Synchronization problems in automata without non-trivial cycles

Andrew Ryzhikov

Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 00 (2018) 1–13

Procedia Computer Science

Synchronization Problems in Automata without Non-trivial Cycles

Andrew Ryzhikov

LIGM, Université Paris-Est, Marne-la-Vallée, France,
Université Grenoble Alpes, Laboratoire G-SCOP, Grenoble, France,
United Institute of Informatics Problems of NASB, Minsk, Belarus,
ryzhikov.andrew@gmail.com

Abstract

We study the computational complexity of various problems related to synchronization of weakly acyclic automata, a subclass of widely studied aperiodic automata. We provide upper and lower bounds on the length of a shortest word synchronizing a weakly acyclic automaton or, more generally, a subset of its states, and show that the problem of approximating this length is hard. We investigate the complexity of finding a synchronizable set of states of maximum size. We also show inapproximability of the problem of computing the rank of a subset of states in a binary weakly acyclic automaton and prove that several problems related to recognizing a synchronizable subset of states in such automata are NP-complete.

Keywords: synchronizing automaton, computational complexity, inapproximability, weakly acyclic automaton, subset rank, synchronizable set

1. Introduction

The concept of synchronization is widely studied in automata theory and has a lot of different applications in such areas as manufacturing, coding theory, biocomputing, semigroup theory and many others [1]. Let $A = (Q, \Sigma, \delta)$ be a complete deterministic finite automaton (which we simply call an *automaton* in this paper), where Q is a set of states, Σ is a finite alphabet and $\delta : Q \times \Sigma \to Q$ is a transition function. Note that our definition of an automaton does not include initial and accepting states. The function δ can be naturally extended to a mapping $Q \times \Sigma^* \to Q$, which we also denote as δ, in the following way: for $x \in \Sigma$ and $a \in \Sigma^*$ we recursively set $\delta(q, xa) = \delta(\delta(q, x), a)$. An automaton is called *synchronizing* if there exists a word that maps all its states to a fixed state. Such word is called a *synchronizing word*. A state $q \in Q$ is called a *sink state* if all letters from Σ map q to itself.

In this paper synchronization of weakly acyclic automata is studied. A *simple cycle* in an automaton $A = (Q, \Sigma, \delta)$ is a sequence $q_1, \dots, q_k$ of its states such that all the states in the sequence are different and there exist letters $x_1, \dots, x_k \in \Sigma$ such that $\delta(q_i, x_i) = q_{i+1}$ for $1 \leq i \leq k-1$ and $\delta(q_k, x_k) = q_1$. A simple cycle is a *self-loop* if it consists of only one state. An automaton is called *weakly acyclic* if all its simple cycles are self-loops. In other words, an automaton is weakly acyclic if and only if there exists an ordering $q_1, q_2, \dots, q_n$ of its states such that if $\delta(q_i, x) = q_j$ for some letter $x \in \Sigma$, then $i \leq j$ (such ordering is called a *topological sort*). Since a topological sort can be found in polynomial time [2], this class can be recognized in polynomial time. Weakly acyclic automata are called acyclic in [3] and partially ordered in [4], where the class of languages recognized by such automata is characterized.

Weakly acyclic automata arise naturally in synchronizing automata theory. For example, the automata in the reductions showing the hardness of the problem of finding a shortest synchronizing word of Eppstein [5] and Berlinkov [6] are weakly acyclic. More precise inapproximability results for this problem in weakly acyclic automata are provided in [7]. Surprisingly, most of the computational problems that are hard for general automata remain very hard in this class despite its very simple structure. Thus, investigation of weakly acyclic automata provides good lower bounds on the complexity of many problems for general automata. An automaton is called *aperiodic* if for any word $w \in \Sigma^*$ and any state $q \in Q$ there exists k such that $\delta(q, w^k) = \delta(q, w^{k+1})$, where w^k is a word obtained by k concatenations of w [8]. Obviously, weakly acyclic automata form a proper subclass of aperiodic automata, thus all hardness results hold for the class of aperiodic automata.

The concept of synchronization is often used as an abstraction of returning control over an automaton when there is no a priori information about its current state, but the structure of the automaton is known. If the automaton is synchronizing, we can apply a synchronizing word to it, and thus it will transit to a known state. If we want to perform the same operation when the current state is known to belong to some subset of states of the automaton, we come to the definition of a synchronizable set. A set $S \subseteq Q$ of states of an automaton A is called *synchronizable* if there exists a word $w \in \Sigma^*$ and a state $q \in Q$ such that the word w maps each state $s \in S$ to the state q. The word w is said to *synchronize* the set S. It follows from the definition that an automaton is synchronizing if and only if the set Q of all its states is synchronizable. Consider the problem Sync Set of deciding whether a given set S of states of an automaton A is synchronizable.

Sync Set
Input: An automaton A and a subset S of its states;
Output: Yes if S is a synchronizable set, No otherwise.

The Sync Set problem is PSPACE-complete [9, 10], even for binary strongly connected automata [11] (an automaton is called *binary* if its alphabet has size two, and *strongly connected* if any state can be mapped to any other state by some word). In [12] it is shown that the Sync Set problem is solvable in polynomial time for orientable automata if the cyclic order respected by the automaton is provided in the input. This problem is also solvable in polynomial time for monotonic automata [13]. The problem of deciding whether the whole set of states of an automaton is synchronizable is also solvable in polynomial time [1].

One of the most important questions in synchronizing automata theory is the famous Černý conjecture stating that any n-state synchronizing automaton has a synchronizing word of length at most $(n-1)^2$. The conjecture is proved for various special cases, including orientable, Eulerian, aperiodic and other automata (see [1] for references), but is still open in general. For more than 30 years, the best upper bound was $\frac{n^3-n}{6}$, obtained in [14]. Recently, a small improvement of this bound has been proved in [15]: the new bound is still cubic in n but improves the coefficient $\frac{1}{6}$ at n^3 by $\frac{4}{46875}$.

While there is a simple cubic bound on the length of a synchronizing word for the whole automaton, there exist examples of automata where the length of a shortest word synchronizing a subset of states is exponential in the number of states [11]. For orientable n-state automata, a tight upper bound of $(n-1)^2$ is known [5], and this bound is also asymptotically tight for monotonic automata [13]. On the other hand, a trivial upper bound $2^n - n - 1$ on the length of a shortest word synchronizing a subset of states in a n-state automaton is known [11]. In [16] Cardoso considers the length of a shortest word synchronizing a subset of states in a synchronizing automaton.

We assume that the reader is familiar with the notions of an NP-complete problem (refer to the book by Sipser [17]), an approximation algorithm and a gap-preserving reduction (for reference, see the book by Vazirani [18]).

Given an automaton A, the *rank* of a word w with respect to A is the number $|\{\delta(s, w) \mid s \in Q\}|$, i.e., the size of the image of Q under the mapping defined in A by w. More generally, the rank of a word w with respect to a subset S of states of A is the number $|\{\delta(s, w) \mid s \in S\}|$. The *rank* of an automaton (resp. of a subset of states) is the minimum among the ranks of all words $w \in \Sigma^*$ with respect to the automaton (resp. to the subset of states).

In this paper we provide various results concerning computational complexity and approximability of the problems related to subset synchronization in weakly acyclic automata. In Section 2 we prove some lower and upper bounds on the length of a shortest word synchronizing a weakly acyclic automaton or, more generally, a subset of its states. We also show that the problem of finding the length of a shortest word synchronizing a subset is hard to approximate. In Section 3 we study inapproximability of the problem of finding a subset of states of maximum size. In Section 4 we give strong inapproximability results for computing the rank of a subset of states in binary weakly acyclic automata.

In Section 5 we show that several other problems related to recognizing a synchronizable set in a weakly acyclic automaton are hard.

A preliminary conference version of this paper was published in [19].

2. Bounds on the Length of Shortest Synchronizing Words

Each synchronizing weakly acyclic automaton is a 0-automaton (i.e., an automaton with exactly one sink state), which gives an upper bound $\frac{n(n-1)}{2}$ on the length of a shortest synchronizing word [20]. The same bound can be deduced from the fact that each weakly acyclic automaton is aperiodic [8]. However, for weakly acyclic automata a more accurate result can be obtained, showing that weakly acyclic automata of rank r behave in a way similar to monotonic automata of rank r (see [21]).

Proposition 1. *Let $A = (Q, \Sigma, \delta)$ be a n-state weakly acyclic automaton, such that there exists a word of rank r with respect to A. Then there exists a word of length at most $n - r$ and rank at most r with respect to A.*

Proof. Observe that the rank of a weakly acyclic automaton is equal to the number of sink states in it. The conditions of the theorem imply that A has at most r sink states.

Consider the sets $S_1, \ldots, S_t$ constructed in the following way. Let p_i be the state in S_{i-1} with the smallest index in the topological sort such that p_i is not a sink state. Let x_i, $1 \leq i \leq t$, be a letter mapping the state p_i to some other state, where $S_i = \{\delta(q, x_i) \mid q \in S_{i-1}\}$, $1 \leq i \leq t$, and $S_0 = Q$. Since A has at most r sink states, we can thus construct step by step a word $w = x_1 \ldots x_t$ with $t \leq n - r$ that has rank at most r with respect to A. □

The following simple example shows that the bound is tight even for the case of alphabet of size one. Consider an automaton $A = (Q, \{x\}, \delta)$ with states $q_1, \ldots, q_n$. For the letter x define the transition function $\delta(q_i, x) = q_{i+1}$ for $1 \leq i \leq n - r$ and $\delta(q_i, x) = q_i$ for $n - r + 1 \leq i \leq n$. Obviously, A has rank r and shortest words of rank r with respect to A have length $n - r$. For an example with a larger alphabet one can duplicate the letter x.

Proposition 2. *Let S be a synchronizable set of states of size k in a weakly acyclic n-state automaton $A = (Q, \Sigma, \delta)$. Then the length of a shortest word synchronizing S is at most $\frac{k(2n-k-1)}{2}$.*

Proof. Consider a topological sort $q_1, \ldots, q_n$ of the set Q. Let q_s be a state such that all states in S can be mapped to it by a shortest word $w = x_1 \ldots x_t$. We can assume that the images of all words $x_1 \ldots x_j$, $j \leq t$, are pairwise distinct, otherwise some letter in this word can be removed. Then a letter x_j maps at least one state of the set $\{\delta(q, x_1 \ldots x_{j-1}) \mid q \in S\}$ to some other state. Thus the maximum total number of letters in w sending all states in S to q_s is at most $(n - k) + (n - k + 1) + \ldots + (n - 1) = \frac{k(2n-k-1)}{2}$, since application of each letter of w increases the sum of the indices of reached states by at least one. □

Consider a binary automaton $A = (Q, \{0, 1\}, \delta)$ with n states $q_1, \ldots, q_{k-1}$, $s_1, \ldots, s_\ell$, t, where $\ell = n - k$. Define $\delta(q_i, 0) = q_i$, $\delta(q_i, 1) = q_{i+1}$ for $1 \leq i \leq k - 2$, $\delta(q_{k-1}, 1) = s_1$. Define also $\delta(s_i, 0) = s_{i+1}$ for $1 \leq i \leq \ell - 1$, $\delta(s_i, 1) = t$ for $1 \leq i \leq \ell - 1$. Define both transitions for s_ℓ and t as self-loops. Set $S = \{q_1, \ldots, q_{k-1}, s_\ell\}$. The shortest word synchronizing S is $(10^{l-1})^{k-1}$ of length $(k - 1)(n - k)$. The automaton in this example is binary weakly acyclic, and even has rank 2. Figure 1 gives the idea of the described construction.

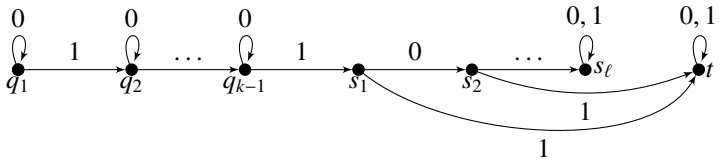

Figure 1. The automaton providing the lower bound for subset synchronization

As was noted by an anonymous reviewer, for alphabet of size $n - 2$, a better lower bound of $\frac{(k-1)(2n-k-2)}{2}$ can be shown as follows. Let $Q = \{-1, 0, 1, \ldots, n - 2\}$, $\Sigma = \{a_1, \ldots, a_{n-2}\}$,

$$\delta(k, a_i) = \begin{cases} k & \text{if } k > i, \\ k-1 & \text{if } k = i, \\ -1 & \text{if } 0 < k < i, \\ k & \text{if } k \in \{-1, 0\} \end{cases}$$

If $k < n$ and $S = \{0, n-2, n-3, \ldots, n-k\}$, then it is easy to see that the shortest word synchronizing S has length $(n-k) + (n-k+1) + \ldots + (n-2) = \frac{(k-1)(2n-k-2)}{2}$. For each n and k, this is less than the upper bound of Proposition 2 by $n-1$ only.

Consider now the following problem.

Shortest Set Sync Word
Input: An automaton A and a synchronizable subset S of its states;
Output: The length of a shortest word synchronizing S.

It follows from Proposition 2 that the decision version of this problem (asking whether there exists a word of length at most k synchronizing S) is in NP for weakly acyclic automata, so it is reasonable to investigate its approximability.

Theorem 1. *The* Shortest Set Sync Word *problem for n-state binary weakly acyclic automata cannot be approximated in polynomial time within a factor of $O(n^{\frac{1}{2}-\epsilon})$ for any $\epsilon > 0$ unless P = NP.*

Proof. To prove this theorem, we construct a gap-preserving reduction from the Shortest Sync Word problem in p-state binary automata, which cannot be approximated in polynomial time within a factor of $O(p^{1-\epsilon})$ for any $\epsilon > 0$ unless P = NP [22]. Let a binary automaton $A = (Q, \{0,1\}, \delta)$ be the input of Shortest Sync Word. Let $Q = \{q_1, \ldots, q_p\}$. Construct a binary automaton $A' = (Q', \{0,1\}, \delta')$ with the set of states $Q' = \{q_i^{(j)} \mid 1 \le i \le p, 1 \le j \le p+1\}$. Define $\delta'(q_i^{(j)}, x) = q_k^{(j+1)}$ for $1 \le i \le p$, $1 \le j \le p$, $x \in \{0,1\}$, where k is such that $q_k = \delta(q_i, x)$. Define $\delta'(q_i^{(p+1)}, x) = q_i^{(p+1)}$ for $1 \le i \le p$ and $x \in \{0,1\}$. Take $S' = \{q_i^{(1)} \mid 1 \le i \le p\}$.

Observe that any word synchronizing S' in A' is a synchronizing word for A because of the definition of δ'. In the other direction, we note that a shortest synchronizing word for a p-state automaton in the construction of Gawrychowski and Straszak [22] has length at most p. Hence, a shortest synchronizing word for A also synchronizes S' in A'. Thus, the length of a shortest synchronizing word for A is equal to the length of a shortest word synchronizing S' in A', and we get a gap-preserving reduction with gap $O(p^{1-\epsilon}) = O(n^{\frac{1}{2}-\epsilon})$, as A' has $O(p^2)$ states. Finally, it is easy to see that A' is binary weakly acyclic. □

3. Finding a Synchronizable Set of Maximum Size

One possible approach to measure and reduce initial state uncertainty in an automaton is to find a subset of states of maximum size where the uncertainty can be resolved, i.e., to find a synchronizable set of maximum size. This is captured by the following problem.

Max Sync Set
Input: An automaton A;
Output: The maximum cardinality of a synchronizable set of states in A.

Türker and Yenigün [23] study a variation of this problem, which is to find a set of states of maximum size that can be mapped by some word to a subset of a given set of states in a given monotonic automaton. They reduce the N-Queens Puzzle problem [24] to this problem to prove its NP-hardness. However, their proof is unclear, since the input has size $O(\log N)$, and the output size is polynomial in N. Also, the N-Queens Puzzle problem is solvable in polynomial time [24].

First we investigate the PSPACE-completeness of the decision version of the Max Sync Set problem, which we will denote as Max Sync Set-D. Its formulation is the following: given an automaton A and a number c, decide whether there is a synchronizable set of states of cardinality at least c in A.

Proposition 3. *The* Max Sync Set-D *problem is PSPACE-complete for binary automata.*

Proof. The Sync Set problem is in PSPACE [10]. Thus, the Max Sync Set-D problem is also in PSPACE, as we can sequentially check whether each subset of states is synchronizable and compare the size of a maximum synchronizable set to c.

To prove that the Max Sync Set-D problem is PSPACE-hard for binary automata, we will reduce a PSPACE-complete Sync Set problem for binary automata [11] to it. Let an automaton A and a subset S of its states be an input to Sync Set. Let n be the number of states of A. Construct a new automaton A' by initially taking a copy of A. For each state $s \in S$, add $n+1$ *new* states to A' and define all the transitions from these new states to map to s, regardless of the input letter. Define the set S' to be a union of all new states and take $c = |S'| = (n+1)|S|$.

Let S_1 be a maximum synchronizable set in A not containing at least one new state q. As S_1 is maximum, it does not contain other n new states that can be mapped to the same state as q. Thus, the size of S_1 is at most $n + (n+1)|S| - (n+1) < (n+1)|S| = c$. Hence, each synchronizable set of size at least c in A' contains S'. The set S is synchronizable in A if and only if S' is synchronizable in A', as each word w synchronizing S in A corresponds to a word xw synchronizing S' in A', where x is an arbitrary letter. Thus, A' has a synchronizable set of size at least c if and only if S is synchronizable in A. □

Now we proceed to inapproximability results for the Max Sync Set problem in several classes of automata. We will need some results from graph theory. An *independent set* I in a graph G is a set of its vertices such that no two vertices in I share an edge. The size of a maximum independent set in G is denoted $\alpha(G)$. The Independent Set problem is defined as follows.

Independent Set
Input: A graph G;
Output: An independent set of maximum size in G.

Zuckerman [25] has proved that, unless P = NP, there is no polynomial $p^{1-\varepsilon}$-approximation algorithm for the Independent Set problem for any $\varepsilon > 0$, where p is the number of vertices in G.

Proposition 4. *The problem* Max Sync Set *for weakly acyclic n-state automata over an alphabet of cardinality $O(n)$ cannot be approximated in polynomial time within a factor of $O(n^{1-\varepsilon})$ for any $\varepsilon > 0$ unless P = NP.*

Proof. We will prove this theorem by constructing a gap-preserving reduction from the Independent Set problem. Given a graph $G = (V, E)$, $V = \{v_1, v_2, \ldots, v_p\}$, we construct an automaton $A = (Q, \Sigma, \delta)$ as follows. For each $v_i \in V$, we add two states s_i, t_i in Q. We also add a state f to Q. Thus, $|Q| = 2p + 1$. The alphabet Σ consists of letters $\tilde{v}_1, \ldots, \tilde{v}_p$ corresponding to the vertices of G.

The transition function δ is defined in the following way. For each $1 \leq i \leq p$, the state s_i is mapped to f by the letter $\tilde{v}_i$. For each $v_i v_j \in E$ the state s_i is mapped to t_i by the letter $\tilde{v}_j$, and the state s_j is mapped to t_j by the letter $\tilde{v}_i$. All yet undefined transitions map a state to itself.

Lemma 1. *Let I be a maximum independent set in G. Then the set $S = \{s_i \mid v_i \in I\} \cup \{f\}$ is a synchronizable set of maximum cardinality (of size $\alpha(G) + 1$) in the automaton $A = (Q, \Sigma, \delta)$.*

Proof. Let w be a word obtained by concatenating the letters corresponding to I in arbitrary order. Then w synchronizes the set $S = \{s_i \mid v_i \in I\} \cup \{f\}$ of states of cardinality $|I| + 1$. Thus, A has a synchronizable set of size at least $\alpha(G) + 1$.

In other direction, let w be a word synchronizing a set of states S' of maximum size in A. We can assume that after reading w all the states in S' are mapped to f, as all the sets of states that are mapped to any other state have cardinality at most two. Then there are no edges in G between any pair of vertices in $I' = \{v_i \mid s_i \in S'\}$, since each time some state s_i is mapped to f, each state s_j such that v_j and v_i share an edge in G, is mapped to t_j. Hence I' is an independent set of size $|S'| - 1$ in G. Thus the maximum size of a synchronizable set in A is equal to $\alpha(G) + 1$. □

Thus we have a gap-preserving reduction from the Independent Set problem to the Max Sync Set problem with a gap $\Theta(p^{1-\varepsilon})$ for any $\varepsilon > 0$. It is easy to see that $n = \Theta(p)$ and A is weakly acyclic, which concludes the proof of the theorem. □

Next we move to a weaker inapproximability result for binary automata.

Proposition 5. *The problem* Max Sync Set *for binary n-state automata cannot be approximated in polynomial time within a factor of $O(n^{\frac{1}{2}-\varepsilon})$ for any $\varepsilon > 0$ unless P = NP.*

Proof. Again, we construct a gap-preserving reduction from the Independent Set problem extending the proof of Proposition 4. Given a graph $G = (V, E), V = \{v_1, v_2, \ldots, v_p\}$, we construct an automaton $A = (Q, \Sigma, \delta)$ in the following way. Let $\Sigma = \{0, 1\}$. First we construct the main gadget A_{main} having a synchronizable set of states of size $\alpha(G)$. For each vertex $v_i \in V, 1 \le i \le p$, we construct a set of new states $L_i = V_i \cup U_i$ in Q, where $V_i = \{v_j^{(i)} : 1 \le j \le p\}, U_i = \{u_j^{(i)} : 1 \le j \le p\}$. We call L_i the ith *layer* of A_{main}. We also add a state f to Q. For each i, $1 \le i \le p$, the transition function δ imitates taking the vertices $v_1, v_2, \ldots, v_p$ into an independent set one by one and is defined as:

$$\delta(v_j^{(i)}, 0) = \begin{cases} u_j^{(i)} & \text{if } i = j, \\ v_j^{(i+1)} & \text{otherwise} \end{cases}$$

$$\delta(v_j^{(i)}, 1) = \begin{cases} u_j^{(i)} & \text{if there is an edge } v_i v_j \in E, \\ v_j^{(i+1)} & \text{otherwise} \end{cases}$$

Here all $v_j^{(n+1)}, 1 \le j \le p$, coincide with f. For each state $u_j^{(i)}$, the transitions for both letters 0 and 1 lead to the originating state (i.e. they are self-loops).

We also add a p-state cycle A_{cycle} attached to f. It is a set of p states $c_1, \ldots, c_p$, mapping c_i to c_{i+1} and c_p to c_1 regardless of the input symbol. Finally, we set c_1 to coincide with f. Thus we get the automaton A_1. Figure 2 presents an example of A_1 for a graph with three vertices v_1, v_2, v_3 and one edge $v_2 v_3$.

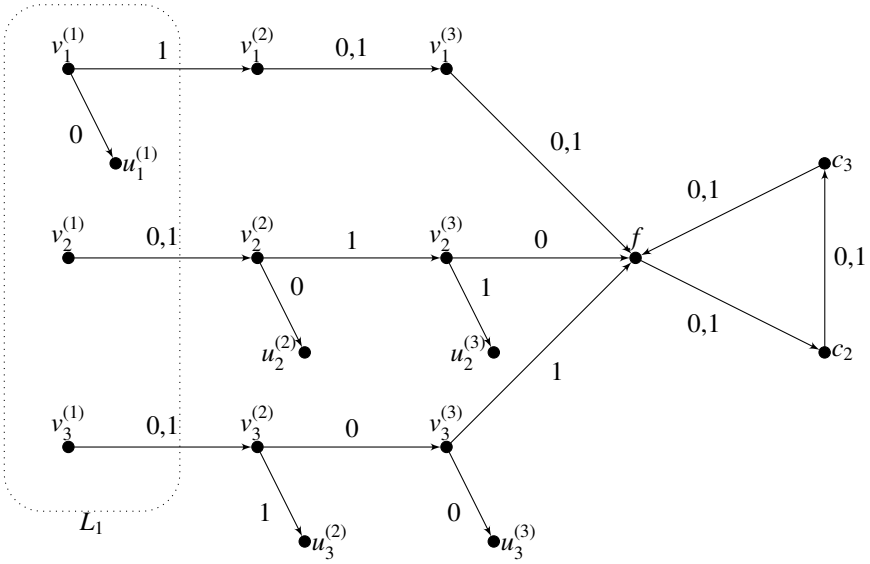

Figure 2. An example of A_1. Unachievable states and self-loops are omitted.

The main property of A_1 is claimed by the following lemma.

Lemma 2. *The size of a maximum synchronizable set of states from the first layer in A_1 equals $\alpha(G)$.*

Proof. Let I be a maximum independent set in G. Consider a word w of length p such that its ith letter is equal to 0 if $v_i \notin I$ and to 1 if $v_i \in I$. By the construction of A_1, this word synchronizes the set $\{v_j^{(1)} \mid v_j \in I\}$.

Conversely, a synchronizable set of at least three states from the first layer can be mapped only to some vertex of A_{cycle}, and the corresponding set of vertices in G is an independent set, since application of each letter corresponds to taking or not taking a vertex into an independent set, and after taking some vertex all the states corresponding to the neighbors of this vertex are sent to separate sink states. □

Some layer in the described construction can contain a synchronizable subset of size larger than the maximum synchronizable subset of the first layer. To avoid that, we modify A_1 by repeating each state (with all transitions) of the first layer p times. More formally, we replace each pair of states $v_j^{(1)}$, $u_j^{(1)}$ with p different pairs of states such that in each pair all the transitions repeat the transitions between $v_j^{(1)}$, $u_j^{(1)}$, and all the other states of the automaton. We denote the automaton thus constructed as A.

The following lemma claims that the described procedure of constructing A from G is a gap-preserving reduction from the Independent Set problem in graphs to the Max Sync Set problem in binary automata.

Lemma 3. *If $\alpha(G) > 1$, then the maximum size of a synchronizable set in A is equal to $p\alpha(G) + 1$.*

Proof. Note that since the length of the cycle in A_{cycle} equals p, each synchronizable set of A is either a subset of a single layer of A together with a state in A_{cycle} (if this set is first mapped to f) or a subset of a set $\{v_j^{(i)} \mid 2 \le i \le \ell\} \cup \{u_j^{(\ell)}\}$ for some ℓ and j, together with p new states that replaced $v_j^{(1)}$ (if this set is mapped to $u_j^{(\ell)}$). Consider the first case. If some maximum synchronizable set S contains a state from the ith layer of A and $i > 1$, then its size is at most $p+1$. A maximum synchronizable set containing some states from the first layer of A consists of $p\alpha(G)$ states from this layer (according to Lemma 2) and some state of A_{cycle}, so this set has size $p\alpha(G) + 1 \ge 2p + 1$. In the second case, the maximum size of a synchronizable set is at most $p + (p-1) + 1 = 2p < p\alpha(G) + 1$. □

It is easy to see that the constructed reduction is gap-preserving with a gap $\Theta(p^{1-\varepsilon}) = \Theta(n^{\frac{1}{2}-\varepsilon})$, where n is the number of states in A, as $n = \Theta(p^2)$. Thus the Max Sync Set for n-state binary automata cannot be approximated in polynomial time within a factor of $O(n^{\frac{1}{2}-\varepsilon})$ for any $\varepsilon > 0$ unless P = NP, which concludes the proof of the theorem. □

Proposition 5 can also be proved by using Proposition 4 and a slight modification of the technique used in [11] for decreasing the size of the alphabet. However, in this case the resulting automaton is far from being weakly acyclic, while the automaton in the proof of Proposition 4 has only one cycle. The next theorem shows how to modify our technique to prove an inapproximability bound for Max Sync Set in binary weakly acyclic automata.

Theorem 2. *The* Max Sync Set *problem for binary weakly acyclic n-state automata cannot be approximated in polynomial time within a factor of $O(n^{\frac{1}{3}-\varepsilon})$ for any $\varepsilon > 0$ unless P = NP.*

Proof. We modify the construction of the automaton A_{main} from Proposition 5 in the following way. We repeat each state (with all transitions) of the first layer p^2 times in the same way as it is done in the proof of Proposition 5. Thus we get a weakly acyclic automaton A_{wa} with $n = \Theta(p^3)$ states, where p is the number of vertices in the graph G. Furthermore, similar to Lemma 3, the size of the maximum synchronizable set of states in A_{wa} is between $p^2\alpha(G)$ and $p^2\alpha(G) + p(p-1) + 1$, because some of the states from the layers other than the first can be also mapped to f. Both of the values are of order $\Theta(p^2\alpha(G))$, thus we have an gap-preserving reduction providing the inapproximability within a factor of $O(p^{1-\varepsilon}) = O(n^{\frac{1}{3}-\varepsilon})$ for any $\varepsilon > 0$, where n is the number of states in A_{wa}. □

We finish by noting that for two classes of automata the Max Sync Set problem is solvable in polynomial time.

Proposition 6. *The problem* Max Sync Set *can be solved in polynomial time for unary automata.*

Proof. Consider the digraph G induced by states and transitions of an unary automaton A. By definition, each vertex of G has outdegree 1. Thus, the set of the vertices of G can be partitioned into directed cycles and a set of vertices not belonging to any cycle, but lying on a directed path leading to some cycle. Let n be the number of states in A. It is easy to see that after performing n transitions, each state of A is mapped into a state in some cycle, and all further transitions will not map any two different states to the same state. Thus, it is enough to perform n transitions and select such state s that the maximum number of states are mapped to s. □

A more interesting case is covered by the following proposition. An automaton $A = (Q, \Sigma, \delta)$ is called *Eulerian* if there exists k such that for each state $q \in Q$ there are exactly k pairs (q', a), $q' \in Q$, $a \in \Sigma$, such that $\delta(q', a) = q$.

Proposition 7. *The problem* Max Sync Set *can be solved in polynomial time for Eulerian automata.*

Proof. According to Theorem 2.1 in [26] (see also [27] for the discussion of the Eulerian case), each word of minimum rank with respect to an Eulerian automaton synchronizes the sets $S_1, S_2, \ldots, S_r$ forming a partition of the states of the automaton into inclusion-maximal synchronizable sets. Moreover, according to this theorem all inclusion-maximal synchronizable sets in an Eulerian automaton are of the same size, thus each inclusion-maximal synchronizable set has maximum cardinality. A word of minimum rank with respect to an automaton can be found in polynomial time [28], which concludes the proof. □

4. Computing the Rank of a Subset of States

Assume that we know that the current state of the automaton A belongs to a subset S of its states. Even if it is not possible to synchronize S, it can be reasonable to minimize the size of the set of possible states of A, reducing the uncertainty of the current state as much as possible. One way to do it is to map S to a set S' of smaller size by applying some word to A. Recall that the size of the smallest such set S' is called the rank of S. Consider the following problem of finding the rank of a subset of states in a given automaton.

Set Rank
Input: An automaton A and a set S of its states;
Output: The rank of S in A.

The rank of an automaton, that is, the rank of the set of its states, can be computed in polynomial time [28]. However, since the automaton in the proof of PSPACE-completeness of Sync Set in [9] has rank 2 (and thus each subset of states in this automaton has rank either 1 or 2), it follows immediately that there is no polynomial c-approximation algorithm for the Set Rank problem for any $c < 2$ unless P = PSPACE. It also follows that checking whether the rank of a subset of states equals the rank of the whole automaton is PSPACE-complete. For monotonic weakly acyclic automata, this problem is hard to approximate within a factor of $\frac{9}{8} - \epsilon$ for any $\epsilon > 0$ [13]. For general weakly acyclic automata it is possible to get much stronger bounds, as it is shown by the results of this section.

We will need the Chromatic Number problem. A *proper coloring* of a graph $G = (V, E)$ is a coloring of the set V in such a way that no two adjacent vertices have the same color. The chromatic number of G, denoted $\chi(G)$, is the minimum number of colors in a proper coloring of G. Recall that a set of vertices in a graph is called *independent* if no two vertices in this set are adjacent. A proper coloring of a graph can be also considered as a partition of the set of its vertices into independent sets.

Chromatic Number
Input: A graph G;
Output: The chromatic number of G.

This problem cannot be approximated within a factor of $O(p^{1-\epsilon})$ for any $\epsilon > 0$ unless P = NP, where p is the number of vertices in G [25].

Proposition 8. *The* Set Rank *problem for n-state weakly acyclic automata with alphabet of size* $O(\sqrt{n})$ *cannot be approximated within a factor of* $O(n^{\frac{1}{2}-\epsilon})$ *for any* $\epsilon > 0$ *unless P = NP.*

Proof. We will prove this theorem by constructing a gap-preserving reduction from the Chromatic Number problem, extending the technique in the proof of Proposition 4. Given a graph $G = (V, E)$, $V = \{v_1, v_2, \ldots, v_p\}$, we construct an automaton $A = (Q, \Sigma, \delta)$ as follows. The alphabet Σ consists of letters $\tilde{v}_1, \ldots, \tilde{v}_p$ corresponding to the vertices of G, together with a *switching* letter ν. We use p identical *synchronizing* gadgets $T^{(k)}$, $1 \le k \le p$, such that each gadget synchronizes a subset of states corresponding to an independent set in G. Gadget $T^{(k)}$ consists of a set $\{s_i^{(k)}, t_i^{(k)} \mid 1 \le i \le p\} \cup \{f^{(k)}\}$ of states.

The transition function δ is defined as follows. For each gadget $T^{(k)}$, for each $1 \le i \le p$, the state $s_i^{(k)}$ is mapped to $f^{(k)}$ by the letter $\tilde{v}_i$. For each $v_i v_j \in E$ the state $s_i^{(k)}$ is mapped to $t_i^{(k)}$ by the letter $\tilde{v}_j$, and the state $s_j^{(k)}$ is mapped to $t_j^{(k)}$ by the letter $\tilde{v}_i$. All yet undefined transitions corresponding to the letters $\tilde{v}_1, \ldots, \tilde{v}_p$ map a state to itself.

It remains to define the transitions corresponding to ν. For each $1 \le k \le p-1$, ν maps $t_i^{(k)}$ and $s_i^{(k)}$ to $s_i^{(k+1)}$, and $f^{(k)}$ to itself. Finally, ν acts on all states in $T^{(p)}$ as a self-loop.

Define $S = \{s_i^{(1)} \mid 1 \le i \le p\}$. We will prove that the rank of S is equal to the chromatic number of G. Consider a proper coloring of G with the minimum number of colors and let $I_1 \cup \ldots \cup I_{\chi(G)}$ be the partition of G into independent sets defined by this coloring. For each I_j, consider a word w_j obtained by concatenating the letters corresponding

to the vertices in I_j in some order. Consider now the word $w_1 v w_2 v \dots v w_{\chi(G)}$. This word maps the set S to the set $\{f^{(i)} \mid 1 \le i \le \chi(G)\}$, which proves that the rank of S is at most $\chi(G)$.

In the other direction, note that after each reading of v all states except $f^{(k)}, 1 \le k \le p-1$, are mapped to the next synchronizing gadget (except the last gadget $T^{(p)}$ which is mapped to itself). Note that each time a state $s_i^{(k)}$ is mapped to $f^{(k)}$, each state $s_j^{(k)}$ such that v_j and v_i share an edge in G, is mapped to $t_j^{(k)}$. Thus, only a subset of states corresponding to an independent set of vertices can be mapped to some particular $f^{(k)}$, and the image of S after reading any word is a subset of the states in some gadget together with some of the states $f^{(k)}, 1 \le k \le p$. Hence, the rank of S is at least $\chi(G)$.

Thus we have a gap-preserving reduction from the Chromatic Number problem to the Set Rank problem with gap $\Theta(p^{1-\varepsilon})$ for any $\varepsilon > 0$. It is easy to see that $n = \Theta(p^2)$, A is weakly acyclic and its alphabet has size $O(\sqrt{n})$, which finishes the proof of the theorem. □

Using the classical technique of reducing the alphabet size (see [11]), $O(n^{\frac{1}{3}-\epsilon})$ inapproximability can be proved for binary automata. To prove the same bound for binary weakly acyclic automata, we have to refine the technique of the proof of the previous theorem.

Theorem 3. *The* Set Rank *problem for n-state binary weakly acyclic automata cannot be approximated within a factor of* $O(n^{\frac{1}{3}-\epsilon})$ *for any* $\epsilon > 0$ *unless P = NP.*

Proof. To prove this theorem we construct a gap-preserving reduction from the Chromatic Number problem, extending the proof of the previous theorem.

Given a graph $G = (V, E), V = \{v_1, v_2, \dots, v_p\}$, we construct an automaton $A = (Q, \{0, 1\}, \delta)$. In our reduction we use two kinds of gadgets: *p synchronizing gadgets* $T^{(k)}$, $1 \le k \le p$, and *p waiting gadgets* $R^{(k)}$, $1 \le k \le p$. Gadget $T^{(k)}$ consists of a set $\{v_{i,j}^{(k)} \mid 1 \le i, j \le p\}$ of states, together with a state $f^{(k)}$, and $R^{(k)}$, $1 \le k \le p$, consists of the set $\{u_{i,j}^{(k)} \mid 1 \le i, j \le p\}$.

For each i, j, k, $1 \le i, j, k \le p$, the transition function δ is defined as:

$$\delta(v_{i,j}^{(k)}, 0) = \begin{cases} u_{i,j}^{(k)} & \text{if } i = j, \\ v_{i+1,j}^{(k)} & \text{otherwise} \end{cases}$$

$$\delta(v_{i,j}^{(k)}, 1) = \begin{cases} u_{i,j}^{(k)} & \text{if there is an edge } v_i v_j \in E, \\ v_{i+1,j}^{(k)} & \text{otherwise} \end{cases}$$

Here all $v_{p+1,j}^{(k)}$, $1 \le j \le p$, coincide with $f^{(k)}$. We set $\delta(u_{i,j}^{(k)}, x) = u_{i+1,j}^{(k)}$ for $x \in \{0, 1\}$, $1 \le i, k \le p-1$, $1 \le j \le p$, and $\delta(u_{p,j}^{(k)}, x) = v_{1,j}^{(k+1)}$ for $1 \le j \le p$, $1 \le k \le p-1$, $x \in \{0, 1\}$. The states $u_{i,j}^{(p)}$ are sink states: both letters 0 and 1 act on them as self-loops. Finally, we set $S = \{v_{1,j}^{(1)} \mid 1 \le j \le p\}$. Figure 3 gives an idea of the described construction.

The idea of the presented construction is essentially a combination of the ideas in the proofs of Proposition 5 and Proposition 8, so we provide only a sketch of the proof. A synchronizing gadget $T^{(k)}$ synchronizes a set $S^{(k)} \subseteq S$ of states corresponding to some independent set in G. All the states corresponding to the vertices adjacent to vertices corresponding to $S^{(k)}$ are mapped to the corresponding waiting gadget $R^{(k)}$, and get to the next synchronizing gadget $T^{(k+1)}$ only after the states of $S^{(k)}$ are synchronized (and thus mapped to $f^{(k)}$). Hence, the minimum size of a partition of V into independent sets is equal to the rank of S. The number of states in A is $O(p^3)$. Thus, we get $\Theta(n^{\frac{1}{3}-\epsilon})$ inapproximability. □

5. Subset Synchronization

In this section, we obtain complexity results for several problems related to subset synchronization in weakly acyclic automata. We adapt Eppstein's construction from [5], which is a powerful and flexible tool for such proofs. We will need the following NP-complete SAT problem [17].

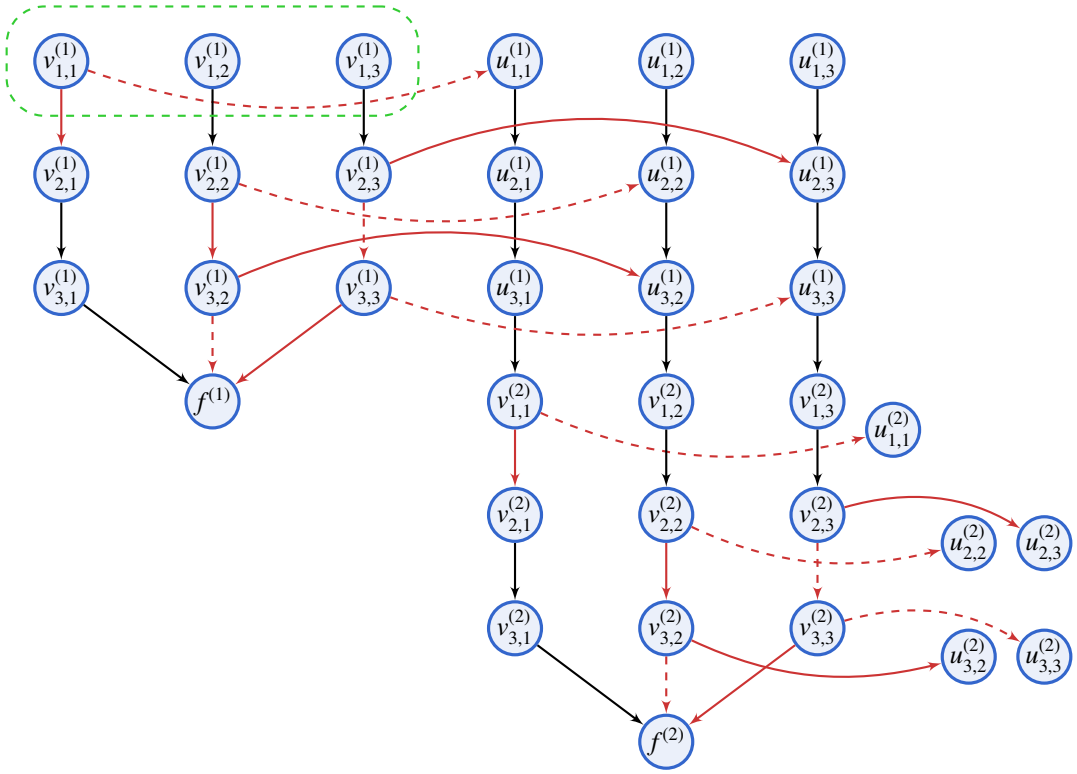

Figure 3. A part of the construction in the reduction for Set Rank. Red dashed arrows represent transitions for letter 0, red solid arrows – for letter 1, black arrows – for both letters. Self-loops are omitted.

SAT
Input: A set X of n boolean variables and a set C of m clauses;
Output: Yes if there exists an assignment of values to the variables in X such that all clauses in C are satisfied, No otherwise.

Theorem 4. *The* Sync Set *problem in binary weakly acyclic automata is NP-complete.*

Proof. Because of the polynomial upper bound on the length of a shortest word synchronizing a subset of states proved in Proposition 2, we can use such word as a certificate. Thus, the problem is in NP.

We reduce the SAT problem. Given X and C, we construct an automaton $A = (Q, \{0,1\}, \delta)$. For each clause c_j, we add $n+1$ states $y^{(j)}_i$, $1 \le i \le n+1$, to Q. We introduce also a state $f \in Q$. The transitions from $y^{(j)}_i$ correspond to the occurrence of x_i in c_j in the following way: for $1 \le i \le n$, $1 \le j \le m$, $\delta(y^{(j)}_i, a) = f$ if the assignment $x_i = a$, $a \in \{0,1\}$, satisfies c_j, and $\delta(y^{(j)}_i, a) = y^{(j)}_{i+1}$ otherwise. The transition function δ also maps $y^{(j)}_{n+1}$ to itself for all $1 \le j \le m$ and both letters 0 and 1.

Let $S = \{y^{(j)}_1 \mid 1 \le j \le m\}$. The word $w = a_1 a_2 \dots a_n$ synchronizes S if a_i is the value of x_i in an assignment satisfying C, and vice versa. Thus, the set is synchronizable if and only if all clauses in C can be satisfied by some assignment of binary values to the variables in X. □

By identifying the states $y^{(j)}_{n+1}$ for $1 \le j \le m$ and adding f to S it is also possible to prove that the problem of checking whether the rank of a subset of states equals the rank of an automaton is coNP-complete for binary weakly acyclic automata (cf. the remarks in the beginning of Section 4).

The proof of Theorem 4 can be used to prove the hardness of a special case of the following problem, which is PSPACE-complete in general [29] and NP-complete for weakly acyclic monotonic automata over a three-letter alphabet [13].

FINITE AUTOMATA INTERSECTION
Input: Automata $A_1, \ldots, A_k$ (with initial and accepting states);
Output: Yes if there is a word which is accepted by all automata, No otherwise.

Proposition 9. FINITE AUTOMATA INTERSECTION *is NP-complete when all automata in the input are binary weakly acyclic.*

Proof. Observe first that if there exists a word which is accepted by all automata, then a shortest such word w has length at most linear in the total number of states in all automata. Indeed, for each automaton consider a topological sort of the set of its states. Each letter of w maps at least one state in some automaton to some other state, which has larger index in the topological sort of the set of states of this automaton. Thus, the considered problem is in NP.

For the hardness proof, we use the same construction as in Theorem 4. Provided X and C, define A in the same way as in Theorem 4. Define $A_j = (Q_j, \{0,1\}, \delta_j)$ as follows. Take $Q_j = \{y_i^{(j)}, 1 \leq i \leq n+1\} \cup \{f\}$ and δ_j to be the restriction of δ to the set Q_j. Set $y_1^{(j)}$ to be the input state and f to be the only accepting state of A_j. Then there exists a word accepted by automata $A_1, \ldots, A_m$ if and only if all clauses in C are satisfiable by some assignment. □

To obtain the next results, we will need a modified construction of the automaton from the proof of Theorem 4, as well as some new definitions. A *partial automaton* is a triple (Q, Σ, δ), where Q and Σ are the same as in the definition of a finite deterministic automaton, and δ is a partial transition function (i.e., a transition function which may be undefined for some argument values). Given an instance of the SAT problem, construct a partial automaton $A_{base} = (Q, \{0,1\}, \delta)$ as follows. We introduce a state $f \in Q$. For each clause c_j, we add $n+1$ states $y_i^{(j)}$, $1 \leq i \leq n+1$, to Q. For each c_j, add also states $z_i^{(j)}$ for $h_i + 1 \leq i \leq n+1$, where h_i is the smallest index of a variable occurring in c_j. The transitions from $y_i^{(j)}$ correspond to the occurrence of x_i in c_j in the following way: for $1 \leq i \leq n$, $\delta(y_i^{(j)}, a) = z_{i+1}^{(j)}$ if the assignment $x_i = a$, $a \in \{0,1\}$, satisfies c_j, and $\delta(y_i^{(j)}, a) = y_{i+1}^{(j)}$ otherwise. For $a \in \{0,1\}$, we set $\delta(z_i^{(j)}, a) = z_{i+1}^{(j)}$ for $h_i + 1 \leq i \leq n$, $1 \leq j \leq m$. The transition function δ also maps $z_{n+1}^{(j)}$, $1 \leq j \leq m$, and f to f for both letters 0 and 1.

A word w is said to *carefully synchronize* a partial automaton A if it maps all its states to the same state q, and each mapping corresponding to a prefix of w is defined for each state. The automaton A is then called *carefully synchronizing*. We use A_{base} to prove the hardness of the following problem.

CAREFUL SYNCHRONIZATION
Input: A partial automaton A;
Output: Yes if A is carefully synchronizing, No otherwise.

For binary automata, CAREFUL SYNCHRONIZATION is PSPACE-complete [30]. For monotonic automata over a four-letter alphabet it is NP-hard. We call a partial automaton *aperiodic* if for any word $w \in \Sigma^*$ and any state $q \in Q$ there exists k such that either $\delta(q, w^k)$ is undefined, or $\delta(q, w^k) = \delta(q, w^{k+1})$.

Proposition 10. CAREFUL SYNCHRONIZATION *is NP-hard for aperiodic partial automata over a three-letter alphabet.*

Proof. We reduce the SAT problem. Given X and C, we first construct A_{base}. Then we add an additional letter r to the alphabet of A_{base} and introduce m new states $s^{(j)}$, $1 \leq j \leq m$. For $1 \leq i \leq n$, $1 \leq j \leq m$, we define $\delta(s^{(j)}, r) = y_1^{(j)}$, $\delta(y_i^{(j)}, r) = y_1^{(j)}$, $\delta(z_i^{(j)}, r) = y_1^{(j)}$, $\delta(f, r) = f$. All other transitions are left undefined. Let us call the constructed automaton A.

The automaton A is carefully synchronizing if and only if all clauses in C can be satisfied by some assignment of binary values to the variables in X. Moreover, the word $w = rw_1w_2 \ldots w_n0$, is carefully synchronizing if w_i is the value of x_i in such an assignment.

Indeed, note that the first letter of w is necessarily r, as it is the only letter defined for all the states. Moreover, each word starting with r maps Q to a subset of $\{y_i^{(j)}, z_i^{(j)} \mid 1 \leq j \leq m+1\} \cup \{f\}$. The only way for a word to map all states to f is to map them first to the set $\{z_{n+1}^{(j)} \mid 1 \leq j \leq m\} \cup \{f\}$, because there are no transitions defined from any $y_{n+1}^{(j)}$, except the transitions defined by r. If only the states of the set $\{z_{n+1}^{(j)} \mid 1 \leq j \leq m\} \cup \{f\}$ are reached by a word $ra_1 \ldots a_n$, then for every clause in C there exists a variable x_i such that taking $x_i = a_i$ satisfies this clause. Thus there exists an assignment satisfying C.

The constructed automaton is aperiodic, because each cycle which is not a self-loop contains exactly one letter r. □

The complexity of the following problem can be obtained from Proposition 10.

Positive Matrix
Input: A set $M_1, \ldots, M_k$ of $n \times n$ binary matrices;
Output: Yes if there exists a sequence $M_{i_1} \times \ldots \times M_{i_k}$ of multiplications (possibly with repetitions) providing a matrix with all elements equal to 1, No otherwise.

Corollary 1. Positive Matrix *is NP-hard for two upper-triangular and two lower-triangular matrices.*

Proof. The proof uses the idea from [31]. Consider three transition matrices corresponding to the letters of the automaton constructed in the proof of Proposition 10. Add the matrix corresponding to the letter mapping the state f to all states and undefined for all other states. Any sequence of matrices resulting in a matrix with only positive elements must contain the new matrix, and before that there must be a sequence of matrices corresponding to a word carefully synchronizing the automaton from the proof of Proposition 10. Thus we get a reduction from Careful Synchronization for aperiodic partial automata over a three-letter alphabet to Positive Matrix. It is easy to see that the reduction uses two upper-triangular and two lower-triangular matrices. □

Finally, we show the hardness of the following problem (PSPACE-complete in general [32]).

Subset Reachability
Input: An automaton $A = (Q, \Sigma, \delta)$ and a subset S of its states;
Output: Yes if there exists a word w such that $\{\delta(q, w) \mid q \in Q\} = S$, No otherwise.

Theorem 5. Subset Reachability *is NP-complete for weakly acyclic automata.*

Proof. Consider a topological sort of Q. Let w be a shortest word mapping Q to some reachable set of states. Then each letter of w maps at least one state to a state with a larger index in the topological sort. Thus w has length $O(|Q|^2)$, since the maximum total number of such mappings is $(|Q| - 1) + (|Q| - 2) + \ldots + 1 + 0$. Thus, the considered problem is in NP.

For the NP-hardness proof, we again reduce the SAT problem. Given an instance of SAT, construct A_{base} first. Next, add a transition $\delta(y^{(j)}_{n+1}, a) = f$ for $1 \leq j \leq m$, $a \in \{0, 1\}$, resulting in a deterministic automaton A.

Similar to the proof of Proposition 10, C is satisfiable if and only if the set $S = \{z^{(n+1)}_j \mid 1 \leq j \leq m\} \cup \{f\}$ is reachable in A. Indeed, consider the word $w = a_1 a_2 \ldots a_n$ where a_i is the value of x_i in an assignment satisfying C. Then the image of Q under the mapping defined by w equals S. On the other hand, any word w with $\{\delta(q, w) \mid q \in Q\} = S$ has length exactly n, since the application of each new letter increases by one the minimal index i such that a state $y^{(j)}_i$ or $z^{(j)}_i$ is reached. Let $w = a_1 a_2 \ldots a_n$. If the assignment $x_i = a_i$ does not satisfy a clause c_j, then $\{\delta(q, w) \mid q \in Q\}$ contains $y^{(j)}_{n+1}$, and thus cannot be equal to S. □

6. Conclusions and Open Problems

As shown in this paper, weakly acyclic automata serve as an example of a small class of automata where most of the synchronization problems are still hard. More precisely, switching from general automata to weakly acyclic usually results in changing a PSPACE-complete problem to a NP-complete one.

Some problems for weakly acyclic automata are still open. One of them is to study the approximability of the Shortest Sync Word problem: there is a drastic gap between known inapproximability results and the $O(n)$-approximation algorithm for general automata. Another natural problem is to study the Max Sync Set and Set Rank problems complexity in strongly connected automata. The technique used by Vorel for proving PSPACE-completeness of the Sync Set problem in strongly connected automata [11] seems to fail here.

Acknowledgments

I would like to thank Peter J. Cameron for introducing me to the notion of synchronizing automata, and Vojtěch Vorel, Yury Kartynnik, Vladimir Gusev and Ilia Fridman for very useful discussions. I also thank Mikhail V. Volkov and anonymous reviewers for their great contribution to the improvement of the paper.

References

[1] M. V. Volkov, Synchronizing automata and the Černý conjecture, in: C. Martín-Vide, F. Otto, H. Fernau (Eds.), LATA 2008. LNCS, vol. 5196, Springer, Heidelberg, 2008, pp. 11–27. doi:10.1007/978-3-540-88282-4_4.

[2] T. H. Cormen, C. E. Leiserson, R. L. Rivest, C. Stein, Introduction to Algorithms, 3rd Edition, MIT Press, 2009.

[3] G. Jirásková, T. Masopust, On the state and computational complexity of the reverse of acyclic minimal DFAs, in: N. Moreira, R. Reis (Eds.), CIAA 2012. LNCS, vol. 7381, Springer, Heidelberg, 2012, pp. 229–239. doi:10.1007/978-3-642-31606-7_20.

[4] J. Brzozowski, F. E. Fich, Languages of R-trivial monoids, J. Comput. Syst. Sci 20 (1) (1980) 32–49. doi:10.1016/0022-0000(80)90003-3.

[5] D. Eppstein, Reset sequences for monotonic automata, SIAM J. Comput 19 (3) (1990) 500–510. doi:10.1137/0219033.

[6] M. V. Berlinkov, On two algorithmic problems about synchronizing automata, in: A. M. Shur, M. V. Volkov (Eds.), DLT 2014. LNCS, vol. 8633, Springer, Cham, 2014, pp. 61–67. doi:10.1007/978-3-319-09698-8_6.

[7] A. Ryzhikov, M. Szykula, Finding Short Synchronizing Words for Prefix Codes, in: I. Potapov, P. Spirakis, J. Worrell (Eds.), 43rd International Symposium on Mathematical Foundations of Computer Science (MFCS 2018), Vol. 117 of Leibniz International Proceedings in Informatics (LIPIcs), Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany, 2018, pp. 21:1–21:14. doi:10.4230/LIPIcs.MFCS.2018.21.

[8] A. N. Trahtman, The Cerný conjecture for aperiodic automata, Discrete Math. Theor. Comput. Sci. 9 (2).

[9] I. K. Rystsov, Polynomial complete problems in automata theory, Inform. Process. Lett. 16 (3) (1983) 147–151. doi:10.1016/0020-0190(83)90067-4.

[10] S. Sandberg, Homing and synchronizing sequences, in: M. Broy, B. Jonsson, J.-P. Katoen, M. Leucker, A. Pretschner (Eds.), Model-Based Testing of Reactive Systems: Advanced Lectures. LNCS, vol. 3472, Springer, Heidelberg, 2005, pp. 5–33. doi:10.1007/11498490_2.

[11] V. Vorel, Subset synchronization and careful synchronization of binary finite automata, Int. J. Found. Comput. Sci. 27 (5) (2016) 557–578. doi:10.1142/S0129054116500167.

[12] B. K. Natarajan, An algorithmic approach to the automated design of parts orienters, in: Proceedings of the 27th Annual Symposium on Foundations of Computer Science, 1986, pp. 132–142.

[13] A. Ryzhikov, A. Shemyakov, Subset synchronization in monotonic automata, Fundam. Inform. 162 (2-3) (2018) 205–221. doi:10.3233/FI-2018-1721.

[14] J.-É. Pin, On two combinatorial problems arising from automata theory, Ann. Discrete Math. 17 (1983) 535–548.

[15] M. Szykuła, Improving the Upper Bound on the Length of the Shortest Reset Word, in: STACS 2018, LIPIcs, Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2018, pp. 56:1–56:13.

[16] A. Cardoso, The Černý Conjecture and Other Synchronization Problems, Ph.D. thesis, University of Porto, Portugal (2014).

[17] M. Sipser, Introduction to the Theory of Computation, 3rd Edition, Cengage Learning, 2012.

[18] V. V. Vazirani, Approximation Algorithms, Springer, 2001.

[19] A. Ryzhikov, Synchronization problems in automata without non-trivial cycles, in: A. Carayol, C. Nicaud (Eds.), CIAA 2017. LNCS, vol. 10329, Springer, Cham, 2017, pp. 188–200.

[20] I. K. Rystsov, Reset words for commutative and solvable automata, Theor. Comput. Sci. 172 (1) (1997) 273–279. doi:10.1016/S0304-3975(96)00136-3.

[21] D. Ananichev, M. Volkov, Synchronizing monotonic automata, Theor. Comput. Sci. 327 (3) (2004) 225–239. doi:10.1016/j.tcs.2004.03.068.

[22] P. Gawrychowski, D. Straszak, Strong inapproximability of the shortest reset word, in: F. G. Italiano, G. Pighizzini, T. D. Sannella (Eds.), MFCS 2015. LNCS, vol. 9234, Springer, Heidelberg, 2015, pp. 243–255. doi:10.1007/978-3-662-48057-1_19.

[23] U. C. Türker, H. Yenigün, Complexities of some problems related to synchronizing, non-synchronizing and monotonic automata, Internat. J. Found. Comput. Sci. 26 (01) (2015) 99–121. doi:10.1142/s0129054115500057.

[24] J. Bell, B. Stevens, A survey of known results and research areas for N-queens, Discrete Math. 309 (1) (2009) 1 – 31. doi:10.1016/j.disc.2007.12.043.

[25] D. Zuckerman, Linear degree extractors and the inapproximability of max clique and chromatic number, Theory Comput. 3 (6) (2007) 103–128.

[26] J. Friedman, On the road coloring problem, Proc. Amer. Math. Soc 110 (1990) 1133–1135.

[27] J. Kari, Synchronizing finite automata on eulerian digraphs, Theor. Comput. Sci. 295 (1) (2003) 223 – 232. doi:10.1016/S0304-3975(02)00405-X.

[28] I. K. Rystsov, Rank of a finite automaton, Cybern Syst Anal 28 (3) (1992) 323–328. doi:10.1007/BF01125412.

[29] D. Kozen, Lower bounds for natural proof systems, in: Proceedings of the 18th Annual Symposium on Foundations of Computer Science, 1977, pp. 254–266. doi:10.1109/SFCS.1977.16.

[30] P. V. Martyugin, Complexity of problems concerning carefully synchronizing words for PFA and directing words for NFA, in: F. Ablayev, E. W. Mayr (Eds.), CSR 2010. LNCS, vol. 6072, Springer, Heidelberg, 2010, pp. 288–302. doi:10.1007/978-3-642-13182-0_27.

[31] B. Gerencsér, V. Gusev, R. Jungers, Primitive sets of nonnegative matrices and synchronizing automata, SIAM J. Matrix Anal. Appl. 39 (1) (2018) 83–98. doi:10.1137/16M1094099.

[32] E. A. Bondar, M. V. Volkov, Completely reachable automata, in: C. Câmpeanu, F. Manea, J. Shallit (Eds.), DCFS 2016. LNCS, vol. 9777, Springer, Cham, 2016, pp. 1–17. doi:10.1007/978-3-319-41114-9_1.