is even and d_{m-1} times if m is odd, and the number of occurrences of c is less than that of a. Since $d_{m-1} < d_m < n$, we conclude that the length of the suffix (5.1) is less than $2n$.

Now it is easy to prove Claim 1. In view of Claim 2, at the beginning of the round, the set S consists of either $\frac{m}{2}$ isolated couples (if m is even) or $\frac{m-1}{2}$ isolated couples and one isolated state (if m is odd). If $\{q_\ell, q_\ell \cdot a\}$

is an isolated couple, we say that q_ℓ is its *left state*. Now we order isolated couples in S according to the distance from their left states to the state q_1: the i-th couple is the one with the distance d_i from its left state to q_1, where $0 \leq d_1 < d_2 < \cdots < d_{\lceil \frac{m}{2} \rceil} < n$. Consider the evolution of the set S under the choice of letters according to the rule (M). The first d_1 choices are all a's. After that the tokens that initially covered the states of the first isolated couple arrive at the states q_1 and q_2, and hence, (M) forces the letter b to be applied. This application removes the token from q_1 and does not change anything else. The state q_2 then becomes isolated. The next $d_2 - d_1$ choices are again all a's, and the successive applications of these a's bring tokens that initially covered the states of the second isolated couple to the states q_1 and q_2. Then, again, b is chosen, removing the token from q_1 and creating yet another isolated state in S, etc. At the end of the round, exactly one token from each isolated couple is removed and all remaining states are isolated. The number of these states is $\frac{m}{2}$ if m is even or $\frac{m+1}{2}$ if m is odd; in short, $\lceil \frac{m}{2} \rceil$, as claimed.

Moreover, the suffix appended to w during the round is of the form

$$a^{d_1} b a^{d_2 - d_1} b \cdots a^{d_{\lceil \frac{m}{2} \rceil} - d_{\lceil \frac{m}{2} \rceil - 1}} b. \tag{5.2}$$

The letter a occurs in this suffix $d_{\lceil \frac{m}{2} \rceil} < n$ times and the letter b occurs $\lceil \frac{m}{2} \rceil < n$ times, whence the length of the suffix (5.2) is less than $2n$.

Claim 1, together with the observation we made about the first merging round, readily implies that the number of merging rounds is at most $\lceil \log_2 n \rceil$. Since merging and pairing rounds alternate, the total number of rounds is upper-bounded by $2\lceil \log_2 n \rceil$. As observed after the proofs of Claims 1 and 2, a suffix of length less than $2n$ is appended to the current word w during each round. Clearly, at the end of the process, w becomes a reset word for $\mathscr{CB}_{n,k}$, and by the construction the length of w is less than $2n \cdot 2\lceil \log_2 n \rceil = 4n\lceil \log_2 n \rceil$. □

2.2. Random Sampling and Exhaustive Search. Every DFA with the transition monoid T_n necessarily has permutation letters that generate the whole symmetric group S_n and a letter of rank $n-1$ (i.e., a letter whose image has $n-1$ elements). It is a well known fact that the converse is also true, i.e., the transition monoid of any automaton with permutation letters generating S_n and a letter of rank $n-1$ is equal to T_n, see, e.g., [GM09, Theorem 3.1.3].

Relying on a group-theoretic result by Dixon [Dix69], Cameron [Cam13] observed that an automaton formed by two permutation letters taken uniformly

at random and an arbitrary non-permutation letter is synchronizing with high probability. We give an extension by using another non-trivial group-theoretical result, namely, the following theorem by Friedman *et al.* [FJR+98]:

Theorem 5.3.2. *For every r and $d \geq 2$ there is a constant C such that for d permutations $\pi_1, \pi_2, \ldots, \pi_d$ of S_n taken uniformly at random, the following property F_r holds with probability tending to 1 as $n \to \infty$: for any two r-tuples of distinct elements in $\{1, 2, \ldots, n\}$, there is a product of less than $C \log n$ of the π_i's which maps the first r-tuple to the second.*

Corollary 5.3.3. *There is a constant C such that the reset threshold of an n-state automaton with two random permutation letters and an arbitrary non-permutation letter does not exceed $Cn \log n$ with probability that tends to 1 as $n \to \infty$.*

Proof. Let $\mathscr{A} = \langle Q, \Sigma, \delta \rangle$ stand for the automaton in the formulation of the corollary. We let $a \in \Sigma$ be the non-permutation letter and assume that the two permutation letters in Σ have the property F_2 of Theorem 5.3.2 for $r = 2$ with some constant C. By Theorem 5.3.2 this assumption holds true with probability that tends to 1 as $n \to \infty$.

There exists two different states $q_1, q_2 \in Q$ such that $q_1 \cdot a = q_2 \cdot a$. The set $Q \cdot a$ contains less than n elements. If $|Q \cdot a| = 1$, then a is a reset word for $\mathscr{A}$. If $|Q \cdot a| > 1$, take two different states $p_1, p_2 \in Q \cdot a$. By F_2, there is a product w of less than $C \log n$ of the permutation letters such that $p_i \cdot w = q_i$ for $i = 1, 2$. Now $|Q \cdot awa| < |Q \cdot a|$. If $|Q \cdot awa| = 1$, awa is a reset word for $\mathscr{A}$. If $|Q \cdot awa| > 1$, we apply the same argument to a pair of different states in $Q \cdot awa$. Clearly, the process results in a reset word in at most $n - 1$ steps while the suffix appended at each step is of length at most $C \log n + 1$. Hence the length of the reset word constructed this way is at most $(C + 1)n \log n$. □

Corollary 5.3.3 indicates that one can hardly discover an n-state automaton with the transition monoid equal to T_n and sufficiently large reset threshold by a random sampling. Therefore, we performed an exhaustive search among all automata with two permutation letters generating S_n and one letter of rank $n - 1$. Our computational results are summarized in Table 1.

Number of states	2	3	4	5	6	7
Reset threshold	1	4	8	14	19	27

Table 1: The largest reset thresholds of n-state automata two permutation letters generating S_n and one letter of rank $n - 1$

As n grows, the reset thresholds of the obtained examples become much smaller than $(n-1)^2$. We were unable to derive a series of n-state three-letter automata with the transition monoid T_n and quadratically growing reset thresholds. We suspect that the reset threshold of automata in this class is $o(n^2)$.

In the case of unbounded alphabet, for every n, we present an n-state automaton $\mathscr{V}_n$ with the transition monoid T_n such that $\mathrm{rt}(\mathscr{V}_n) = \frac{n(n-1)}{2}$. The state set of $\mathscr{V}_n$ is $Q_n = \{q_0, \ldots, q_{n-1}\}$ and the input alphabet consists of n letters $a_1, \ldots, a_n$. The transition function is defined as follows:

$$\begin{cases} q_i{\cdot}a_j = q_i & \text{for } 0 \le i < n,\ 1 \le j < n,\ i \ne j,\ i \ne j+1,\ j \ne n, \\ q_i{\cdot}a_i = q_{i-1} & \text{for } 0 < i \le n-1, \\ q_i{\cdot}a_{i+1} = q_{i+1} & \text{for } 0 \le i < n-1, \\ q_0{\cdot}a_n = q_1{\cdot}a_n = q_0, & \\ q_i{\cdot}a_n = q_i & \text{for } 2 \le i \le n-1. \end{cases}$$

Simply speaking, every letter a_i for $i \le n-1$ swaps the states q_i and q_{i-1} and fixes the other states. The letter a_n brings both q_0 and q_1 to q_0 and fixes the other states. The automaton $\mathscr{V}_5$ is depicted in Fig. 5.8.

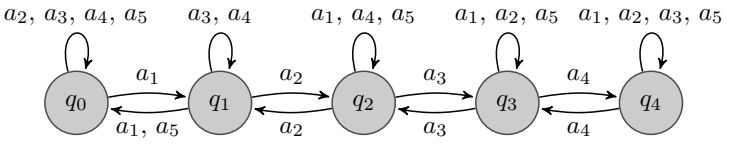


Figure 5.8: The automaton $\mathscr{V}_5$.

Recall that a state z of an DFA is said to be a *sink state* (or *zero*) if $z{\cdot}a = z$ for every input letter a. It is known that every n-state synchronizing automaton with zero can be reset by a word of length $\frac{n(n-1)}{2}$, cf. [Rys97]. To show that this upper bound is tight for each n, Rystsov [Rys97] constructed an n-state and $(n-1)$-letter synchronizing automaton $\mathscr{R}_n$ with zero which cannot be reset by any word of length less than $\frac{n(n-1)}{2}$. In fact, our automaton $\mathscr{V}_n$ is a slight modification of $\mathscr{R}_n$ as the latter automaton is nothing but $\mathscr{V}_n$ without the letter a_1.

Theorem 5.3.4. *For every n, the automaton $\mathscr{V}_n$ has T_n as its transition monoid and $\mathrm{rt}(\mathscr{V}_n) = \frac{n(n-1)}{2}$.*

Proof. The letters $a_1, \ldots, a_{n-1}$ generate S_n because the product $a_1 \cdots a_{n-1}$ is a full cycle and any full cycle together with any transposition generates S_n.

Since the letter a_n has rank $n-1$, it together with $a_1, \ldots, a_{n-1}$ generates T_n.

The automaton $\mathscr{V}_n$ is synchronizing because so is the restricted automaton $\mathscr{R}_n$, and $\mathrm{rt}(\mathscr{V}_n) \leq \frac{n(n-1)}{2}$ because every reset word for $\mathscr{R}_n$ resets $\mathscr{V}_n$ as well. It remains to verify that the length of any reset word for $\mathscr{V}_n$ must be at least $\frac{n(n-1)}{2}$. Let w be a reset word of minimum length for $\mathscr{V}_n$. Since a_n is the only non-permutation letter, we must have $w = w'a_n$ for some w' such that $|Q_n{\cdot}w'| > 1$. This is only possible when $Q_n{\cdot}w' = \{q_0, q_1\}$ whence $Q_n{\cdot}w = \{q_0\}$. Consider the function f from the set of all non-empty subsets of Q_n into the set of non-negative integers defined as follows: if $S = \{q_{s_1}, \ldots, q_{s_t}\}$, then $f(S) = \sum_{i=1}^{t} s_i$. Clearly, $f(\{q_0\}) = 0$ and $f(Q_n) = \frac{n(n-1)}{2}$. For any set S and any letter a_j, we have $f(S{\cdot}a_j) \geq f(S) - 1$ since each letter only exchanges two adjacent states or maps q_1 and q_0 to q_0. Thus, when we apply the word w letter-by-letter, the value of f after the application of the prefix of w of length i cannot be less than $\frac{n(n-1)}{2} - i$. Hence, to reach the value 0, we need at least $\frac{n(n-1)}{2}$ letters. □

2.3. Upper Bound on the Reset Threshold. We now provide a quadratic upper bound on the reset words of automata with the transition monoid equal to T_n. Our proof is inspired by the method of Rystsov [Rys00] adapted to our case.

Let $\mathscr{A} = \langle Q, \Sigma, \delta \rangle$ be a DFA. Given a proper non-empty subset $R \subset Q$ and a word w over Σ, we say that R *can be extended by* w if the cardinality of the set $Rw^{-1} = \{q \in Q \mid q{\cdot}w \in R\}$ is greater than $|R|$. Now assume that $|Q| = n$ and the transition monoid of $\mathscr{A}$ coincides with the full transformation monoid T_n. Then there is a letter x of rank $n-1$. The set $Q \setminus Q{\cdot}x$ consists of a unique state, which is called the *excluded state* for x and is denoted by $\mathrm{excl}(x)$. Furthermore, the set $Q{\cdot}x$ contains a unique state p such that $p = q_1{\cdot}x = q_2{\cdot}x$ for some $q_1 \neq q_2$; this state p is called the *duplicate state* for x and is denoted by $\mathrm{dupl}(x)$. We notice that a non-empty subset $R \subset Q$ can be extended by x if and only if $\mathrm{dupl}(x) \in R$ and $\mathrm{excl}(x) \notin R$. Moreover, if a word w is a product of permutation letters, R can be extended by the word wx if and only if $\mathrm{dupl}(x) \in Rw^{-1}$ and $\mathrm{excl}(x) \notin Rw^{-1}$. To better understand which extensions are possible, we construct a series of directed graphs (digraphs) $\Gamma_{1,i}$, $i = 0, 1, \ldots$, with the set Q as the vertex set.

The digraph $\Gamma_{1,0}$ has the set $E_0 = \{(\mathrm{excl}(x), \mathrm{dupl}(x))\}$ as its edge set. Let Π be the set of permutation letters of $\mathscr{A}$. Notice that Π generates the symmetric group S_n. By $\Pi^{\leq i}$ we denote the set of words of length at most i over the letters

in Π. The digraph $\Gamma_{1,i}$ for $i > 0$ has the edge set $E_i = \{(\mathrm{excl}(x)\cdot w, \mathrm{dupl}(x)\cdot w) \mid w \in \Pi^{\leq i}\}$. The digraphs $\Gamma_{1,i}$, $i = 0, 1, \ldots$, form a sort of stratification for the graph Γ_1 with the edge set $E = \cup_{i=0}^{\infty} E_i$; the latter digraph has been studied in [Rys00] and [BV16] (in the context of arbitrary completely reachable automata). Observe that none of the digraphs $\Gamma_{1,i}$, $i = 0, 1, \ldots$, have loops.

Recall that a digraph is said to be *strongly connected* if for every pair of its vertices, there exists a directed path from the first vertex to the second. We need the two following lemmas.

Lemma 5.3.5. *If the digraph $\Gamma_{1,k}$ is strongly connected, then every proper non-empty subset in Q can be extended by a word of length at most $k + 1$.*

Proof. Recall that Q is the set of states of $\Gamma_{1,k}$. Let R be a proper non-empty subset in Q. If $\Gamma_{1,k}$ is strongly connected, there exists an edge $(q, p) \in E_k$ that connects $Q \setminus R$ and R in the sense that $q \in Q \setminus R$ while $p \in R$. As $(q, p) \in E_k$, there exists a word $w \in \Pi^{\leq k}$ such that $(q, p) = (\mathrm{excl}(x)\cdot w, \mathrm{dupl}(x)\cdot w)$. Then $\mathrm{dupl}(x) \in Rw^{-1}$ and $\mathrm{excl}(x) \notin Rw^{-1}$, whence the word xw extends R and has length at most $k + 1$. □

Lemma 5.3.6. *The digraph $\Gamma_{1,2n-3}$ is strongly connected.*

Proof. We start with showing that the digraph $\Gamma_{1,n-1}$ contains an oriented cycle.

Consider the underlying digraph Δ of the automaton $\langle Q, \Pi, \delta|_{Q\times\Pi}\rangle$, i.e., the digraph with the vertex set Q and the edge set $\{(q, q\cdot a) \mid q \in Q, a \in \Pi\}$. This digraph is strongly connected since Π generates the whole symmetric group S_n. Therefore, for every $q \in Q\cdot x$, there exists a directed path in Δ from $\mathrm{excl}(x)$ to q. If one takes such a path $\mathrm{excl}(x) \xrightarrow{a_1} \cdots \xrightarrow{a_\ell} q$ of minimum length, it does not cross any vertex in Q more than once, whence the length ℓ of the path is at most $n - 1$. Thus, the word $u = a_1 \cdots a_\ell$ belongs to $\Pi^{\leq n-1}$ and the pair $(\mathrm{excl}(x)\cdot u, \mathrm{dupl}(x)\cdot u)$ is an outgoing edge of the vertex $q = \mathrm{excl}(x)\cdot u$ in the digraph digraph $\Gamma_{1,n-1}$. We see that every state in $Q\cdot x$ has an outgoing edge in $\Gamma_{1,n-1}$. Now, we can walk along the edges of $\Gamma_{1,n-1}$, starting at $\mathrm{excl}(x)$, which has the outgoing edge $(\mathrm{excl}(x), \mathrm{dupl}(x))$, until we reach an already visited state, thus getting an oriented cycle in the graph.

If $\Gamma = (V, E)$ is a digraph, we say that a vertex $v' \in V$ is *reachable* from a vertex $v \in V$ if either $v' = v$ or there is a directed path from v to v'. The mutual reachability relation is an equivalence on the set V, and the digraphs induced on the classes of the mutual reachability relation are either strongly connected or singletons (i.e., digraphs with 1 vertex and no edge). Slightly

abusing terminology, we call these induced digraphs (including singletons) the *strongly connected components* of the digraph Γ.

Consider the strongly connected components of the digraph $\Gamma_{1,n-1}$ and let $C_1, \ldots, C_m$ denote their vertex sets. Without any loss we may assume that $|C_1| \geq |C_2| \geq \cdots \geq |C_m|$. Observe that $m < n$ since $\Gamma_{1,n-1}$ contains an oriented cycle which is not a loop whence at least one strongly connected component is non-singleton. (Recall that digraphs of the form $\Gamma_{1,i}$ are loopless.) If $m = 1$, then already the digraph $\Gamma_{1,n-1}$ is strongly connected, and we are done. Otherwise we analyze the evolution of the partition of $\Gamma_{1,k}$ with $k \geq n-1$ into strongly connected components under the action of the letters in Π. Since Π generates the symmetric group S_n, it cannot preserve any non-trivial partition of Q. Thus, there is a non-singleton component C among $C_1, \ldots, C_m$ and a letter a in Π whose action sends two elements of C to different components, i.e. $C{\cdot}a \cap C_i \neq \varnothing$ and $C{\cdot}a \cap C_j \neq \varnothing$ for some $C_i \neq C_j$.

By the definition of the sets E_k, if $(p, q) \in E_{n-1}$, then $(p{\cdot}a, q{\cdot}a) \in E_n$. Therefore each edge from E_{n-1} that connects some vertices in C_s, $s = 1, \ldots, m$, translates into an edge from E_n that connects the images of these vertices in $C_s{\cdot}a$. Therefore, the digraphs of $\Gamma_{1,n}$ induced on the sets $C_1{\cdot}a, \ldots, C_m{\cdot}a$ are either strongly connected or singletons. In particular, the digraph of $\Gamma_{1,n}$ induced on $C{\cdot}a$ is strongly connected. Since $C{\cdot}a \cap C_i \neq \varnothing$ and $C{\cdot}a \cap C_j \neq \varnothing$, the digraph of $\Gamma_{1,n}$ induced on the set $C{\cdot}a \cup C_i \cup C_j$ also is strongly connected. This implies that the number m' of strongly connected components in $\Gamma_{1,n}$ is less than m. If $\Gamma_{1,n}$ is not yet strongly connected, the same reasoning applied to its strongly connected components, shows that the number of strongly connected components in $\Gamma_{1,n+1}$ is less than m', etc.

Since at each step the number of strongly connected components is reduced at least by 1, we conclude that we reach a strongly connected digraph in at most $n-2$ steps. Therefore, $\Gamma_{1,2n-3}$ is strongly connected. □

Theorem 5.3.7. *Let $\mathscr{A}$ be an n-state automaton with the transition monoid equal to T_n. The reset threshold of $\mathscr{A}$ is at most $2n^2 - 6n + 5$.*

Proof. Let x be a letter of rank $n-1$ and $h = \mathrm{dupl}(x)$. We extend the set $\{h\}$ by x, getting a subset R_2 with $|R_2| \geq 2$. Lemmas 5.3.5 and 5.3.6 imply that proper non-empty subsets in Q can be extended by words of length at most $2n-2$. Starting with R_2, we extend subsets until we reach the full state set. Let u_i be the word of length at most $2n-2$ used for the i-th of these extensions and let m be the number of the extensions. Observe that

$m \leq n-2$. Clearly, the word $u_m \cdots u_1 x$ resets $\mathscr{A}$ and has the length at most $1 + (n-2)(2n-2) = 2n^2 - 6n + 5$. □

Remark 5.3.8. *Let $\mathscr{A} = \langle Q, \Sigma, \delta \rangle$ be an n-state DFA that has a letter of rank $n-1$, and let P be the subgroup of the symmetric group S_n generated by the permutation letters from Σ. Our proof of Theorem 5.3.7 actually works in the case if P is a* 2-transitive *group, that is, P acts transitively on the set of ordered pairs of Q.*

5.4 Conclusion

The first problem that we considered was the construction of the Γ_1-graph of an automaton, motivated by the work of Bondar and Volkov [BV16]. We added the concept of *root states* of 1-deficient letter to the already well studied excluded state and duplicate state. With these concepts in hand, we provided an analysis of the way 1-deficient words combine with each other and with permutations, and their influence on the Γ_1-graph. This analysis first allowed to build a polynomial time algorithm for constructing the Γ_1-graph, then to build a counterexample to Conjecture 5.2.1, and finally to prove a modified version of this conjecture, namely Theorem 5.2.11.

To conclude, we propose the following problem, which is the restriction of Conjecture 4.2.1 to completely reachable automata.

Problem 5.4.1. *Let $\mathscr{A} = (Q; \Sigma; \delta)$ be a n-state completely reachable automaton. For any $0 < k < n$ and any set $S \in Q$ of size k, is it true that there exists a word w such that $Qw = S$ and $|w| \leq n(n-k)$?*

A positive answer to this problem would prove the Černý conjecture for completely reachable automata.

The second problem we studied was the hybrid Babai–Černý problem, where the question is to find tight bounds on the reset threshold for automata with the full transition monoid. We presented a series of n-state automata $\mathscr{V}_n$ in this class with reset threshold equal to $\frac{n(n-1)}{2}$, thus establishing a lower bound for the problem, and found an upper bound with the same growth rate, namely, $2n^2 + o(n^2)$. We also described a series of n-state automata with diameter of the pair digraph equal to $\frac{n^2}{4} + o(n^2)$.

For follow-up work, one direction is to refine the bounds with respect to the constants that do not match yet. Also, a lower bound for the hybrid problem

using only three letters (generators) is of interest, since the number of letters of the presented family $\mathscr{V}_n$ is equal to the number of states.

Conclusion and Perspectives

In this thesis, we focused on problems arising in finite state systems concerning subsets of states. Our research was motivated by the Černý conjecture and the recent developments observed in this field. Our results were twofold. On the one hand, we solved open problems on synchronizing automata, by providing both positive and negative answers. On the other hand, we opened new research tracks and provided possible approaches to the Černý conjecture.

Results Summary

In Chapter 2, we first formalized the concept of triple rendezvous time as an intermediate step towards the Černý conjecture. Using the synchronizing probability function, a tool which allows one to represent the synchronization process within an automaton, we were able to prove an upper bound on the triple rendezvous time.

Then, by providing an infinite family of automata for which $T_3 = n+3$ (with n being the number of states of the automaton), we refuted Conjecture 2.3.3, formulated in [Jun12]. Conjecture 2.3.3 was stated as a tentative roadmap towards a proof of the Černý conjecture with the help of the synchronizing probability function, and in that sense our conterexample is a negative result towards that direction.

A straightforward question is how to narrow the gap between $n + 3$ and $n^2/(6.4...)$ for the triple rendezvous time. A natural continuation of this research would be to find non-trivial bounds for T_l, with $3 < l \leq n$ (i.e., the smallest number such that the set of reachable vectors includes a column of weight at least l).

In Chapter 3, we studied the diameter of the square graph of automata and of permutation sets. We first described a series of n-state automata with diameter of the square graph equal to $\frac{n^2}{4} + o(n^2)$. We then used this construction to improve a theorem on the joint spectral radius of matrix sets.

Second, we showed that the diameter of the square graph of a permutation set could not reach the bound $n(n-1)/2 - 1$. Extending this result to synchronizing automata would lead to an improvement of the best general bound on reset threshold of synchronizing automata.

In Chapter 4, we focused on subset reachability in synchronizing automata. The first problem considered was the upper bound on the length of the shortest word reaching some subsets, motivated by [Don16, Conjecture 18]. The second problem was the length of shortest word such that a precise state is not in its image.

We started by presenting a family of automata built from a set of permutations with a large square graph diameter. This family has subsets of $n-2$ states that cannot be reached by words shorter than a quadratic value, and therefore is a counterexample to Conjecture 4.2.1. Then, we analyzed two modified versions of Conjecture 4.2.1. Both cases led to negative answers. In particular, we built a family of strongly connected synchronizing automata with subsets that cannot be reached with words shorter than $\frac{2^n}{n}$.

In Chapter 5, the first problem that we considered was the construction of the Γ_1-graph of an automaton, motivated by the work of Bondar and Volkov [BV16]. We added the concept of *root states* of 1-deficient letter to the already well studied excluded state and duplicate state. With these concepts in hand, we provided an analysis of the way 1-deficient words combine with each other and with permutations, and their influence on the Γ_1-graph. This analysis first allowed to build a polynomial time algorithm for constructing the Γ_1-graph, then to build a counterexample to Conjecture 5.2.1, and finally to prove a modified version of this conjecture, namely Theorem 5.2.11.

The second problem considered was the hybrid Babai–Černý problem, where the question is to find tight bounds on the reset threshold for automata generating the full transition monoid. We first presented a series of n-state automata $\mathscr{V}_n$ in this class having a reset threshold equal to $\frac{n(n-1)}{2}$, thus establishing a lower bound for the problem. Second, we proved an upper bound with the same growth rate, namely, $2n^2 + o(n^2)$.

For follow-up work, one direction is to refine the bounds with respect to the constants. Also, a lower bound for the hybrid problem using only three letters

(generators) is of interest, since the number of letters of the presented family $\mathscr{V}_n$ is equal to the number of states, and three letters is the minimal number of letters such that generating T_n is possible.

Perspectives

For open perspectives, we would like to recall the three main open problems which were stated in this thesis.

The first one is the restriction of Conjecture 4.2.1 to completely reachable automata:

Problem 5.4.2. *Let $\mathscr{A} = (Q; \Sigma; \delta)$ be a n-state completely reachable automaton. For any $0 < k < n$ and any set $S \in Q$ of size k, is it true that there exists a word w such that $Qw = S$ and $|w| \leq n(n-k)$?*

A positive answer to this problem would prove the Černý conjecture for completely reachable automata.

Second, we conjectured that the square graph diameter of a set of permutations was at most of the order of $O(n^2/4)$:

Conjecture 5.4.3. *The diameter of the pair digraph for a subset of S_n is bounded above by $\frac{n^2}{4} + o(n^2)$.*

And finally, we have the excluding state problem:

Problem 5.4.4. *Let Q be the set of states of a synchronizing strongly connected $n-$state DFA.*

Is there a constant c such that, for any state $q \in Q$, there exists a word w of length not greater than cn such that $q \notin Qw$?

A positive answer to this problem would directly imply a $O(7n^3/48)$ bound on Conjecture 1.2.2.

Final Word

We hope that our results show that the analysis of subsets of states and their images can provide additional insight on longstanding problems such as Conjecture 1.2.2. Although this conjecture is still open, we believe that our research, by closing doors and opening new ones, will provide helpful guidance to future research.

Appendices

A.1 Values of the Function N for the Pairs of $SG(\mathscr{F}_n)$

For $n > 7$, $n \equiv 3 \pmod 4$, the values of the function N used in the proof of Theorem 3.2.1 are provided in the two lists below. Some of the formulas in the lists involve one or two positive integer parameters denoted by m and m'. We always use m' for the index of the first state of a pair and m for the index of the second state; we do not specify the ranges of these parameters as they should be clear from the context. We use the following conventions: $k = \frac{n-5}{2}$, $N_1 = \frac{k+3}{2}$, $N_2 = (k+4)(k-1)$.

Our first list contains the values of N for the pairs that involve one or two of the "central" states q_1, q_2, q_3, q_4 of the automaton $\mathscr{F}_n$ or one or two of its "extreme" states q_{2k+4} and q_{2k+5}. (Our terminology follows the pictorial presentation of $\mathscr{F}_n$ in Fig. 3.3.)

- $N(q_1q_2) = N_1 + N_2 + 2k + 1$;
- $N(q_1q_3) = N_1 + N_2 + 4k + 7$;
- $N(q_1q_4) = N_1 + N_2 + 2k + 2$;
- $N(q_1q_{4m+1}) = \begin{cases} \frac{k+3}{2} & \text{if } m = N_1 - 1, \\ N_1 + (k+4)(k-2m-1) + 2k + 3 & \text{otherwise;} \end{cases}$
- $N(q_1q_{4m+2}) = N_1 + (k+4)(k-2m-1) + 2k - 2m + 2$;
- $N(q_1q_{4m+3}) = \begin{cases} N_1 + N_2 + 2k + 6 & \text{if } m = 1, \\ N_1 + (k+4)(k-2m+1) + 2k + 8 & \text{otherwise;} \end{cases}$
- $N(q_1q_{4m+4}) = N_1 + (k+4)(k-2m+1) + 2k - 2m + 3$;

- $N(q_1q_{2k+4}) = N_1 + k + 2$;
- $N(q_1q_{2k+5}) = N_1 + 2k + 8$;

- $N(q_2q_3) = N_1 + N_2 + 2k + 4$;
- $N(q_2q_4) = N_1 + N_2 + 4k + 8$;
- $N(q_2q_{4m+1}) = \begin{cases} N_1 + N_2 + 2k + 5 & \text{if } m = 1, \\ N_1 + (k+4)(k-2m+1) + 2k + 7 & \text{otherwise;} \end{cases}$
- $N(q_2q_{4m+2}) = N_1 + (k+4)(k-2m+1) + 2k - 2m + 2$;
- $N(q_2q_{4m+3}) = N_1 + (k+4)(k-2m-1) + 2k + 6$;
- $N(q_2q_{4m+4}) = N_1 + (k+4)(k-2m-1) + 2k - 2m + 1$;
- $N(q_2q_{2k+4}) = N_1 + k + 1$;
- $N(q_2q_{2k+5}) = N_1 - 1$;

- $N(q_3q_4) = N_1 + N_2 + 2k + 3$;
- $N(q_3q_{4m+1}) = \begin{cases} \frac{k-1}{2} & \text{if } m = N_1 - 1, \\ N_1 + (k+4)(k-2m-1) + 2k + 5 & \text{otherwise;} \end{cases}$
- $N(q_3q_{4m+2}) = N_1 + (k+4)(k-2m-1) + 2k - 2m + 4$;
- $N(q_3q_{4m+3}) = \begin{cases} N_1 + N_2 + 2k + 5 & \text{if } m = 1, \\ N_1 + (k+4)(k-2m+1) + 2k + 6 & \text{otherwise;} \end{cases}$
- $N(q_3q_{4m+4}) = N_1 + (k+4)(k-2m+1) + 2k - 2m + 1$;
- $N(q_3q_{2k+4}) = N_1 + k$;
- $N(q_3q_{2k+5}) = N_1 + 2k + 6$;

- $N(q_4q_{4m+1}) = \begin{cases} N_1 + N_2 + 2k + 4 & \text{if } m = 1, \\ N_1 + (k+4)(k-2m+1) + 2k + 5 & \text{otherwise;} \end{cases}$

- $N(q_4 q_{4m+2}) = N_1 + (k+4)(k-2m+1) + 2k - 2m + 4$;
- $N(q_4 q_{4m+3}) = N_1 + (k+4)(k-2m-1) + 2k + 4$;
- $N(q_4 q_{4m+4}) = N_1 + (k+4)(k-2m-1) + 2k - 2m + 3$;
- $N(q_4 q_{2k+4}) = N_1 + k + 3$;
- $N(q_4 q_{2k+5}) = N_1 + 1$;

- $N(q_{4m'+1} q_{2k+4}) = \begin{cases} N_1 + N_2 + 3k + 7 & \text{if } 2m' = k+1, \\ N_1 + (k+4)(2m') + k + 1 & \text{otherwise;} \end{cases}$
- $N(q_{4m'+1} q_{2k+5}) =$
  $\begin{cases} N_1 + (k+4)(2m'-2) + 2k + 4 + 2m' & \text{if } m' = N_1 - 1, \\ N_1 + (k+4)(2m'-2) + 2k + 5 + 2m' & \text{otherwise;} \end{cases}$
- $N(q_{4m'+2} q_{2k+4}) = N_1 + (k+4)2m' + k + 1 + 4m'$;
- $N(q_{4m'+2} q_{2k+5}) =$
  $\begin{cases} N_1 + 2k + 9 & \text{if } m' = 1, \\ N_1 + N_2 + 2k + m' + 8 & \text{if } 2m' = k+1, \\ N_1 + (k+4)(2m'-2) + 2k + 2m' + 7 & \text{otherwise;} \end{cases}$
- $N(q_{4m'+3}, q_{2k+4}) = N_1 + (k+4)(2m') + k$;
- $N(q_{4m'+3} q_{2k+5}) = \begin{cases} N_1 + (k+4)(2m') + 2k + 2m' + 5 & \text{if } 2m' = k-1, \\ N_1 + (k+4)(2m') + 2k + 2m' + 6 & \text{otherwise;} \end{cases}$
- $N(q_{4m'+4} q_{2k+4}) = N_1 + (k+4)2m' + k + 2 + 4m'$;
- $N(q_{4m'+4}, q_{2k+5}) = \begin{cases} N_1 + N_2 + 3k + 5 & \text{if } 2m' = k-1, \\ N_1 + (k+4)(2m') + 2k + 2m' + 8 & \text{otherwise;} \end{cases}$
- $N(q_{2k+4} q_{2k+5}) = N_1 + N_2 + 3k + 6$.

Our second list contains the values of N for the remaining pairs. In addition to our earlier conventions, we also use $M = m + m'$ and $M' = m - m'$ here.

- $N(q_{4m'+1} q_{4m+1}) = N_1 + (k+4)(k - 2M' + 1) + 2k + 2m' + 5$;

- $N(q_{4m'+1}q_{4m+2}) =$
$$\begin{cases} N_1 + N_2 + 4k + 8 - 2m & \text{if } m' = m, \\ N_1 + (k+4)(k - 2M' + 1) + 2k - 2m + 2 & \text{otherwise;} \end{cases}$$

- $N(q_{4m'+1}q_{4m+3}) =$
$$\begin{cases} N_1 + (k+4)(k - 2M - 1)) + 2k + 2m' + 4 & \text{if } 2M < k+1, \\ \frac{4m-k-1}{2} & \text{if } 2M = k+1, \\ N_1 + (k+4)(2M - k - 3) + 2k + 2m' + 5 & \text{otherwise;} \end{cases}$$

- $N(q_{4m'+1}q_{4m+4}) =$
$$\begin{cases} N_1 + (k+4)(k - 2M - 1) + 2k - 2m + 3 & \text{if } 2M < k+1, \\ N_1 + 2m & \text{if } 2M = k+1, \\ N_1 + (k+4)(2M - k - 3) + 2k + 2m + 8 & \text{otherwise;} \end{cases}$$

- $N(q_{4m'+2}q_{4m+1}) =$
$$\begin{cases} N_1 + N_2 + 2k + 2m' + 5 & \text{if } m' = m - 1, \\ N_1 + (k+4)(k - 2M' + 1) + 2k + 2m' + 7 & \text{otherwise;} \end{cases}$$

- $N(q_{4m'+2}q_{4m+2}) = N_1 + (k+4)(k - 2M' + 1) + 2k - 2m + 4m' + 2;$

- $N(q_{4m'+2}q_{4m+3}) =$
$$\begin{cases} N_1 + (k+4)(k - 2M - 1) + 2k - 2m + 4 & \text{if } 2M < k+1, \\ N_1 + 2m' - 1 & \text{if } 2M = k+1, \\ N_1 + (k+4)(2M - k - 3) + 2k + 2m' + 7 & \text{otherwise;} \end{cases}$$

- $N(q_{4m'+2}q_{4m+4}) =$
$$\begin{cases} N_1 + (k+4)(k - 2M - 1) + 2k - 2m' + 1 & \text{if } 2M < k+1, \\ N_1 + k + 2m' + 1 & \text{if } 2M = k+1, \\ N_1 + (k+4)(2M - k - 1) + 4m' + 2m & \text{otherwise;} \end{cases}$$

- $N(q_{4m'+3}q_{4m+1}) =$
$$\begin{cases} N_1 + (k+4)(k - 2M - 1)) + 2k + 2m' + 5 & \text{if } 2M < k+1, \\ \frac{4m-k-3}{2} & \text{if } 2M = k+1, \\ N_1 + (k+4)(2M - k - 3) + 2k + 2m' + 6 & \text{otherwise;} \end{cases}$$

- $N(q_{4m'+3}q_{4m+2}) =$
$$\begin{cases} N_1 + (k+4)(k-2M+1) + 2k - 2m + 4 & \text{if } 2M < k+1, \\ N_1 + 2m - 1 & \text{if } 2M = k+1, \\ N_1 + (k+4)(2M-k-3) + 2k + 2m + 7 & \text{otherwise;} \end{cases}$$

- $N(q_{4m'+3}q_{4m+3}) =$
$$\begin{cases} N_1 + (k+4)(k-2M'+1) + 2k + 2m + 3 & \text{if } m = m'+1, \\ N_1 + (k+4)(k-2M'+1) + 2k + 2m' + 6 & \text{otherwise;} \end{cases}$$

- $N(q_{4m'+3}q_{4m+4}) =$
$$\begin{cases} N_1 + N_2 + 4k + 7 - 2m & \text{if } m' = m, \\ N_1 + (k+4)(k-2M'+1) + 2k - 2m + 1 & \text{otherwise;} \end{cases}$$

- $N(q_{4m'+4}q_{4m+1}) =$
$$\begin{cases} N_1 + (k+4)(k-2M-1) + 2k - 2m' + 3 & \text{if } 2M < k+1, \\ N_1 + 2m' & \text{if } 2M = k+1, \\ N_1 + (k+4)(2M-k-3) + 2k + 2m' + 8 & \text{otherwise;} \end{cases}$$

- $N(q_{4m'+4}, q_{4m+2}) =$
$$\begin{cases} N_1 + (k+4)(k-2M-1) + 2k - 2m' + 1 & \text{if } 2M < k+1, \\ N_1 + k + 2m' + 2 & \text{if } 2M = k+1, \\ N_1 + (k+4)(2M-k-1) + 4m + 2m' - 1 & \text{otherwise;} \end{cases}$$

- $N(q_{4m'+4}q_{4m+3}) =$
$$\begin{cases} N_1 + N_2 + 2k + 2m' + 6 & \text{if } m = m'+1, \\ N_1 + (k+4)(k-2M'+1) + 2k + 2m' + 8 & \text{otherwise;} \end{cases}$$

- $N(q_{4m'+4}q_{4m+4}) = N_1 + (k+4)(k-2M'+1) + 2k - 2m + 4m' + 3.$

It can be routinely verified that the function N defined this way satisfies Equation (3.1). Therefore the shortest word mapping q_2q_4 to $q_{k+2}q_{k+4}$ is at least $N(q_2q_4) - N(q_{k+2}q_{k+4})$ letters long. Unfolding the definition, we obtain

$$N(q_{k+2}q_{k+4}) = 0$$

and

$$N(q_2q_4) = N_1 + N_2 + 4k + 8 = \frac{k+3}{2} + (k+4)(k-1) + 4k + 8$$
$$= k^2 + \frac{15k}{2} + \frac{11}{2} = \frac{n^2}{4} + \frac{5n}{4} - 7.$$

A.2 Proofs on Complete Numbering of Permutations

A.2.1 One-letter Cycle of Size n

We start with the case where the largest cycle labelled by a single letter is a cycle with all the states, and $n > 3$. We show that the square graph is partitioned in cycles and that Lemma 3.3.11 applies.

Proposition A.2.1. *Let A be a permutation set with a strongly connected square graph, with $n > 3$ states, and a cyclic letter. Then a complete numbering of the square graph is impossible.*

Proof. Let letter a be the letter labelling a cycle in the base graph containing all the states. Then, the effect of a on the square graph is a partition in cycles, with at least two cycles. For example, in Fig. 9, letter a acts as a cycle, and defines the partition in cycles of the square graph shown in Fig. 10.

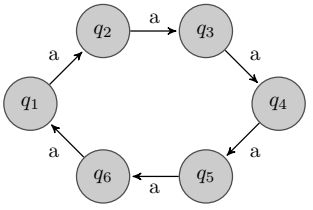


Figure 9: Letter a is a cyclic permutation in the base graph.

First, we notice that, in each cycle of the square graph, each basic state is represented the same number of times in pairs (once or twice). For example in Fig. 10, in the top cycle, q_1 is in the first and last pair and q_2 is in the second and third pair.

Second, as the square graph is strongly connected, there must be, for any cycle in the square graph, a letter that maps one of the pairs outside of that

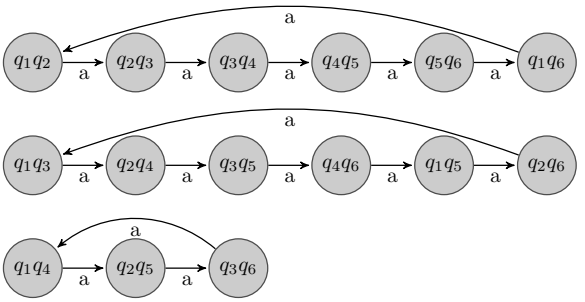


Figure 10: Letter a defines a partition of the pairs of the square graph in cycles.

cycle. Without loss of generality, let us assume letter b does this. Since b is a permutation, then $Qb = Q$ (the image of the set of all the states is the set of all the states). Therefore, applying the letter b to a set of pairs of the same cycle in the square graph, and which therefore have the same number of time each state, results in a set of pairs that also includes the same number of time every state.

Third, let us now assume that there is only exactly one outgoing edge labelled b from a cycle. This means that the $n-1$ or $n/2-1$ other pairs of the considered cycle stays in the cycle when letter b is applied to them.

However, since b is a permutation in the square graph, any two pairs that are different from each other are mapped on two different pairs by b. Therefore, the $n-1$ pairs which do not leave the cycle are still all different from each other after applying b. As they are all in the same cycle of pairs, the number of each state is represented the same number of times, except for the two states corresponding to the pair that was mapped outside of the cycle by b.

Since all the states must be represented the same number of time because $Qb = Q$, the two states of the pair which is mapped out of the the cycle must be the two states which are missing, and this pair is therefore equal to the last pair of the cycle. This is a contradiction to the fact that this pair was outside of the cycle. It therefore implies that there cannot be a single outgoing edge with letter b from the cycle.

As the square graph is strongly connected, there must be outgoing edges from every cycle of the square graph. As we proved that any letter with an outgoing edge from a cycle labels at least a second outgoing edge, Lemma 3.3.11 applies. A complete numbering is not possible if one of the letters labels a cycle with all the states. □

A.2.2 Letter labelling a largest cycle of length larger or equal to 5

We consider the case in which the largest cycle of the base graph identified by a single letter is of size m, with $5 \leq m < n$. Without loss of generality, let assume that the letter identifying that cycle is a. Let us write the states of the cycle $c_1, c_2, ..., c_m$ and call the set of states of the cycle C.the situation for $m = 5$.

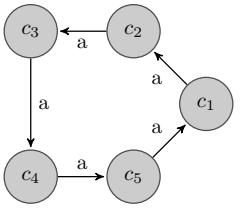


Figure 11: Cycle C of length 5.

Proposition A.2.2. *Let A be a permutation set with a strongly connected square graph, with $n > 6$ states, and a largest cycle labelled by a single letter of length m, with $5 \leq m \leq n$. Then a complete numbering of the square graph is impossible.*

Proof. Since the base graph is strongly connected, there must be a second letter mapping one of the states of C to a state not in C. Without loss of generality, let us name this letter b, the states c_1 and d_2. Therefore, we have $c_1 b = d_2$. The situation for a cycle of length 5 is presented in Fig. 12, and the effect on the square graph starting from pair $c_1 c_3$ is shown in Fig. 13.

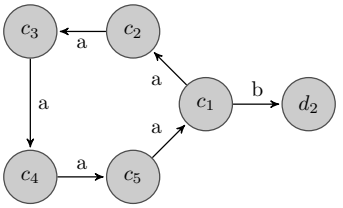


Figure 12: Cycle C of length 5, and an outgoing edge labelled b from c_1.

As b is also a permutation letter, the states c_1, d_2, $d_2 b$ ..., $c_1 b^{-1}$ are also states of a cycle with oriented edges identified by b. Let us call this cycle D,

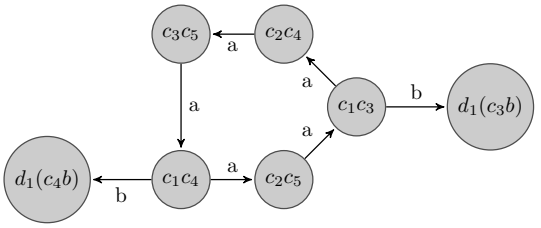


Figure 13: Square graph situation obtained from Fig. 12, starting from pair c_1c_3.

and name its states which are not in C d_2, d_3, etc. in that order. Depending on the way the cycles C and D share states, we obtain different situations. For each of them we prove that a complete numbering is impossible. The situations are the following:

1. Letter b maps at least two states from C outside of C.

2. Letter b maps one state outside of C, C and D have only one common state, c_1.

3. Letter b maps one state outside of C, C and D have more than one common states, and the common states are following each other in both cycles.

4. Letter b maps one state outside of C, C and D have more than one common states, and the common states are not all following each other in both cycles.

Case 1. In order to treat the first case, we consider the pairs of states of the type c_ic_{i+2} with $1 \leq i \leq m-2$. We notice that these pairs are in a cycle labelled by a in the square graph. The situation for $m = 5$ is shown in Fig. 13. If a second state c_k of C is mapped outside of C by a letter b, then both c_kc_{k+2} and c_kc_{k-2} (modulo m for $k+2$ and $k-2$) are mapped outside of the cycle by b. Since at least one of them is different from c_1c_3 and c_1c_{m-1}, there are at least three pairs in the square graph mapped outside of the cycle by b. Therefore Lemma 3.3.9 applies, and no complete numbering is possible.

Case 2. In the second case, the cycle D has only one common state with C, which is c_1. The situation is shown in Fig. 14 for $m = 5$ and a second cycle of length 3.

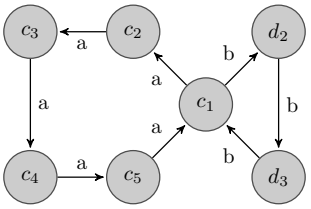


Figure 14: State c_1 is the only common state of the two cycles.

Let us consider the cycle of pairs P_1, containing all the pairs $(c_1c_3)b^i$, for $i > 0$ (this cycle was already represented in Fig. 13). It has two outgoing edges labelled b mapping pairs c_1c_3 and $c1c_{m-1}$ outside of it. Therefore, Lemma 3.3.10 applies, and either the cycle composed of the pairs $(c_1c_3)b^i$, for $i > 0$, should include $(c_1c_{m-2})a$ or the cycle $(c_1c_{m-2})b^i$, for $i > 0$, should include $(c_1c_3)a$. However, both cycles have either state c_1 or states from D in each of their pairs, while these states are neither in $(c_1c_{m-2})a$, neither in $(c_1c_3)a$, since $c_1a, c_{m-2}a, c_3a$ are all in C and different from c_1. Therefore a complete numbering is impossible.

Case 3. The third situation is when the cycle D has more than shard state with C, i.e. rejoins C at an other state than c_1, and then both letters have the same effect, up to c_1. This situation is represented in Fig. 15, with two common states.

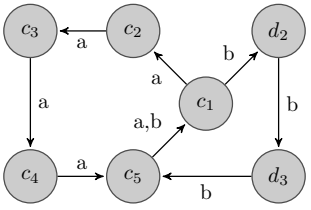


Figure 15: Some states are in both C and D, and the order is the same.

We notice that c_1 is the last common state of C and D, as $c_1a \neq c_1b$. The pair we consider in order to analyze the square graph is c_1c_m. Let us call P_1 the cycle composed of the pairs $(c_1c_m)a^i$, $i > 0$, and P_2 the cycle $(c_1c_m)b^i$, $i > 0$. Both cycles are of length at least 3. We notice that $(c_1c_m)a = c_1c_2$ and $(c_1c_m)b = c_1d_2$. The effect on the square graph is represented in Fig. 16.

First, to allow a complete numbering, c_1c_m must be the highest number of one of the cycles, and the lowest for the other one.

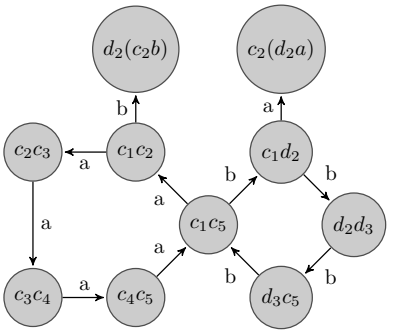


Figure 16: The effect on the graph of pairs.

Second, since (c_1c_2) and (c_1d_2) both follows c_1c_m in a cycle of at least three states, one of them must be a middle pair in its cycle, let call it p_1.

Third, $(c_1c_2)b = d_2(c_2b)$ and $(c_1d_2)a = c_2(d_2a)$ are outside of the two cycles we were considering, because d_2 is not in the first and c_2 is not in the second. For the one following p_1, this leads to a contradiction. Indeed, this pair cannot have the number of p_1 minus one, because this number is already taken by the pair following p_1 with the other letter. It can also not have a higher number because it would have to rejoin p_1 by applying consecutively the same letter and going through c_1c_m (to avoid to have twice the number $N(p_1) + 1$), but c_1c_m is in a independent cycle from p_1 with this letter. Therefore a complete numbering is impossible.

Case 4. In this case, two situations occurs: either there is an edge labelled by both a and b in C and D at the same time (i), or the effect of a and b on any state common to C and D is different (ii).

(i) If there is an edge labelled by both a and b, there exists a state c_k such that $c_ka = c_kb$ and $c_ka^2 = c_kba \neq c_kab = c_kb^2$.

Now, let us consider the pair c_1c_k. We have that this pair is in a cycle labelled by a of length at least 3, and a cycle labelled by b. If both lengths are more than 3, then either $(c_1c_k)a = c_2(c_ka)$ or $(c_1c_k)b = d_2(c_kb)$ are a middle pair in their cycles. However, again we have that $(c_1c_k)ab \neq (c_1c_k)a^2$ and $(c_1c_k)ba \neq (c_1c_k)b^2$. For the one which was a middle pair, it implies that a complete numbering is impossible.

The only situation where the second cycle in the pair graph is a swap is when the cycle is of length 4 in the base graph, but labels a swap in the square graph between c_1c_k and $d_2(c_kb)$ (in that case, $d_2(c_kb)$ is not a middle pair in the

cycle labelled by b, and the argument above does not apply directly). However, this situation implies that $d_2b = c_k$ and $c_kb^2 = c_1$. Recall that we chose c_k such that $c_ka = c_kb$, hence $c_kb = c_{k+1}$ and $c_ka^2 = c_kba \neq c_kab = c_kb^2$. This situation is shown in Fig 17.

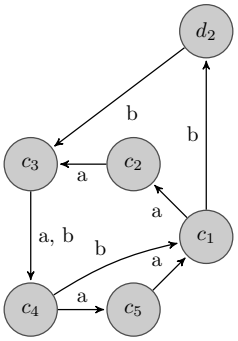


Figure 17: Case where letter b is a swap between the pair c_1c_k and d_2c_{k+1}. Here we have $k = 3$.

Now, we consider the cycle starting with the pair c_1c_2. We have that both c_1c_2 and c_1c_m are mapped outside of this cycle. Moreover, c_kc_{k+1} is in the same cycle, and therefore cannot be mapped outside of this cycle by b otherwise there would be three outgoing edges (it is the case in Fig 17). As $(c_kc_{k+1})b = c_1c_{k+1}$, we must have that $c_{k+1} = c_m$ to stay inside of the cycle. However, $c_{k+1}b = c_1$ and $c_ma = c_1$, while we chose c_k such that $c_{k+1}a \neq c_{k+1}b$. Therefore it is a contradiction and a complete numbering is also impossible in this case.

(ii) In this case, we assume that C and D have at least two common states, that only one edge labelled by a is outgoing from D (because $|D| \leq |C|$), that one edge labelled by b is outgoing from C, and that a and b have never the same effect on states (both in C and D). Such situation is shown in in Fig. 18, and the effect on the square graph is represented in Fig. 19.

□

We consider the pair c_1c_2 and name P_1 the cycle $(c_1c_2)a^i$, $i > 0$. We notice that both c_1c_2 and c_mc_1 are mapped outside of P_1 by b. Since they are also consecutive, c_1c_2 must have the highest number in P_1 and c_mc_1 the lowest one, and there should be a jump from c_1c_2 with letter b.

We notice that none of the other pairs than c_1c_m can be mapped on c_1c_2 by letter b, otherwise the jump in the second cycle cannot be at this position, as it would imply two increases of numbers in the cycle $(c_1c_2)b^i$, $i > 0$.

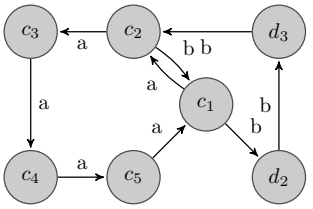


Figure 18: Some states are common to C and D, but not following the same order in both cycles.

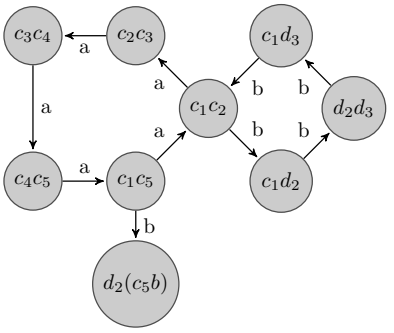


Figure 19: The effect on the graph of pairs.

Now, we know that all the states in C except for c_1 are mapped inside of C by letter b. This implies that the pair c_2c_3, which cannot be mapped outside of P_1 nor on c_1c_2, but has the second highest number of P_1, must be mapped either on c_3c_4 (the third highest number of P_1), or on itself by b:

- If c_2c_3 is mapped on c_3c_4, then c_3c_4 has to be either mapped on c_4c_5, which implies that letter a and b have the same effect on these states, a contradiction, or on c_2c_3 again, which would imply a self loop on state c_3 and a swap between c_2 and c_4. However in the latter case, c_4c_5 is mapped on $c_2(c_5b)$, which is either outside of the cycle or equal to c_1c_2, which we show was impossible, therefore it leads to a contradiction.

- If c_2c_3 is mapped on itself, then c_3c_4 can not be mapped on c_4c_5 and therefore must be also a self loop. If b labels self loops for the pair c_2c_3 and c_3c_4, it implies that b labels a self loop for c_3, and then also for c_2 and c_4. Extending this result to the following pairs gets that b can label only self loops on all the states, except for c_1. However, c_1b^i was supposed to

have more than one common state with C, a contradiction.

Therefore, a complete numbering is impossible. In the example in Fig. 19, the contradiction is directly obtained from pair c_2c_3, because $(c_2c_3)b = c_1(c_3b)$, which is different both from c_2c_3 and c_3c_4, whatever c_3b is.

A.2.3 One-letter cycles of length 3 or 4, and a self loop with the same letter

If the largest cycle is of length 4 and there is a self loop labelled with the same letter, then there exists an outgoing edge from the self loop labelled with a second letter, and we show that this letter labels at least three outgoing edges from a cycle in the square graph, hence Lemma 3.3.9 applies.

If there are at least 10 states and if the largest cycle is of length 3, then there are at least 4 self loops or swaps, or two cycles of length 3 and a self-loop. Therefore we can either be in the situation of Fig. 20, in which Lemma 3.3.9 applies, or refer to the case with cycles of length two and three simultaneously hereunder (Proposition A.2.6).

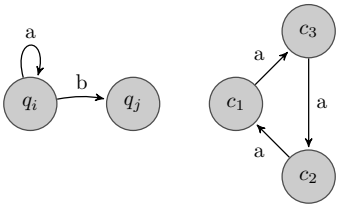


Figure 20: A self loop and a cycle of length 3 for letter a.

Proposition A.2.3. *Let A be a permutation set with a strongly connected square graph, with $n > 10$ states, a cycle of length 3 or 4 labelled by a single letter and a self loop labelled by the same letter. Then a complete numbering of the square graph is impossible.*

Proof. If a letter a is identifying a cycle C of length $l = 3$ or $l = 4$, and is identifying a self loop for a state outside of this cycle, let say q_i, then the pairs composed by this state and the states of the cycle are in a cycle of pairs of length l. As the graph is strongly connected, there must exists a letter b and a state q_j such that $q_ib = q_j$. Then, in the square graph, letter b sends all l pairs to different states, which are outside of the original cycle. From Lemma 3.3.9, a complete numbering is not possible.

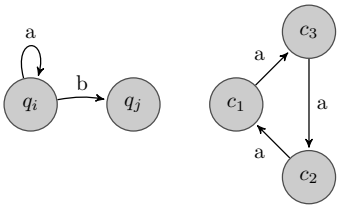


Figure 21: A self loop and a cycle of length 3 for letter a.

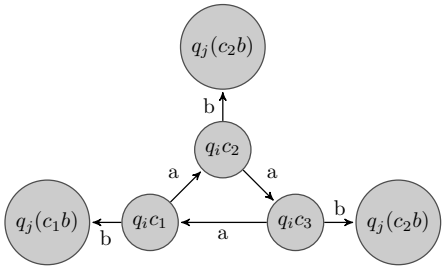


Figure 22: The effect on the graph of pairs.

In the third case, more complicated cases where the applications of b rejoins the cycle and leave it again, and maybe multiple times, before getting back to c_1, we can easily find pairs for which Lemma 3.3.8 applies, and so a complete numbering is impossible. □

A.2.4 Letter labelling a cycle of length 4 and a cycle of length 3

We consider a letter labelling a cycle of length 4 and a cycle of length 3 in a set of at least 8 states. In that case, pairs composed of one state of each are in a cycle of length 12 which has at least three outgoing edges labelled by a second letter, hence Lemma 3.3.9 applies.

Proposition A.2.4. *Let A be a permutation set with a strongly connected square graph, with $n > 7$ states, a cycle of length 3 and a cycle of length 4 labelled by a single letter. Then a complete numbering of the square graph is impossible.*

Proof. Let assume that letter a identifies cycles of length 3 and cycles of length 4. Let C be a cycle of length 3, and D a cycle of length 4. for a graph large enough, there must be an outgoing edge from one of these cycles. Let assume

that $c_1 b = q_{out}$. The pairs of states formed by one in C and one in D are in a cycle of length 12 identified by letter a. All the pairs of this cycle containing state c_1 are sent to a pair outside of the cycle by letter b. Since there are at least three such pairs, from Lemma 3.3.9, a complete numbering is not possible. Figure 23 shows the base graph, and Fig. 24 the square graph.

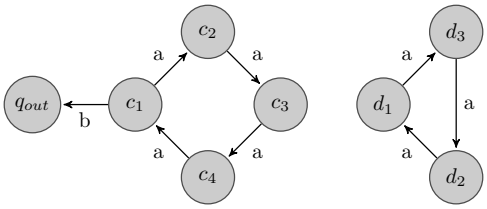


Figure 23: Cycles of length 3 and 4 for letter a.

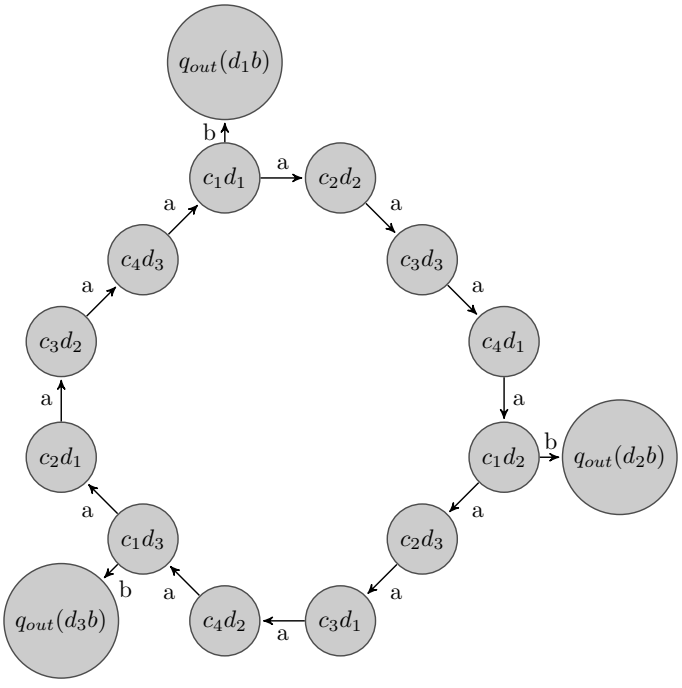


Figure 24: The effect on the graph of pairs.

□

A.2.5 Letters labelling both cycles of length 4 and swaps.

We now consider a letter labelling only cycles of length four and swaps. If there are more than 12 states, the situation of Fig. 25 happens. In that case, we show that letter b must have the same effect as letter a on most of the states, and finally that a complete numbering is impossible.

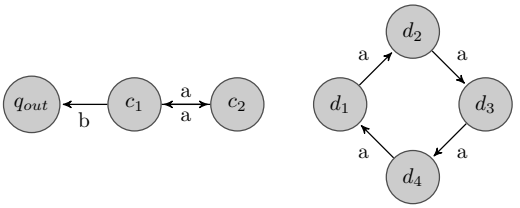


Figure 25: Letter a labels a cycle of length 4 and a swap, and a letter b maps a state of the swap to a 7th state.

Proposition A.2.5. *Let A be a permutation set with a strongly connected square graph, with $n > 12$ states, and a cycle of length 2 and a cycle of length 4 labelled by a single letter. Then a complete numbering of the square graph is impossible.*

Proof. We consider here that letter a identifies a cycle of length 4 D (states d_1, d_2, d_3, d_4) and a swap C (states c_1 and c_2), and that letter b maps one of the swapped states on q_{out}, a state which is not in C or D. This situation always exists if $n \geq 13$ (if $n = 12$ there could be a situation with one cycle of length 4 and four swaps, with only one edge connecting each of them to the cycle of length 4). The base situation is shown in Fig. 25, and the effect on the part of the square graph based on pair c_1d_1 is shown in Fig. 26.

In order to have a complete numbering, the four pairs c_1d_1, c_2d_2, c_1d_3 and c_2d_4 must have numbers k, $k+1$, $k+2$ and $k+3$, for some integer k. Since there are two outgoing edges, in order to have a complete numbering, one must be a jump and the other one must be a decrease to $k-1$.

Without loss of generality, let us assume that the pair with middle numbers is c_1c_3. Therefore, the pair $q_{out}(d_3b)$ must have a number higher than the other numbers of the cycle. This situation is shown in Fig. 27.

The cycle labelled b starting with c_1d_3 and $q_{out}(d_3b)$ has to return to c_1d_3 after some applications of letter b. In order to have a complete numbering, as the numbers of the consecutive pairs of the cycle have to decrease, we need to

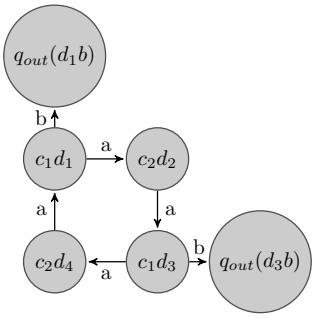


Figure 26: The effect on the square graph of a swap and a cycle of length 4.

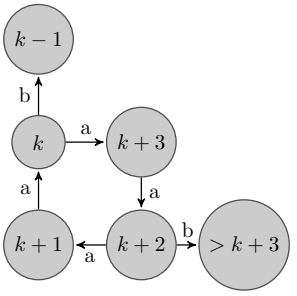


Figure 27: A numbering that would allow a complete numbering in this situation.

have that the pair before c_1c_3 in the cycle has number $k+3$, and therefore to have a complete numbering, it must be c_2d_2. Therefore, a and b have the same effect on c_2d_2, which gives in the graph of pairs the situation shown in Fig. 28.

In the base graph, we chose C and D independent from each other. Therefore, $(c_2d_2)b = c_1d_3$ implies that $c_2b = c_1$ and that $d_2b = d_3$. Figure 29 shows the resulting base graph.

With this situation, we now have that the pair c_2d_4, which has number $k+1$ in Fig. 27, is sent on $c_1(d_4b)$ by letter b. If d_4b is outside of the structure of Fig. 29, then this pair is also sent outside of the structure of Fig. 28, which implies that three edges are outgoing of the same cycle in the square graph, in which case a complete numbering is impossible. Moreover, if the pair c_2d_4 is sent to any other pair than the ones considered in Fig. 28, the argument holds. Therefore, the only remaining possibility to allow for a complete numbering is that $d_4b = d_1$. This leads to the structure of Fig. 30.

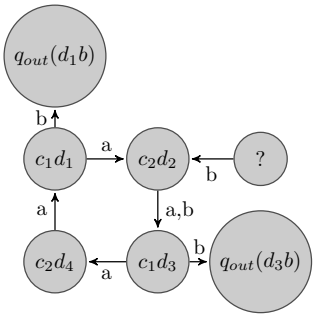


Figure 28: Letter b labels some additional edges to allow for a complete numbering.

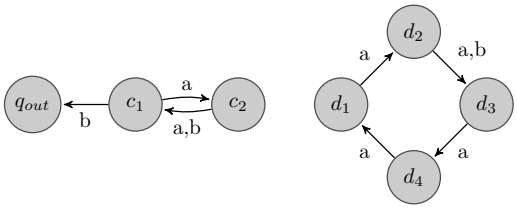


Figure 29: Additional effects of letter b in the base graph.

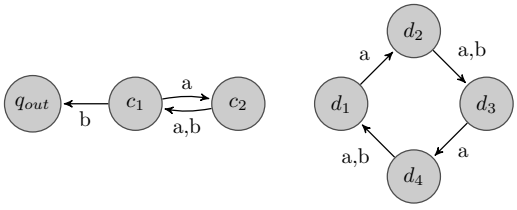


Figure 30: The edge from d_4 to d_1 is also labeled b.

With these deductions on the effect of letter b shown in Fig. 30, let us now consider the part of the square graph which starts with c_1d_2 instead of c_1d_1. The result is shown in Fig. 31.

If $c_1(d_3b)$ or $c_1(d_1b)$ are outside of the structure, we have at least three outgoing edge from the same cycle. Moreover, either c_1d_2 or c_1d_4 must have the lowest number, which means that this pair can have incoming edges from only one other pair. Therefore, we must have that $c_1(d_3b) = c_1d_4$ and $c_1d_2 = c_1(d_1b)$.

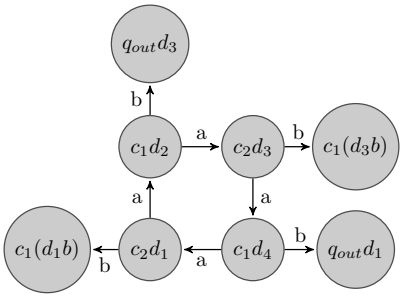


Figure 31: Resulting square graph, considering c_1d_2 as starting pair.

This implies that $d_1b = d_2$ and that $d_3b = d_4$, and that therefore a and b are both labelling the same cycle D. This is shown in Fig. 32.

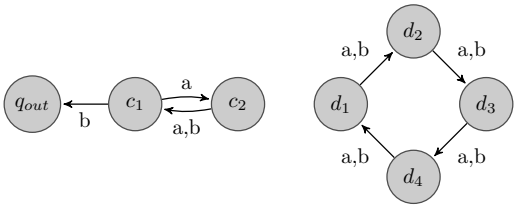


Figure 32: Letters a and b have the same effect on cycle D.

From this structure, let us now look more in details at the effect that letter b has on the base graph, starting from C. As we deduced that, to allow a complete numbering, b has a cycle of length 4, due to previous sections, we only need to consider cases where all the structures formed by b are cycles of length 4 or swaps.

This implies that c_1, c_2 and q_{out} are in the same cycle of length 4 labelled by letter b. Let us rename q_{out} as q_{out1} and consider the 4th state of that cycle q_{out2}. We have the structure of Fig. 33.

The resulting square graph starting from the pair c_1d_2 is shown in Fig. 34.

This particular structure allows only two different complete numberings, either taking $q_{out1}d_1$ as highest number, or $q_{out1}d_3$, and in both cases decreasing by one following the edges in a unique way. One of these possibilities is shown in Fig. 35, with k some integer.

We notice that pairs of states $q_{out1}d_1$ and $q_{out1}d_3$ have numbers $k+6$ and $k+1$ in the two possible numberings, which implies that one of the two pairs

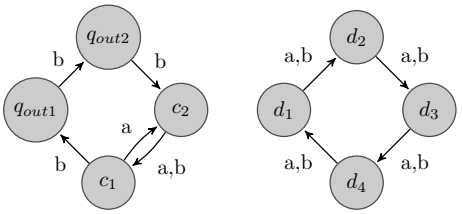


Figure 33: Letter b is also labelling only swaps and cycles of length 4.

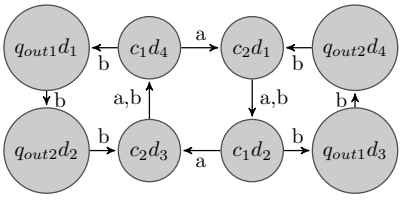


Figure 34: Considering two cycles of length 4 for b.

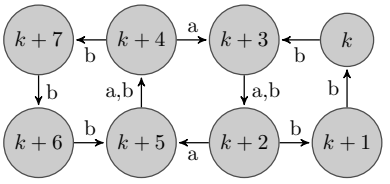


Figure 35: One of the possible numbering allowing for complete numbering associated with this square graph.

is a middle pair. Moreover, in both case, the effect of letters a and b on the pair that have number $k+2$ is defined. This implies that if both of the pairs are sent out of the structure by letter a, there is an incompatibility with the complete numbering, as a cycle labelled a starting by the middle pair is not able to re-enter the structure with a good numbering.

Now, let us look at the effect of letter a on q_{out1}. Either we have $q_{out1}a = q_{out2}$, or we have that $q_{out1}a = q_{out3}$, with q_{out3} a state not considered yet.

If $q_{out1}a = q_{out3}$, a complete numbering is impossible as both pairs $q_{out1}d_1$ and $q_{out1}d_3$ are sent out of the structure.

Now, if $q_{out1}a = q_{out2}$, if q_{out2} is such that $q_{out2}a = q_{out3}$, one of the pairs $q_{out2}d_4$ or $q_{out2}d_2$, which can have numbers k or $k+6$, is sent outside of the structure, and we have the same argument as above.

Therefore, the only case left is if $q_{out1}a = q_{out2}$ and $q_{out2}a = q_{out1}$. This would imply the structure of Fig. 36, which also doesn't allow for a complete numbering, as one of the two added edges connects $k + 6$ to $k + 1$ (in either or the two distributions). We therefore proved that it is not possible to have a complete numbering if cycles of length 4 and swaps are in the same letter.

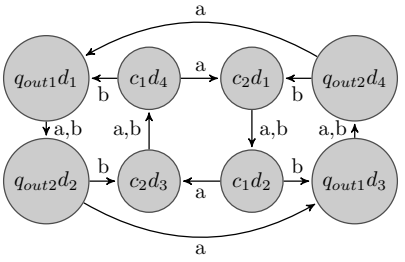


Figure 36: Considering two cycles of length 4 for b.

□

A.2.6 Letters labelling both cycles of length 3 and swaps

Before considering letters labelling only cycles of length 4, we set the case of a letter labelling swaps and cycles of length 3. In this situation, for $n > 9$, it is possible to find a cycle in the square graph with three outgoing hedges, hence Lemma 3.3.9 applies.

Proposition A.2.6. *Let A be a permutation set with a strongly connected square graph, with $n > 9$ states, with a letter labelling both cycles of length 3 and swaps. Then a complete numbering of the square graph is impossible.*

Proof. Let assume that letter a identifies cycles of length 3 and swaps. Let C be a swap, and D a cycle of length 3. For a graph of size $n \geq 11$, there must be an outgoing edge from the swap not related to D. Let assume that $c_1b = q_{out}$. The pairs of states formed by one state in C and one in D are in a cycle of length 6 identified by letter a. All the pairs of this cycle containing state c_1 are sent to a pair outside of the cycle by letter b. Since there are at least three such pairs, a complete numbering is impossible. Figure 37 shows the base graph, and Fig. 38 the square graph.

□

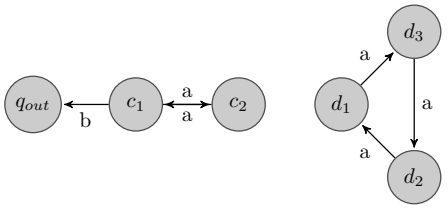


Figure 37: A swap and a cycle of length 3 for letter a.

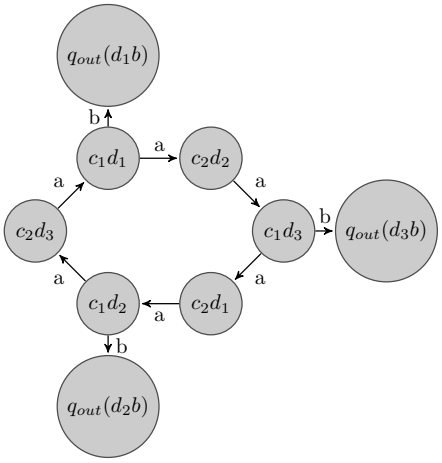


Figure 38: The effect on the graph of pairs.

A.2.7 Letter labelling only cycles of length 4

In that case, there are two cycles of length 4 connected with a second letter. Based on this setting, it is possible to show that a complete numbering is impossible.

Proposition A.2.7. *Let A be a permutation set with a strongly connected square graph, with $n > 8$ states, and a letter labelling only cycles of length 4. Then a complete numbering of the square graph is impossible.*

Proof. In this case, we consider that letter a labels only cycles of length 4. Let us therefore consider a first cycle C, with states c_1, c_2, c_3, c_4. Since the graph has to be strongly connected, there exists a letter b mapping one of these states to a state which is not in C. Without loss of generality, let assume that the outgoing edge starts from c_1. Let us call the attained state d_1. Since letter a labels only cycles of length 4, we must have that d_1 is also in a cycle of length

4 labelled by a. Let us name this cycle D, and let the other states of this cycle be d_2, d_3 and d_4. We have the situation shown in Fig. 39.

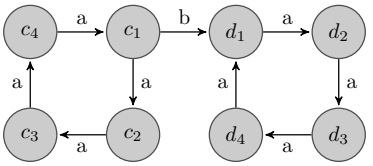


Figure 39: Two cycles of length 4 labelled by a, and letter b labelling one edge between them.

With this situation, let us now consider the cycle labelled by b and containing c_1 and d_1. Three cases are to be analyzed: b labels a swap, b labels a cycle of length 3, b labels a cycle of length 4.

In the two latter cases, the cycle has three outgoing edges labelled a, namely one defined by cycle C, and one defined by cycle D. This leads to a direct contradiction due to lemma 3.3.9.

Therefore, the only case to analyze is if b labels a swap, and that b doesn't label any outgoing edge from C or from D. Since there can be only one outgoing edge from each of these cycles, b has only internal effects on the remaining states of C and D, and b cannot label cycles of length 3 at the same time as a cycle of length 2 (see proposition A.2.6). The situation is shown in Fig. 40.

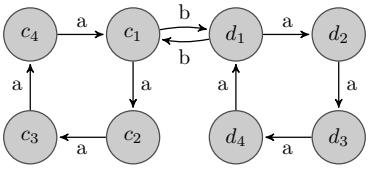


Figure 40: Two cycles of length 4 labelled by a, and letter b labelling a swap between them.

Let us consider the effect that letter b has on the states of C. Either there is a swap and one self loop, either there are only self loops.

First, if there are only self loops, we have the situation shown in Fig. 41.

From this situation, the effect on the square graph based on pair c_1c_2 is shown in Fig. 42. This situation doesn't allow for a complete numbering. Indeed there are four cycles of length 4. In a complete numbering, one of these cycle would have the highest numbers. However each of the cycles is

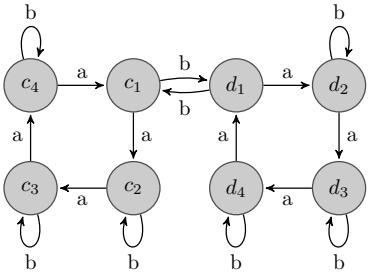


Figure 41: Two cycles of length 4 labelled by a, letter b labelling a swap between them and self loops on the other states.

connected with two other cycles, therefore we have the same argument as in Lemma 3.3.11.

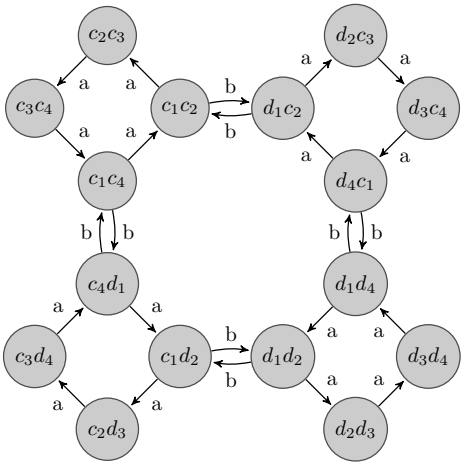


Figure 42: Square graph obtained from the basic graph in Fig. 41.

Now, let us consider the case where there is a swap in C and a self loop. There are three different possible situations, depending on the position of the self loop. The base graph are shown in Fig. 43, Fig. 45, Fig. 47 and the resulting square graphs are shown in Fig. 44, Fig. 46, Fig. 48.

We notice that there are three outgoing edges labelled b from a cycle labelled a in the cases shown in Fig. 44 and Fig. 48. Therefore these two situations don't allow for a complete numbering. However the situation of Fig. 46 doesn't give a direct argument. We therefore have to analyze the case where this

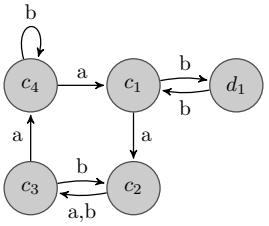


Figure 43: Cycle of length 4 labelled by a, letter b doing a swap outside, a swap inside and a self loop, first case.

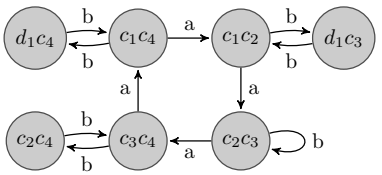


Figure 44: Square graph of the base situation of Fig. 43.

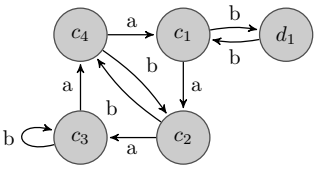


Figure 45: Cycle of length 4 labelled by a, letter b doing a swap outside, a swap inside and a self loop, second case.

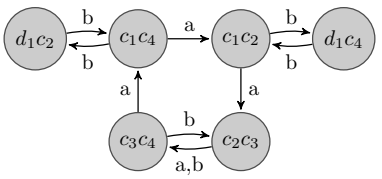


Figure 46: Square graph of the base situation from Fig. 45.

situation is present also in the cycle D, and the case where this situation is in cycle C and when b labels self loops in D. These two cases are represented in Fig. 49 and Fig. 50.

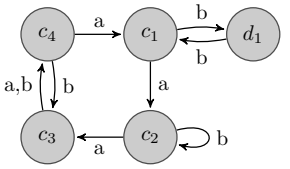


Figure 47: Cycle of length 4 labelled by a, letter b doing a swap outside, a swap inside and a self loop, third case.

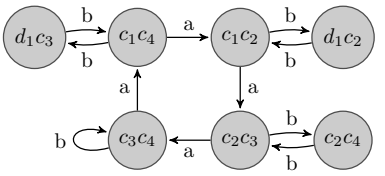


Figure 48: Square graph of the base situation from Fig. 47.

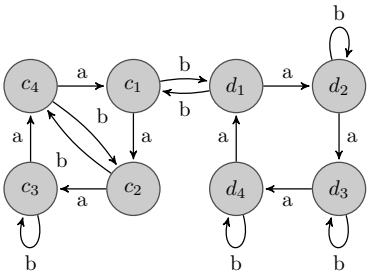


Figure 49: Two cycles of length 4 labelled by a, letter b labelling a swap between them, structure with a swap on C.

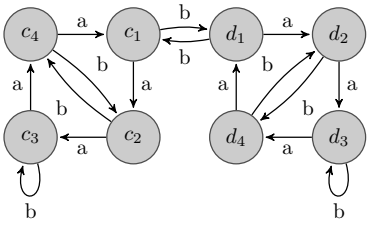


Figure 50: Two cycles of length 4 labelled by a, letter b labelling a swap between them, structure with a swap on both side.

In both cases, the square graph starting from the pair c_1d_2 provides a cycle of four states labelled by letter a with three outgoing edges labelled b, making a complete numbering impossible. The situation is shown in Fig. 51 and Fig. 52.

Therefore, we showed that a complete numbering is not possible if one of the letters is only labelling cycles of length 4.

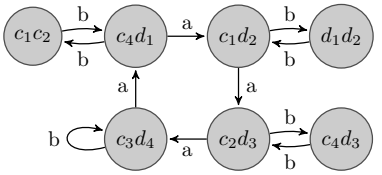


Figure 51: Square graph of the situation from Fig. 49.

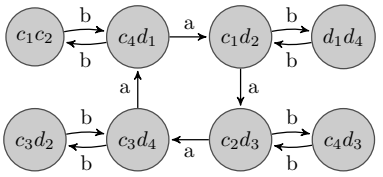


Figure 52: Square graph of the situation from Fig 50.

□

A.2.8 Letter labelling only cycles of length 3

In that case, there are two cycles of length 3 connected with a second letter. Based on this setting, it is possible to show that a complete numbering is impossible.

Proposition A.2.8. *Let A be a permutation set with a strongly connected square graph, with $n > 6$ states, and a letter labelling only cycles of length 3. Then a complete numbering of the square graph is impossible.*

Proof. Again, as the graph is strongly connected, there must be two cycles of length 3 C and D labelled by a, and a letter b such that $c_1b = d_1$.

Let us now consider the cycle labelled by b and starting with the pair c_1d_1. If this cycle is larger than 2, then it has three outgoing edges, and therefore doesn't allow for a complete numbering.

Therefore, as in the case with one letter labelling only cycles of length 4, the only case left is when letter b labels a swap between c_1 and d_1. This situation is shown in Fig. 53.

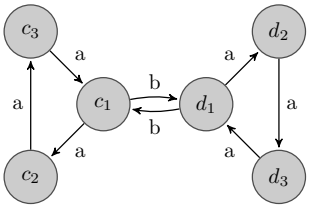


Figure 53: Two cycles of length 3 labelled by a, letter b labelling a swap between them.

Again, if letter b is mapping some other states of C or D outside of them, a complete numbering won't be possible. Therefore, b is either acting as swap or as self loop inside of C and D. The three possible situations are shown in Fig. 54, Fig. 56 and Fig. 58, and the square graphs based on them proving that complete numberings are impossible are shown in Fig. 55, Fig. 57 and Fig. 59. In the two first cases, there are three outgoing edges from a cycle in the square graph, in the third case there are four cycles with two outgoing edges connecting each of them. Therefore, no complete numbering is possible if one letter labels only cycles of length 3.

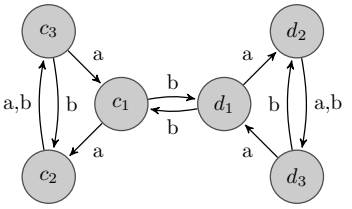


Figure 54: Two cycles of length 3 labelled by a, letter b labelling a swap between them and swaps between the other states of each cycle.

□

A.2.9 Largest one-letter cycle of length 2

If all the permutations contain only swaps and self loops on states, then at least three letters are needed in order to obtain a strongly connected square

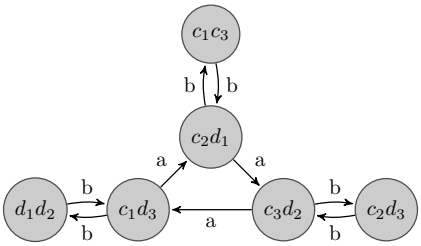


Figure 55: Square graph of the base situation from Fig. 54.

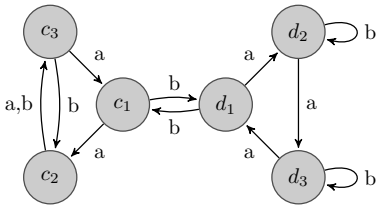


Figure 56: Two cycles of length 3 labelled by a, letter b labelling a swap between them, a swap for the two other states of the first cycle, and self loops on the other states of the second cycle.

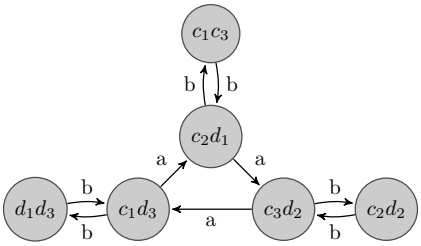


Figure 57: Square graph of the base situation from Fig. 56.

graph. Moreover, there is a pair from which three letters labels different outgoing edges, therefore Lemma 3.3.9 applies.

Proposition A.2.9. *Let A be a permutation set with a strongly connected square graph, with $n > 6$ states, and such that all letters label only self loops and swaps. Then a complete numbering of the square graph is impossible.*

Proof. We notice that the only possible effects on the square graph are the following:

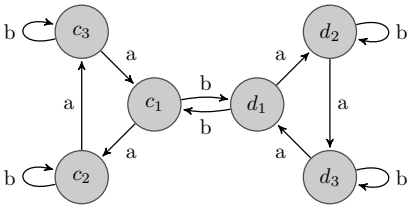


Figure 58: Two cycles of length 3 labelled by a, letter b labelling a swap between them and self loops for the remaining states.

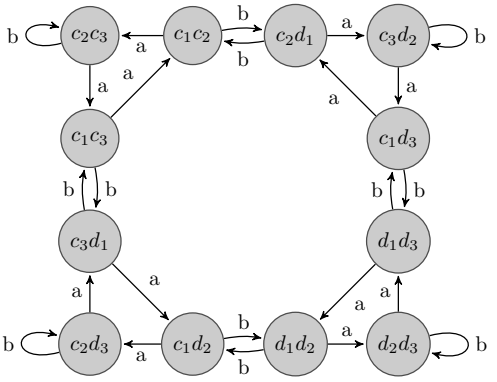


Figure 59: Square graph of the base situation from Fig. 58.

- if $q_ia = q_i$ and $q_ja = q_j$, then the edge a is a self loop for the pair q_iq_j,
- if $q_ia = q_j$ and $q_ja = q_i$, then the edge a is a self loop for the pair q_iq_j,
- if $q_ia = q_i$ and $q_ja = q_l$, then a is acting as a swap between the pair q_iq_j and the pair q_iq_l,
- if $q_ia = q_k$ and $q_ja = q_l$, then a is acting as a swap between the pair q_iq_j and the pair q_jq_l.

This implies that the letters also act as self loops and swaps in the square graph. For example, the effect of the letter of Fig. 60 on the square graph is shown in Fig. 61.

In order to obtain a complete numbering, the highest and lowest pair must swap only with the second highest and the second lowest. All the other pairs must swap both with the pair with its number plus one and with the pair with

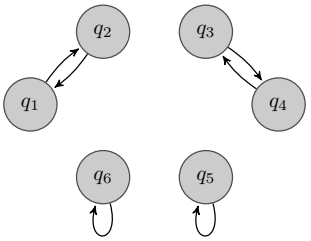


Figure 60: Letter having only swaps and self loops.

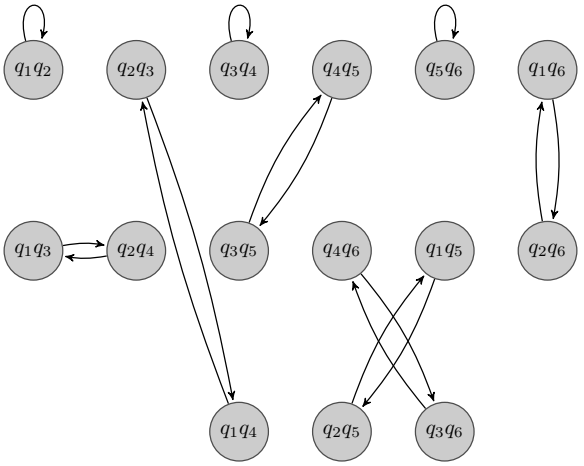


Figure 61: Effect of swaps and self loops on the graph of pairs.

its number minus one. This implies that there must be at least $n(n-1)/2-1$ swaps in total in the square graph.

We notice that one single letter with only swaps and self loops creates at least two self loops in the square graph, since a swap between two states is a self loop on that pair, and two self loops on states are also a self loop for that pair (therefore above 4 nodes, there are at least two self loops). As this kind of matrix creates at most one swap for each two pairs, it can create at most $(n(n-1)/2-2)\times 1/2 = (n(n-1)/4-1)$ swaps. As the total of swaps need to be at least $n(n-1)/2-1 > 2\times(n(n-1)/4-1)$, it implies that at least three letters are needed to obtain a strongly connected graph of pairs. We therefore restrict ourselves to cases with at least three different letters.

First, let assume that there exists a state q_i such that three letters a, b, c

swap that state with three different states q_{j1}, q_{j2} and q_{j3}. Let us consider a state q_k independent of q_i, i.e. such that q_k, $q_k a$, $q_k b$, $q_k c$ are all different from q_i, q_{j1}, q_{j2} and q_{j3}, as shown in Fig. 62. Note that $q_k a$, $q_k b$ and $q_k c$ can be equal to each other or to q_k. Then, the three letters swap the pair $q_i q_k$ with three different pairs, as shown in Fig. 63, and therefore the complete numbering is impossible. We therefore restrict ourselves to cases where one single state is only connected with at most two different states. Therefore, for each state, either one of the letter is a self loop, or two letters have the same effect.

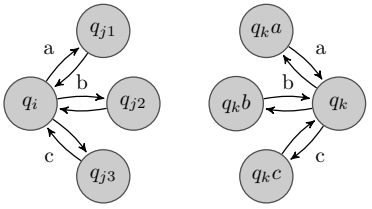


Figure 62: A state q_i swaps with three different states, and an independent state q_k.

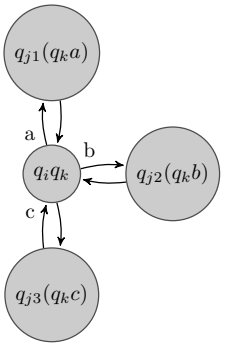


Figure 63: The effect on the pair graph.

Let us now take letter a as having the largest number of self loops in all the letters. Since there can't be an identity matrix, there must exists states q_i and q_j such that $q_i a = q_j$. Now consider a state q_{k1} such that a is a self loop for this state, that is not extremal, and is independent of q_i. For this state, there must exists a letter b such that $q_{k1} b = q_{k2}$ and a letter c such that $q_{k1} c = q_{k3}$, as it has to be connected with two other states. In that setting,

- the pair $q_i q_{k1}$ is sent to $q_j q_{k1}$ by letter a,

- the pair $q_i q_{k1}$ is sent to $(q_i b)q_{k2}$ by letter b,
- the pair $q_i q_{k1}$ is sent to $(q_i c)q_{k3}$ by letter c.

Since we assumed that q_{k1} was independent of q_i, these three pairs are different, and therefore a complete numbering is not possible. The situation is shown in Fig. 64 and Fig. 65.

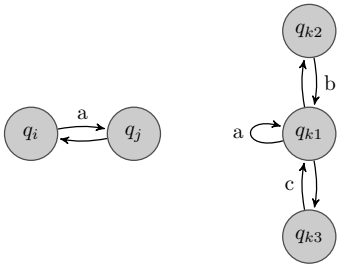


Figure 64: A state q_i has a swap with q_j labelled a. An independent state q_{k1} has a self loop labelled a and swaps labelled by b and c.

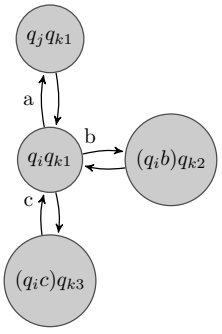


Figure 65: The effect on the pair graph.

□

Scientific Communication

I provide here an overview of the journal articles, conference articles, posters and talks done during my doctoral training. Notice that I worked on two different research topics, synchronizing automata and air traffic management, and that the present thesis only contains the results on synchronizing automata.

Journal Articles

1. Gonze, François, and Raphaël M. Jungers. "Hardly Reachable Subsets and Completely Reachable Automata" *Journal of Automata, Languages and Combinatorics*, submitted.

2. Gonze, François, Vladimir V. Gusev, Balázs Gerencsér, Raphaël M. Jungers and Mikhail V. Volkov "On the Interplay Between Černý and Babai's Conjectures." *International Journal of Foundations of Computer Science* 30.01 (2019): 93-114. [GGJ+19]

3. Gonze, François, Andrea Simonetto, Etienne Huens, Jean Boucquey and Raphaël M. Jungers "Probabilistic Occupancy Counts and Flight Criticality Measures in Air Traffic Management." *Journal of Air Transportation* 26.3 (2018): 94-103. [GSH+18]

4. Gonze, François and Raphaël M. Jungers. "On the synchronizing probability function and the triple rendezvous time for synchronizing automata." *SIAM Journal on Discrete Mathematics* 30.2 (2016): 995-1014. [GJ16]

5. Gonze, François, Raphaël M. Jungers and Avraham N. Trahtman. "A Note on a Recent Attempt to Improve the Pin-Frankl Bound." *Discrete Mathematics & Theoretical Computer Science* 17.1 (2015): 307-308. [GJT15]

Conference Articles

1. Goyens, Florentin, François Gonze, Andrea Simonetto, Etienne Huens, Jean Boucquey and Raphaël M. Jungers "A Probabilistic Model for Precedence Rules and Reactionary Delay in Air Traffic Management." International Conference on Research on Air Transportation, 2018. [GGS+18]

2. Gonze, François and Raphaël M. Jungers. "On completely reachable automata and subset reachability." International Conference on Developments in Language Theory. Springer, Cham, 2018. [GJ18]

3. Gonze, François, Vladimir V. Gusev, Balázs Gerencsér, Raphaël M. Jungers and Mikhail V. Volkov "On the interplay between Babai and Černý's conjectures." International Conference on Developments in Language Theory. Springer, Cham, 2017. [GGG+17]

4. Boucquey, Jean, François Gonze, Andrew Hately, Etienne Huens, Richard Irvine, Stephan Steurs and Raphaël M. Jungers "Probabilistic Traffic Models for Occupancy Counting." 7th SESAR Innovation Days. 2017. [BGH+17]

5. Gonze, François, Andrea Simonetto, Etienne Huens, Jean Boucquey and Raphaël M. Jungers "Probabilistic occupancy counts and flight criticality measures for ATM." Proceedings of the 12th USA/Europe ATM R&D Seminar. 2017. [GSH+17]

6. Gonze, François and Raphaël M. Jungers. "On the synchronizing probability function and the triple rendezvous time." International Conference on Language and Automata Theory and Applications. Springer, Cham, 2015. [GJ15]

Poster

1. COPTRA Consortium, "Being Certain of our Uncertainty ... Probably", 6th SESAR Innovation Days. 2016.

Seminars and Talks

1. Conference presentation "On completely reachable automata and subset reachability.", at the International Conference on Developments in Lan-

guage Theory 2018 held from 10 to 14 September 2018 in Tokyo (Japan).

2. Conference presentation "A Probabilistic Model for Precedence Rules and Reactionary Delay in Air Traffic Management.", International Conference on Research on Air Transportation, held from 26 to 29 June 2018 in Barcelona (Spain).

3. Conference presentation "On the interplay between Babai and Cerny's conjectures", at the International Conference on Developments in Language Theory 2017 held from 7 to 11 August 2017 in Liège (Belgium).

4. Conference presentation "Probabilistic Occupancy Counts and Flight Criticality Measures for ATM", at the 12th USA/Europe ATM Research and Development Seminar held from 26 to 30 June 2017 in Seattle (USA).

5. Conference presentation "Sector Congestion and Criticality Measures for Flights in ATM" at the 36th Benelux Meeting on Systems and Control held from 28 to 30 March 2017 in Spa (Belgium).

6. Seminar presentation "On the Synchronizing Probability Function and the Triple Rendezvous Time" at the Seminar of discrete mathematics of the Liège University on the 26th of May 2016, Liège (Belgium).

7. Conference presentation "Combining Probable Trajectories in Air Traffic Management" at the 35th Benelux Meeting on Systems and Control held from 22 to 24 March 2016 in Soesterberg (Netherlands).

8. Conference presentation "Synchronization Approached Through the Lens of Primitivity" at the 35th Benelux Meeting on Systems and Control held from 22 to 24 March 2016 in Soesterberg (Netherlands).

9. Conference presentation "New Approaches of Cerny's Conjecture" at the 34th Benelux Meeting on Systems and Control held from 24 to 26 March 2015 in Lommel (Belgium).

10. Conference presentation "The Synchronizing Probability Function and the Triple Rendezvous Time as Approaches to Cerny's Conjecture" at the 9th International Conference on Language and Automata Theory and Applications LATA 2015 held from 2 to 6 March 2015 in Nice (France).

11. Seminar presentation "The Synchronizing Probability Function and the Triple Rendezvous Time as Approaches to Cerny's Conjecture" at the

Seminar in Systems and Control (CESAME) on the 14th of October 2014, Louvain-la-Neuve (Belgium).

12. Conference presentation "The Synchronizing Probability Function and the Triple Rendezvous Time as Approaches to Cerny's Conjecture" at the 15th Mons Theoretical Computer Science Days, held from 23 to 26 september 2014 in Nancy (France).

Nomenclature

Terminology

Alphabet	The alphabet is the set of letters Σ of the automaton.
Automaton (DFA)	Other name of finite state system.
Diameter	The diameter of a graph is the maximal length of the shortest path between any two of its states.
Directed graph	A directed graph (N, E) is composed of a set of nodes N and a set of directed edges E between the nodes.
Deficiency	The deficiency of a word w is the value $n - \|Qw\|$. It is equal to the number of states which are not in its image.
Finite state system	A finite state system is a triplet (Q, Σ, δ) with Q the set of states, Σ the alphabet of letters and δ the transition function.
Γ_1-graph	A directed graph obtained from letters of deficiency one of an automaton.
Image	The image of a set of states S by a word w is the set Sw. By extension, the image of a word w is Qw.
Letter	The letters are the elements of an alphabet.

Power automaton	The power automaton of an automaton $\mathscr{A} = (Q, \Sigma, \delta)$ is the automaton with all possible subsets of Q as states, the same alphabet, and the natural extension of δ to subsets.
Preimage	The preimage of a set of states S by a word w is the set of states which are mapped in S by w.
Rank	The rank of a word is the cardinality of its image $\|Qw\|$.
Reset threshold	The reset threshold of an automaton is the length of its shortest synchronizing word.
Square graph	The square graph of an automaton is the power automaton restricted to pairs, edges starting from pairs and single states with incoming edges starting from pairs.
Synchronize	A word w synchronizes a set of states $S \subset Q$ if $\|Sw\| = 1$.
Synchronizing	A word w is synchronizing if $\|Qw\|$=1. An automaton is synchronizing if it has a synchronizing word.
Synchronizing probability function (SPF)	The synchronizing probability function of an automaton is a function of $l \in \mathbb{N}$ which returns the results of an optimization program over the set of words of length lower or equal to l.
Transition function	The function defining the effect of the letters on the automaton.
Word	A word is a sequence of letters.

Mathematical Notation

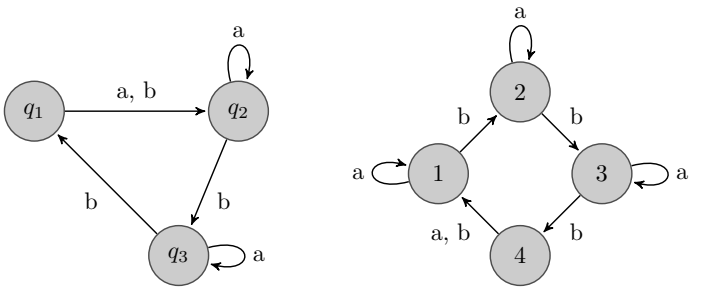
$\mathscr{A}$	Generic automaton
$\Gamma_1()$	Γ_1-graph of an automaton
Q	Set of states of the generic automaton
Σ	Alphabet of the generic automaton
Σ^*	Set of all words composed of letters from Σ
$\Sigma^{\leq k}$	Set of all words of length at most k composed of letters from Σ
S	Generic subset of state
$SG()$	Square graph of an automaton
$P()$	Power automaton of an automaton
T_3	Triple rendezvous time

Acronyms

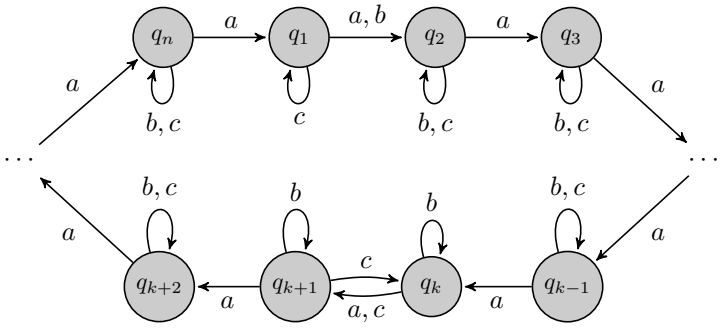
DFA	Deterministic finite automaton
JSR	Joint spectral radius
SPF	Synchronizing probability function

Automata Presented in the Thesis

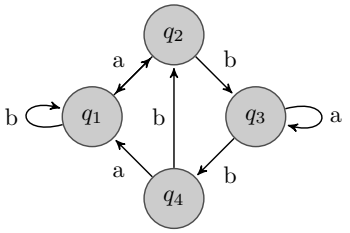
$\mathscr{C}_n$



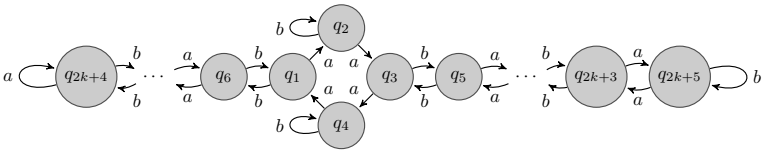
$\mathscr{CB}_{n,k}$



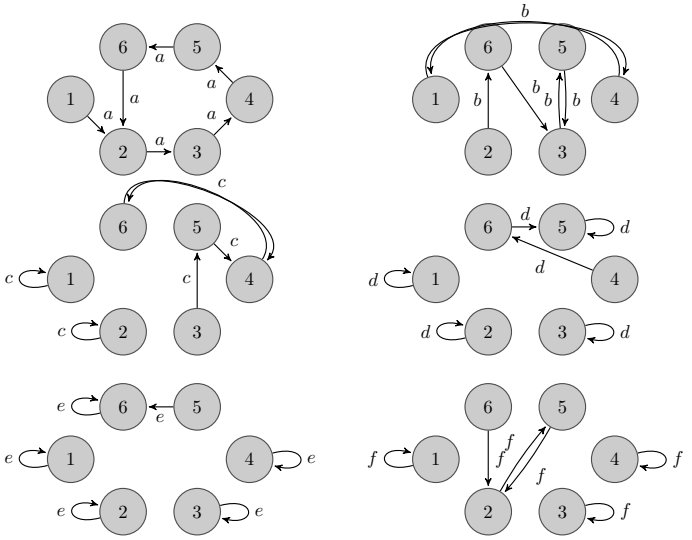
$\mathscr{E}$



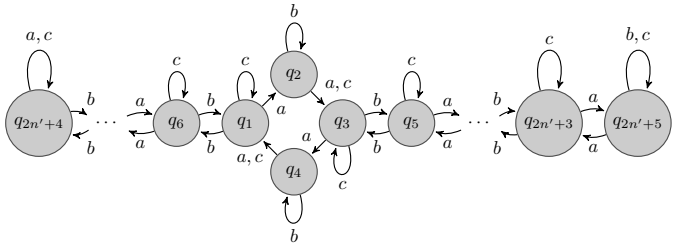
$\mathscr{F}_{2k+5}$



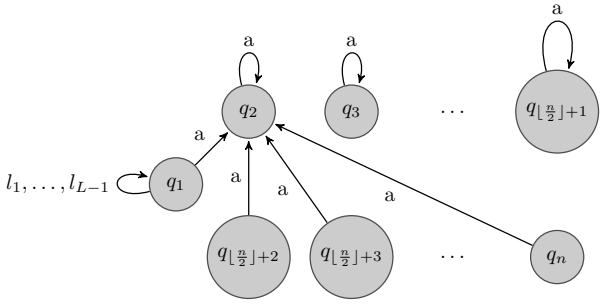
$\mathscr{G}$



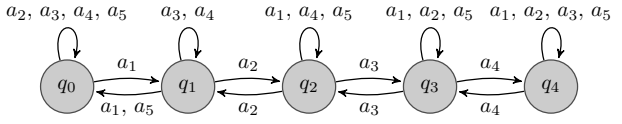
$\mathscr{P}_{2,n}$



$\mathscr{P}_{3,n}$



$\mathscr{V}_n$



Bibliography

[ACCJ19] Umer Azfar, Costanza Catalano, Ludovic Charlier, and Raphaël M. Jungers. A linear bound on the k-rendezvous time for primitive sets of nz matrices. *arXiv preprint arXiv:1903.10421*, 2019.

[ACS17] João Araújo, Peter J. Cameron, and Benjamin Steinberg. Between primitive and 2-transitive: Synchronization and its friends. *EMS Surveys in Mathematical Sciences*, 4(2):101–184, 2017.

[ACV02] Dmitry S. Ananichev, Alessandra Cherubini, and Mikhail V. Volkov. An inverse automata algorithm for recognizing 2-collapsing words. In *International Conference on Developments in Language Theory*, pages 270–282. Springer, 2002.

[Adl70] Roy L. Adler. *Similarity of automorphisms of the torus*, volume 98. American Mathematical Society, 1970.

[AGV10] Dmitry S. Ananichev, Vladimir V. Gusev, and Mikhail V. Volkov. Slowly synchronizing automata and digraphs. *Mathematical Foundations of Computer Science 2010*, pages 55–65, 2010.

[AGV13] Dmitry S. Ananichev, Vladimir V. Gusev, and Mikhail V. Volkov. Primitive digraphs with large exponents and slowly synchronizing automata. *Journal of mathematical sciences*, 192(3):263–278, 2013.

[AS09] Jorge Almeida and Benjamin Steinberg. Matrix mortality and the Černỳ-Pin conjecture. In *DLT 2009*, volume 5583 of *Lecture Notes in Computer Science*, pages 67–80. Springer, 2009.

[AV04] Dmitry S. Ananichev and Mikhail V. Volkov. Some results on Černỳ type problems for transformation semigroups. In *Semigroups and languages*, pages 23–42. World Scientific, 2004.

[AV05] Dmitry S. Ananichev and Mikhail V. Volkov. Synchronizing generalized monotonic automata. *Theoretical Computer Science*, 330(1):3–13, 2005.

[AVZ07] Dmitry S. Ananichev, Mikhail V. Volkov, and Yulia I. Zaks. Synchronizing automata with a letter of deficiency 2. *Theoretical computer science*, 376(1-2):30–41, 2007.

[BBP11] Marie-Pierre Béal, Mikhail. V. Berlinkov, and Dominique Perrin. A quadratic upper bound on the size of a synchronizing word in one-cluster automata. *International Journal of Foundations of Computer Science*, 22(2):277–288, 2011.

[Ber11] Mikhail. V. Berlinkov. On a conjecture by Carpi and D'Alessandro. *International Journal of Foundations of Computer Science*, 22(07):1565–1576, 2011.

[BFS18] Mikhail V. Berlinkov, Robert Ferens, and Marek Szykuła. Complexity of Preimage Problems for Deterministic Finite Automata. In *MFCS 2018*, volume 117 of *Leibniz International Proceedings in Informatics*, pages 32:1–32:14. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2018.

[BGH+17] Jean Boucquey, François Gonze, Andrew Hately, Etienne Huens, Richard Irvine, Stefan Steurs, and Raphaël M. Jungers. Probabilistic traffic models for occupancy counting. In *7th SESAR Innovation Days*, 2017.

[BJO15] Vincent D. Blondel, Raphaël M. Jungers, and Alex Olshevsky. On primitivity of sets of matrices. *Automatica*, 61:80–88, 2015.

[BP14] Marie-Pierre Béal and Dominique Perrin. A quadratic algorithm for road coloring. *Discrete Applied Mathematics*, 169:15–29, 2014.

[BPQR14] Simon Busard, Charles Pecheur, Hongyang Qu, and Franco Raimondi. Improving the model checking of strategies under partial observability and fairness constraints. In *ICFEM 2014*, vol-

ume 8829 of *Lecture Notes in Computer Science*, pages 27–42. Springer, 2014.

[BPR09] Jean Berstel, Dominique Perrin, and Christophe Reutenauer. *Codes and Automata*. CUP, 2009.

[BR10] Valérie Berthé and Michel Rigo. *Combinatorics, automata and number theory*, volume 135. Cambridge University Press, 2010.

[Brz13] Janusz A. Brzozowski. In search of most complex regular languages. *International Journal of Foundations of Computer Science*, 24(6):691–708, 2013.

[BS92] László Babai and Ákos Seress. On the diameter of permutation groups. *European Journal of Combinatorics*, 13(4):231–243, 1992.

[BV04] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[BV16] Eugenija A. Bondar and Mikhail V. Volkov. Completely reachable automata. In *DCFS 2016*, volume 9777 of *Lecture Notes in Computer Science*, pages 1–17. Springer, 2016.

[BV18] Eugenija A. Bondar and Mikhail V. Volkov. A characterization of completely reachable automata. In *DLT 2018*, volume 11088 of *Lecture Notes in Computer Science*, pages 145–155. Springer, 2018.

[Cam13] Peter J. Cameron. Dixon's theorem and random synchronization. *Discrete Mathematics*, 313(11):1233–1236, 2013.

[Car14] Ângela F. P. Cardoso. *The Černý conjecture and other synchronization problems*. PhD. thesis, Faculdade de Ciencias da Universidade do Porto, 2014.

[Cat19] Costanza Catalano. *Probabilistic methods for primitive matrix semigroup*. PhD. thesis, Gran Sasso Science Institute, L'Aquila, 2019.

[CD09] Arturo Carpi and Flavio D'Alessandro. Strongly transitive automata and the Černỳ conjecture. *Acta Informatica*, 46(8):591–607, 2009.

[CD10] Arturo Carpi and Flavio D'Alessandro. On the hybrid Černỳ-road coloring problem and Hamiltonian paths. In *DLT 2010*, volume 6224 of *Lecture Notes in Computer Science*, pages 124–135. Springer, 2010.

[CD13] Arturo Carpi and Flavio D'Alessandro. Independent sets of words and the synchronization problem. *Advances in Applied Mathematics*, 50(3):339–355, 2013.

[Černý64] Jan Černý. Poznámka k homogénnym experimentom s konečnými automatami. *Mat.-fyz. Časopis Slovenskej Akadémie Vied*, 14(3):208–216, 1964. In Slovak.

[ČernýPR71] Jan Černý, Alica Pirická, and Blanka Rosenauerová. On directable automata. *Kybernetica*, 7:289–298, 1971.

[CHJ15] Pierre-Yves Chevalier, Julien. M. Hendrickx, and Raphaël M. Jungers. Reachability of consensus and synchronizing automata. In *2015 IEEE 54th Annual Conference on Decision and Control (CDC)*, pages 4139–4144. IEEE, 2015.

[CJ18a] Costanza Catalano and Raphaël M. Jungers. On random primitive sets, directable NDFAs and the generation of slowly synchronizing DFAs. 2018.

[CJ18b] Costanza Catalano and Raphaël M. Jungers. The synchronizing probability function for primitive sets of matrices. In *DLT 2018*, volume 11088 of *Lecture Notes in Computer Science*, pages 194–205. Springer, 2018.

[CP09] Sébastien Combéfis and Charles Pecheur. A bisimulation-based approach to the analysis of human-computer interaction. In *Proceedings of the 1st ACM SIGCHI symposium on Engineering interactive computing systems*, pages 101–110. ACM, 2009.

[CR10] Krzysztof Chmiel and Adam Roman. COMPAS-A computing package for synchronization. In *CIAA 2010*, volume 6482 of *Lecture Notes in Computer Science*, pages 79–86. Springer, 2010.

[CW96] Edmund M. Clarke and Jeannette M. Wing. Formal methods: State of the art and future directions. *ACM Computing Surveys (CSUR)*, 28(4):626–643, 1996.

[DBDZ17] Michiel De Bondt, Henk Don, and Hans Zantema. DFAs and PFAs with long shortest synchronizing word length. In *DLT2017*, volume 10396 of *Lecture Notes in Computer Science*, pages 122–133. Springer, 2017.

[Dix69] John D. Dixon. The probability of generating the symmetric group. *Mathematische Zeitschrift*, 110:199–205, 1969.

[Don16] Henk Don. The Černý conjecture and 1-contracting automata. *Electronic Journal of Combinatorics*, 23(3):3–12, 2016.

[Dub98] L. Dubuc. Sur les automates circulaires et la conjecture de Černý. *RAIRO Informatique Theorique et Appliquée*, 32:21–34, 1998.

[Epp90] David Eppstein. Reset sequences for monotonic automata. *SIAM Journal on Computing*, 19(3):500–510, 1990.

[FJR+98] Joel Friedman, Antoine Joux, Yuval Roichman, Jacques Stern, and Jean-Pierre Tillich. The action of a few permutations on r-tuples is quickly transitive. *Random Structures & Algorithms*, 12(4):335–350, 1998.

[FK17] Lukas Fleischer and Manfred Kufleitner. Green's relations in finite transformation semigroups. In *CSR 2017*, volume 10304 of *Lecture Notes in Computer Science*, pages 112–125. Springer, 2017.

[Fra82] Péter Frankl. An extremal problem for two families of sets. *European Journal of Combinatorics*, 3:125–127, 1982.

[FS07] Dirk Frettlöh and Bernd Sing. Computing modular coincidences for substitution tilings and point sets. *Discrete & Computational Geometry*, 37(3):381–407, 2007.

[GGG+17] François Gonze, Vladimir V. Gusev, Balázs Gerencsér, Raphaël M. Jungers, and Mikhail V. Volkov. On the interplay between Babai and Černý's conjectures. In *DLT 2017*, volume 10396 of *Lecture Notes in Computer Science*, pages 185–197. Springer, 2017.

[GGJ16a] Balázs Gerencsér, Vladimir V. Gusev, and Raphaël M. Jungers. Primitive sets of nonnegative matrices and synchronizing automata. *SIAM Journal on Matrix Analysis and Applications*, 39(1):83–98, 2016.

[GGJ16b] François Gonze, Balázs Gerencsér, and Raphaël M. Jungers. Synchronization approached through the lenses of primitivity. In *35th Benelux Meeting on Systems and Control*, page 96, 2016.

[GGJ+19] François Gonze, Vladimir V. Gusev, Raphaël M. Jungers, Balázs Gerencsér, and Mikhail V. Volkov. On the interplay between Černỳ and Babai's conjectures. *International Journal of Foundations of Computer Science*, 30(01):93–114, 2019.

[GGS+18] Florentin Goyens, François Gonze, Andrea Simonetto, Etienne Huens, Jean Boucquey, and Raphaël M. Jungers. A probabilistic model for precedence rules and reactionary delay in air traffic management. In *8th International Conference on Research on Air Transportation*, 2018.

[Gin58] Seymour Ginsburg. On the length of the smallest uniform experiment which distinguishes the terminal states of a machine. *Journal of the ACM*, 5(3):266–280, 1958.

[GING09] Zsolt Gazdag, Szabolcs Iván, and Judit Nagy-György. Improved upper bounds on synchronizing nondeterministic automata. *Information Processing Letters*, 109(17):986–990, 2009.

[GJ15] François Gonze and Raphaël M. Jungers. On the synchronizing probability function and the triple rendezvous time: New approaches to Černý's conjecture. In *LATA 2015*, volume 8977 of *Lecture Notes in Computer Science*, pages 212–223. Springer, 2015.

[GJ16] François Gonze and Raphaël M. Jungers. On the synchronizing probability function and the triple rendezvous time for synchronizing automata. *SIAM Journal on Discrete Mathematics*, 30(2):995–1014, 2016.

[GJ18] François Gonze and Raphaël M. Jungers. on completely reachable automata and subset reachability. In *DLT 2018*, volume 11088

of *Lecture Notes in Computer Science*, pages 330–341. Springer, 2018.

[GJT15] François Gonze, Raphaël M. Jungers, and Avraham N. Trahtman. A note on a recent attempt to improve the Pin-Frankl bound. *Discrete Mathematics and Theoretical Computer Science*, 17(1):307–308, 2015.

[GK13] Mariusz Grech and Andrzej Kisielewicz. The Černý conjecture for automata respecting intervals of a directed graph. *Discrete Mathematics and Theoretical Computer Science*, 15(3):61–72, 2013.

[GM09] Olexandr Ganyushkin and Volodymyr Mazorchuk. *Classical Finite Transformation Semigroups: An Introduction.* Springer, 2009.

[GPS17] Vladimir V. Gusev, Elena V. Pribavkina, and Marek Szykuła. Around the road coloring theorem. In *Proceedings of the Fourth Russian Finnish Symposium on Discrete Mathematics*, volume 26 of *TUCS Lecture Notes*, pages 52–56. Turku center for computer science, 2017.

[GS15] Paweł Gawrychowski and Damian Straszak. Strong inapproximability of the shortest reset word. In *MFCS 2015*, volume 9234 of *Lecture Notes in Computer Science*, pages 243–255. Springer, 2015.

[GSH+17] François Gonze, Andrea Simonetto, Etienne Huens, Jean Boucquey, and Raphaël M. Jungers. Probabilistic occupancy counts and flight criticality measures for atm. In *12th USA/Europe ATM R&D Seminar*, 2017.

[GSH+18] François Gonze, Andrea Simonetto, Etienne Huens, Jean Boucquey, and Raphaël M. Jungers. Probabilistic occupancy counts and flight criticality measures in air traffic management. *Journal of Air Transportation*, 26(3):94–103, 2018.

[Hel15] Harald A. Helfgott. Growth in groups: ideas and perspectives. *Bulletin of the American Mathematical Society*, 52(3):357–413, 2015.

[HS14] Harald A. Helfgott and Ákos Seress. On the diameter of permutation groups. *Annals of mathematics*, 179(2):611–658, 2014.

[Jav02] Denis Javaux. A method for predicting errors when interacting with finite state systems. how implicit learning shapes the user's knowledge of a system. *Reliability Engineering & System Safety*, 75(2):147–165, 2002.

[Jun09] Raphaël M. Jungers. *The joint spectral radius: theory and applications*, volume 385. Springer Science & Business Media, 2009.

[Jun12] Raphaël M. Jungers. The synchronizing probability function of an automaton. *SIAM Journal on Discrete Mathematics*, 26(1):177–192, 2012.

[Jun14] Raphaël M. Jungers. Research note on the synchronizing probability function of an automaton. 2014.

[Kar01] Jarkko Kari. A counter example to a conjecture concerning synchronizing words in finite automata. *EATCS Bulletin*, 73:146, 2001.

[Kar03] Jarkko Kari. Synchronizing finite automata on Eulerian digraphs. *Theoretical Computer Science*, 295:223–232, 2003.

[Kar13] Jarkko Kari. Automata and formal languages. *University of Turku*, 2013.

[KRS87] A. A. Klyachko, Igor K. Rystsov, and M. A. Spivak. An extremal combinatorial problem associated with the bound on the length of a synchronizing word in an automaton. *Kibernetika*, 2:16–20, 1987.

[KS13] Andrzej Kisielewicz and Marek Szykuła. Generating small automata and the Černý conjecture. In *CIAA 2013*, volume 7982 of *Lecture Notes in Computer Science*, pages 340–348. Springer, 2013.

[KS15] Andrzej Kisielewicz and Marek Szykuła. Synchronizing automata with extremal properties. In *MFCS 2015*, volume 9234 of *Lecture Notes in Computer Science*, pages 331–343. Springer, 2015.

[Lae63] Arthur E. Laemmel. Study on application of coding theory. Technical report, POLYTECHNIC INST OF BROOKLYN NY MICROWAVE RESEARCH INST, 1963.

[Liu63] Chung Laung Liu. *Some memory aspects of finite automata*. PhD. thesis, MASSACHUSETTS INST OF TECH CAMBRIDGE RESEARCH LAB OF ELECTRONICS, 1963.

[Mar13] Pavel V. Martyugin. Careful synchronization of partial automata with restricted alphabets. In *CSR 2013*, volume 7913 of *Lecture Notes in Computer Science*, pages 76–87. Springer, 2013.

[Mas12] Marina I. Maslennikova. Reset complexity of ideal languages. In *SOFSEM 2012*, volume II of *Proceedings Institute of Computer Science Academy of Sciences of the Czech Republic*, pages 33–44, 2012.

[MN17] Juan Montoya and Christian Nolasco. On the synchronization of small sets of states. *Applied Mathematical Sciences*, 11(44):2151–2173, 2017.

[Moo56] Edward F. Moore. Gedanken-experiments on sequential machines. *Annals of mathematics studies*, 34:129–153, 1956.

[Nat86] Balas K. Natarajan. An algorithmic approach to the automated design of parts orienters. In *27th Annual Symposium on Foundations of Computer Science*, pages 132–142. IEEE, 1986.

[Nat89] Balas K. Natarajan. Some paradigms for the automated design of parts feeders. *The International journal of robotics research*, 8(6):98–109, 1989.

[OU10] Jörg Olschewski and Michael Ummels. The complexity of finding reset words in finite automata. In *MFCS 2010*, volume 6281 of *Lecture Notes in Computer Science*, pages 568–579, 2010.

[Pan15] Pavel Panteleev. Preset distinguishing sequences and diameter of transformation semigroups. In *LATA 2015*, volume 8977 of *Lecture Notes in Computer Science*, pages 353–364. Springer, 2015.

[PCC02] Charles Pecheur, Alessandro Cimatti, and Ro Cimatti. Formal verification of diagnosability via symbolic model checking. In *Workshop on Model Checking and Artificial Intelligence (MoChArt-2002), Lyon, France*, 2002.

[Pin78] Jean-Eric Pin. *Le problème de la synchronisation et la conjecture de Černý*. Thèse de 3ème cycle, Université Paris VI, 1978.

[Pin81] Jean-Eric Pin. Le problème de la synchronisation et la conjecture de Černỳ. In *Non-commutative structures in algebra and geometric combinatorics*, volume 109 of *Quaderni de la Ricerca Scientifica*, pages 37–48. CNR, Roma, 1981.

[Pin83] Jean-Eric Pin. On two combinatorial problems arising from automata theory. *Annals of Discrete Mathematics*, 17:535–548, 1983.

[Rig14a] Michel Rigo. *Formal Languages, automata and numeration systems 1: Introduction to combinatorics on words*, volume 1. John Wiley & Sons, 2014.

[Rig14b] Michel Rigo. *Formal languages, automata and numeration systems 2: Applications to recognizability and decidability*, volume 2. John Wiley & Sons, 2014.

[Rom08] Adam Roman. A note on Černý conjecture for automata over 3-letter alphabet. *Journal of Automata, Languages and Combinatorics*, 13(2):141–143, 2008.

[Rys92] Igor K. Rystsov. Rank of a finite automaton. *Cybernetics and Systems Analysis*, 28(3):323–328, 1992.

[Rys95] Igor K. Rystsov. Quasioptimal bound for the length of reset words for regular automata. *Acta Cybernetica*, 12(2):145–152, 1995.

[Rys97] Igor K. Rystsov. Reset words for commutative and solvable automata. *Theoretical Computer Science*, 172(1):273–279, 1997.

[Rys00] Igor K. Rystsov. Estimation of the length of reset words for automata with simple idempotents. *Cybernetics and Systems Analysis*, 36(3):339–344, 2000.

[Sal03] Arto Salomaa. Composition sequences for functions over a finite domain. *Theoretical Computer Science*, 292(1):263–281, 2003.

[SC04] Laurent Saloff-Coste. Random walks on finite groups. In *Probability on Discrete Structures*, volume 110 of *Encyclopaedia of Mathematical Sciences*, pages 263–346. Springer, 2004.

[Shi19] Yaroslav Shitov. An improvement to a recent upper bound for synchronizing words of finite automata. *arXiv preprint arXiv:1901.06542*, 2019.

[ST11] Evgeny Skvortsov and Evgeny Tipikin. Experimental study of the shortest reset word of random automata. In *CIAA 2011*, volume 6807 of *Lecture Notes in Computer Science*, pages 290–298. Springer, 2011.

[Ste11] Benjamin Steinberg. The averaging trick and the Černý conjecture. *International Journal of Foundations of Computer Science*, 22:1697–1706, 2011.

[Szy18] Marek Szykuła. Improving the Upper Bound on the Length of the Shortest Reset Word. In *STACS 2018*, volume 96 of *Leibniz International Proceedings in Informatics*, pages 56:1–56:13. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2018.

[Tra02] Avraham N. Trahtman. A package TESTAS for checking some kinds of testability. In *CIAA 2002*, volume 2608 of *Lecture Notes in Computer Science*, pages 228–232. Springer, 2002.

[Tra06] Avraham N. Trahtman. An efficient algorithm finds noticeable trends and examples concerning the černy conjecture. In *MFCS 2006*, volume 4162 of *Lecture Notes in Computer Science*, pages 789–800. Springer, 2006.

[Tra07] Avraham N. Trahtman. The Černý conjecture for aperiodic automata. *Discrete mathematics and Theoretical Computer Science*, 9(2):3–10, 2007.

[Tra09] Avraham N. Trahtman. The road coloring problem. *Israel Journal of Mathematics*, 172(2):51–60, 2009.

[Tra11] Avraham N. Trahtman. Modifying the upper bound on the length of minimal synchronizing word. In *FCT 2011*, volume 6914 of *Lecture Notes in Computer Science*, pages 173–180. Springer, 2011.

[Vol08] Mikhail V. Volkov. Synchronizing automata and the Černý conjecture. In *LATA 2008*, volume 5196 of *Lecture Notes in Computer Science*, pages 11–27. Springer, 2008.

[Vol09] Mikhail V. Volkov. Synchronizing automata preserving a chain of partial orders. *Theoretical Computer Science*, 410(37):3513–3519, 2009.

[Vor14] Vojtech Vorel. Subset synchronization of transitive automata. In *Proceedings of the 14th International Conference on Automata and Formal Languages*, pages 370–381, 2014.

[Vor16] Vojtech Vorel. Subset synchronization and careful synchronization of binary finite automata. *International Journal of Foundations of Computer Science*, 27(05):557–577, 2016.

[VR15] Vojtech Vorel and Adam Roman. Parameterized complexity of synchronization and road coloring. *Discrete Mathematics and Theoretical Computer Science*, 17:283–306, 2015.

[Zub98] Anatolii Yu. Zubov. On the diameter of the group S_N with respect to a system of generators consisting of a complete cycle and a transposition. *Trudy po Diskretnoi Matematike*, 2:112–150, 1998. In Russian.

Index