P(l)aying for Synchronization*

F. M. Fominykh P. V. Martyugin M. V. Volkov

Institute of Mathematics and Computer Science
Ural Federal University, 620000 Ekaterinburg, Russia

Abstract

Two topics are presented: synchronization games and synchronization costs. In a synchronization game on a deterministic finite automaton, there are two players, Alice and Bob, whose moves alternate. Alice wants to synchronize the given automaton, while Bob aims to make her task as hard as possible. We answer a few natural questions related to such games. Speaking about synchronization costs, we consider deterministic automata in which each transition has a certain price. The problem is whether or not a given automaton can be synchronized within a given budget. We determine the complexity of this problem.

1 Introduction and overview

A complete deterministic finite automaton (DFA) $\mathscr{A} = (Q, \Sigma)$ (here and below Q stands for the state set and Σ for the input alphabet) is called *synchronizing* if there exists a word $w \in \Sigma^*$ whose action brings $\mathscr{A}$ to one particular state no matter at which state w is applied: $q{\cdot}w = q'{\cdot}w$ for all $q, q' \in Q$. Any word w with this property is said to be a *reset* word for the automaton.

Synchronizing automata serve as transparent and natural models of error-resistant systems in many applications (coding theory, robotics, testing of reactive systems) and reveal interesting connections with symbolic dynamics, substitution systems and other parts of mathematics. The literature on synchronizing automata and their applications is rapidly growing so

*Supported by the Russian Foundation for Basic Research, grant 10-01-00793, and by the Presidential Program for young researchers, grant MK-266.2012.1.

that even the most recent surveys [14, 17] are becoming obsolete. A majority of research in the area focuses on the so-called Černý conjecture but the theory of synchronizing automata also offers many other interesting questions. In the present paper we introduce two new directions of the theory and obtain some initial results in these directions.

Section 2 concerns with synchronization games on DFAs. In such a game on a DFA $\mathscr{A}$, there are two players, Alice (Synchronizer) and Bob (Desynchronizer), whose moves alternate. Alice who plays first wants to synchronize $\mathscr{A}$, while Bob aims to prevent synchronization or, if synchronization is unavoidable, to delay it as long as possible. Provided that both players play optimally, the outcome of such a game depends only on the underlying automaton so studying synchronization games may be considered as a way to study synchronizing automata. The most natural questions here are the following. Given a DFA $\mathscr{A}$, how to decide who wins in the synchronization game on $\mathscr{A}$? If Alice wins, how many moves may she need in the worst case, in particular, is there a polynomial of n that bounds from above the number of moves in any game on a DFA with n states for which Alice has a winning strategy? How difficult is it to predict whether or not Alice can win after a certain number of moves? It turns out that these questions can be answered by applying more or less standard techniques. This may be a bit disappointing but as a byproduct, we reveal a somewhat unexpected relation between synchronization games and a version of the Černý conjecture.

In Section 3 we consider weighted automata. A *deterministic weighted automaton* (DWA) is a DFA $\mathscr{A} = (Q, \Sigma)$ endowed with a function $\gamma : Q \times \Sigma \to \mathbb{Z}_+$ where $\mathbb{Z}_+$ stands for the set of all positive integers. In other words, each transition of a DWA has a certain price being a positive integer. Then every computation performed by $\mathscr{A}$ also gets a certain cost, namely, the sum of the costs of the transitions involved. If a DWA happens to be synchronizing and $w \in \Sigma^*$ is its reset word, then one can assign to w a cost measured, say, by the maximum among all costs of applying the word w at a state in Q. While in the non-weighted case one is usually interested in minimizing synchronization time, that is, the length of reset words, in the weighted case it is quite natural to minimize synchronization costs. A basic problem here is to determine, whether or not a given DWA can be synchronized within a given budget $B \in \mathbb{Z}_+$, in other words, whether or not $\mathscr{A}$ admits a reset word whose cost does not exceed B. We demonstrate that this problem is PSPACE-complete.

Besides initial questions discussed in this paper, each of the two outlined research directions leads to several intriguing open problems. We present and briefly discuss two such problems in Section 4.

The paper has grown from the extended abstract [7] by the first and the third authors. The second author has solved one of the problems left open in [7] and his solution has been incorporated in the present paper.

2 Playing for synchronization

The idea to consider synchronization as a game has independently arisen in [1] and [3]. In [1] a one-player game has been used to prove a lower bound on the minimum length of reset words for a certain series of 'slowly' synchronizing automata. In [3] a specific synchronization process arising in software testing has been analyzed in terms of a two-player game. The game that we consider here basically follows the model of [1] but is a two-player game as in [3]. A further game-theoretic setting related to synchronization has been recently suggested in [8].

Now we describe the rules of our synchronization game. It is played by two players, Alice and Bob say, on an arbitrary but fixed DFA $\mathscr{A} = (Q, \Sigma)$. In the initial position each state in Q holds a coin but, as the game progresses, some coins may be removed. The game is won by Alice when all but one coins are removed. Bob wins if he can keep at least two coins unremoved indefinitely long.

Alice moves first, then players alternate moves. The player whose turn it is to move proceeds by selecting a letter $a \in \Sigma$. Then, for each state $q \in Q$ that held a coin before the move, the coin advances to the state $q{\cdot}a$. (In the standard graphical representation of $\mathscr{A}$ as the labelled digraph with Q as the vertex set and the labelled edges of the form $q \xrightarrow{a} q{\cdot}a$, one can visualize the move as follows: all coins simultaneously slide along the edges labelled a.) If after this several coins happen to arrive at the same state, all of them but one are removed so that when the move is completed, each state holds at most one coin.

Fig. 1 illustrates the rules. Its upper part shows a typical position in a game on a 5-state automaton with 2 input letters a and b. The left lower part shows the effect of the move b while the right lower part demonstrates the result of the move a. Observe that in the latter case the dark-gray coin has been removed because it and the light-gray coin had arrived at the same state.

Let $a_1, a_2, \ldots, a_k$ with $a_i \in \Sigma$ be a sequence of moves in the synchronization game on $\mathscr{A} = (Q, \Sigma)$ and let $w = a_1 a_2 \cdots a_k$. It is easy to see that the set of states holding coins after this sequence of moves coincides with the image of Q under the action of the word w. Thus, sequences of

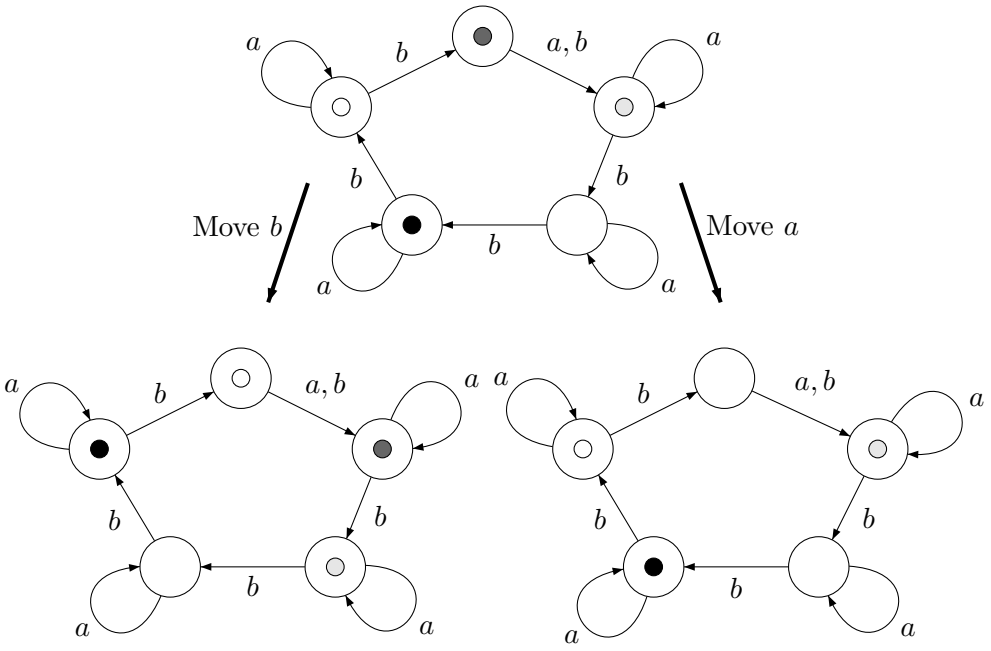

Figure 1: Moves in a synchronization game

moves that lead to Alice's win correspond precisely to reset words for $\mathscr{A}$. Therefore Bob wins on each DFA which is not synchronizing. Can he win on a synchronizing automaton? Yes, he can: for instance, we show that Bob wins on automata in the famous Černý series.

Černý [4] found for each $n > 1$ a synchronizing automaton $\mathscr{C}_n$ with n states and 2 input letters whose shortest reset word has length $(n-1)^2$. The states of $\mathscr{C}_n$ are the residues modulo n and the input letters a and b act as follows:

$$m{\cdot}a = \begin{cases} 1 & \text{for } m = 0, \\ m & \text{for } 1 \le m < n; \end{cases} \qquad m{\cdot}b = m + 1 \pmod{n}.$$

The automaton is shown in Fig. 2.

Example 1. For each $n > 3$, Bob has a winning strategy in the synchronization game on $\mathscr{C}_n$.

Proof. Observe that in $\mathscr{C}_n$, the only state where two coins can meet is the state 1; moreover, this can happen only provided the move a has been played and before the move the states 0 and 1 both held a coin. We may assume for certainty that in this situation it is the coin arriving from 0 that is removed after the move.

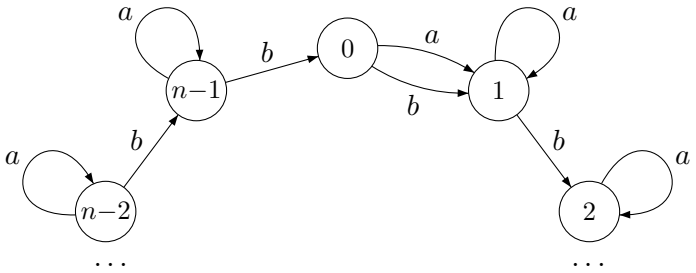

Figure 2: The Černý automaton $\mathscr{C}_n$

Under this convention, the winning strategy for Bob is as follows. Bob only has to trace the coins that cover the states $n-1$ and 1 in the initial position. For his moves he must always select the letter a except two cases: when the chosen coins cover either $n-2$ and 0 or 0 and 2 in which cases Bob must select b. This way Bob can always keep the coins two steps apart from each other thus preventing them of being removed. □

On the other hand, it is easy to find DFAs on which Alice has a winning strategy. For instance, a DFA $\mathscr{A}$ is called *definite* in [11] if there exists an $n > 0$ such that every input word of length at least n is a reset word for $\mathscr{A}$. Clearly, on each definite automaton, Alice always wins by selecting her moves at random.

The rules of our game readily guarantee that, given a DFA $\mathscr{A} = (Q, \Sigma)$, one of the players must have a winning strategy in the synchronization game on $\mathscr{A}$. If Alice has a winning strategy, consider a shortest winning sequence of her moves. Then it is clear that each move in this sequence creates a position that could not have appeared after an earlier move. However, the number of possible positions of the game does not exceed $2^{|Q|} - 1$ since each position is specified by the subset of states that currently hold coins. Therefore, if Alice has a winning strategy, she should be able to win after less than $2^{|Q|}$ moves. Thus, one can decide which player has a winning strategy in the game on $\mathscr{A}$ by an exhaustive search through all $|\Sigma|^{2^{|Q|+1}}$ words of length $2^{|Q|+1}$ over Σ in which letters in the odd positions are Alice's moves and ones in the even positions are Bob's replies. Of course, this brute force procedure is extremely inefficient and it is natural to ask whether an efficient—say, polynomial in the number of states—algorithm exists. A positive answer can be deduced from the next observation.

Lemma 1. *Alice has a winning strategy in the synchronization game on a DFA if and only if she has a winning strategy in every position in which only two states of the DFA hold coins.*

Proof. If Alice has no winning strategy for a position P with two coins, C and C' say, then Bob has a winning strategy for P. If Bob plays in the initial position according to this strategy, that is, selects his moves only on the basis of the location of the coins C and C', as if there were no other coins, the two coins persist forever so that Alice loses the game. (Here we assume that whenever one of the coins C and C' meets some third coin on some state in the course of the game, then it is this third coin that gets removed.)

Conversely, if Alice can win in every position in which only two states hold coins, she can use the following strategy. In the initial position she chooses a pair of coins, C and C' say, and plays as if there were no other coins, that is, she applies her winning strategy for the position in which C and C' cover the same states as they do in the initial position and all other coins are removed. This brings the game to a position in which either C or C' is removed. Then Alice chooses another pair of coins and again plays as if these were the only coins, and so on. Since at least one coin is removed in each round, Alice eventually wins. □

Observe that Lemma 1 implies a cubic (in the number n of states of the underlying DFA) upper bound on the number of moves in any game that Alice wins. Indeed, suppose she uses the strategy just described and works with a pair of coins C and C'. Let q_i and q'_i be the states holding the coins C and C' after the i^{th} move of Alice. Then if Alice plays optimally, we must have $\{q_i, q'_i\} \neq \{q_{i+j}, q'_{i+j}\}$ whenever $j > 0$. Indeed, the equality $\{q_i, q'_i\} = \{q_{i+j}, q'_{i+j}\}$ means that wherever Alice moves C and C' by her $(i+1)^{th}$, $\ldots$, $(i+j-1)^{th}$ moves, Bob can force Alice to return the coins by her $(i+j)^{th}$ move to the same states that the coins occupied after her i^{th} move. Then Bob can force Alice to return C and C' to the same states also by her $(i+2j)^{th}$, $(i+3j)^{th}$, $\ldots$ moves, whence none of the two coins can ever be removed, a contradiction.

Hence the number of Alice's moves in any round in which she works with any fixed pair of coins does not exceed $\binom{n}{2}$. Moreover, in every synchronizing automaton there exist states q and q' such that $q \cdot a = q' \cdot a$ for some letter a. Therefore Alice can remove one coin by her first move. After that she needs at most $n-2$ rounds to remove $n-2$ of the remaining $n-1$ coins. We thus obtain:

Corollary 2. *If Alice has a winning strategy in the synchronization game on a DFA with n states, she can win in at most $\binom{n}{2}(n-2)+1$ moves.*

Now we return to the decidability question.

Theorem 3. *Let $\mathscr{A} = (Q, \Sigma)$ be a DFA with $|Q| = n$ and $|\Sigma| = k$. There exists an algorithm that in $O(n^2k)$ time decides who has a winning strategy in the synchronization game on $\mathscr{A}$.*

Proof. We describe the algorithm rather informally. First we construct a new DFA $\mathscr{P} = (P \times \{0,1\} \cup \{s\}, \Sigma)$ where P is the set of all positions with two coins (each such position is specified by a couple of states holding coins) and s is an extra state. The action of the letters is defined as follows: all letters fix s and if $p \in P$ is the position in which two states $q, q' \in Q$ hold coins, $x \in \{0,1\}$, and $a \in \Sigma$, then

$$(p,x)\cdot a = \begin{cases} (p', 1-x) & \text{if } q\cdot a \neq q'\cdot a, \\ s & \text{otherwise,} \end{cases}$$

where p' is the position in which $q\cdot a$ and $q'\cdot a$ hold coins. Thus, the automaton $\mathscr{P}$ encodes 'transcripts' of all games starting in positions in P; the extra bit x controls whose turn it is to move: Alice moves if $x = 0$ and Bob moves if $x = 1$. Clearly, $\mathscr{P}$ has $n^2 - n + 1$ states and $k(n^2 - n + 1)$ edges (transitions).

We mark the state s and then recursively propagate the marking to $P \times \{0,1\}$: a state of the form $(p,0)$ is marked if and only if there is an $a \in \Sigma$ such that $(p,0)\cdot a$ is marked and a state of the form $(p,1)$ is marked if and only if for all $a \in \Sigma$ the states $(p,1)\cdot a$ are marked. Clearly, the marking can be done by a breadth-first search in the underlying digraph of $\mathscr{P}$ with all edges reversed. The well known time estimate for breadth-first search in a graph with v vertices and e edges is $O(v+e)$, see, e.g., Section 22.2 in [5], whence we conclude that the marking can be completed in $O(n^2k)$ time. It follows from the construction of $\mathscr{P}$ and from the marking rules that Alice can win in the game starting at a position $p \in P$ if and only if the state $(p,0)$ is marked. This and Lemma 1 readily imply that Alice has a winning strategy in the game on $\mathscr{A}$ if and only if all states of the form $(p,0)$ get marked (or, equivalently, all states of $\mathscr{P}$ get marked). □

In contrast, we show that it is rather hard to decide whether or not Alice can win after a certain number of moves. Namely, consider the following decision problem:

Short SynchroGame: *given a DFA $\mathscr{A}$ and a positive integer ℓ, is it true that Alice can win the synchronization game on $\mathscr{A}$ in at most ℓ moves?*

Theorem 4. Short SynchroGame *is PSPACE-complete.*

Proof. We first verify that Short SynchroGame lies in the class PSPACE. Take an arbitrary instance $(\mathscr{A} = (Q, \Sigma), \ell)$ of the problem with $|Q| = n$, $|\Sigma| = k$. If $\ell \geq \binom{n}{2}(n-2)+1$, then by Corollary 2 Alice can win on $\mathscr{A}$ in at most ℓ moves whenever she can win on $\mathscr{A}$. Thus, for instances of Short SynchroGame with $\ell \geq \binom{n}{2}(n-2)+1$, we can solve the problem even in polynomial time invoking the algorithm from the proof of Theorem 3. Therefore we can restrict the problem to instances with $\ell < \binom{n}{2}(n-2)+1$. (This explains in particular that it does not really matter whether ℓ is given in binary or in unary.)

For each P being a non-empty subset of Q and each non-negative integer $m \leq \ell$, introduce two Boolean variables $A(P,m)$ and $B(P,m)$. The value of $A(P,m)$ is 1 if and only if Alice wins in at most m moves starting from the position in which only states in P hold coins. The value of $B(P,m)$ is 1 if and only if Alice wins in at most m moves in every position that can arise after Bob's move in the position in which only states in P hold coins. Then the answer to the instance $(\mathscr{A} = (Q, \Sigma), \ell)$ of Short SynchroGame is 'YES' if and only if $A(Q, \ell) = 1$ and there is a straightforward recursion that allows one to calculate the values of the variables $A(P,m)$ and $B(P,m)$:

$$\begin{aligned} A(P,m) &= \bigvee_{a\in\Sigma} B(P{\cdot}a, m-1) && \text{for } m > 0, \\ B(P,m) &= \mathop{\&}_{a\in\Sigma} A(P{\cdot}a, m) && \text{for all } m, \\ A(P,0) &= 1 && \text{if and only if } P \text{ is a singleton.} \end{aligned} \tag{1}$$

Here $P{\cdot}a$ stands for the set $\{q{\cdot}a \mid q \in P\}$.

The total number of the variables $A(P,m)$ and $B(P,m)$ is of magnitude $2^{n+1}\ell$ so exponential in the size of the input. Nevertheless it is easy to unfold the recursion (1) in polynomial space by a sort of depth-first search. As above, we prefer to describe our algorithm informally. At its generic step, the algorithm tries to evaluate some $A(P,m)$ or $B(P,m)$. Consider the case of $A(P,m)$. If $m = 0$, the algorithm simply checks whether P is a singleton (in other words, if Alice has won) and sets $A(P,0) = 1$ if this is the case and $A(P,0) = 0$ otherwise. If $m > 0$, the algorithm verifies if all variables of the form $B(P{\cdot}a, m-1)$ where $a \in \Sigma$ have already been evaluated and if this is the case, evaluates $A(P,m)$ according to (1). As soon as the value of $A(P,m)$ has been found, the algorithm stores the value but forgets the set P and the used values of $B(P{\cdot}a, m-1)$ so that the space previously occupied by these data can be re-used. In the case when some of the variables $B(P{\cdot}a, m-1)$ have not yet been evaluated, the algorithm postpones evaluation of $A(P,m)$

and tries to evaluate the yet unknown variable $B(P{\cdot}a, m-1)$ with the least a (according to a fixed ordering of the alphabet Σ). In the same manner the algorithm works when evaluating $B(P, m)$.

Clearly, each set P that appears in the course of the implementation must be of the form $P = Q{\cdot}a_1 \cdots a_s$, where $a_1, \ldots, a_s \in \Sigma$ and $2s \leq \ell$. When calculating the value of $A(P, \ell - \frac{s}{2})$ (if s is even) or $B(P, \ell - \frac{s+1}{2})$ (if s odd), the algorithm only needs to maintain the following data:

- the sets Q, $Q{\cdot}a_1$, $Q{\cdot}a_1a_2$, $\ldots$, $Q{\cdot}a_1 \cdots a_{s-1}$ that appear on the way from Q to P and the set P itself;
- the already known but not yet used values of variables of the form $B(Q{\cdot}b_1, \ell-1)$, $A(Q{\cdot}a_1b_2, \ell-1)$, $\ldots$, $B(Q{\cdot}a_1 \cdots a_{t-1}b_t, \ell - \frac{t+1}{2})$ for odd $t \leq s$, $A(Q{\cdot}a_1 \cdots a_{t-1}b_t, \ell - \frac{t}{2})$ for even $t \leq s$;
- pointers to the yet unknown variables of the form $B(Q{\cdot}b_1, \ell - 1)$, $A(Q{\cdot}a_1b_2, \ell - 1)$, $\ldots$, $B(Q{\cdot}a_1 \cdots a_{t-1}b_t, \ell - \frac{t+1}{2})$ for odd $t \leq s$, $A(Q{\cdot}a_1 \cdots a_{t-1}b_t, \ell - \frac{t}{2})$ for even $t \leq s$ with the least $b_1, b_2, \ldots, b_t$ respectively.

At each step one has to store at most 2ℓ sets of size at most n, at most $(k-1)(2\ell-1)$ bits for the already calculated but not yet used values, and at most $2\ell - 1$ pointers to the 'next' variables to be evaluated. Thus, a polynomial space suffices.

In order to show that Short SynchroGame is PSPACE-complete, we use a reduction from the well known PSPACE-complete problem QSAT (Quantified Satisfiability) in its game-theoretic form, see Section 19.1 in [10]. An instance of QSAT is a Boolean formula in conjunctive normal form with variables $x_1, \ldots, x_n$. Alice and Bob play on such an instance alternatingly: first Alice assigns a value 0 or 1 to x_1, then Bob assigns a value to x_2 and so on. Alice wins the game if after all the variables get some values, the formula becomes true; Bob wins if the formula becomes false.

For the reduction we use Eppstein's construction [6]. We reproduce it here for the reader's convenience. Given an arbitrary instance ψ of QSAT with n variables $x_1, \ldots, x_n$ and m clauses $c_1, \ldots, c_m$, we construct a DFA $\mathscr{A}(\psi)$ with 2 input letters a and b as follows. The state set Q of $\mathscr{A}(\psi)$ consists of $(n+1)m$ states $q_{i,j}$, $1 \leq i \leq m$, $1 \leq j \leq n+1$, and a special state

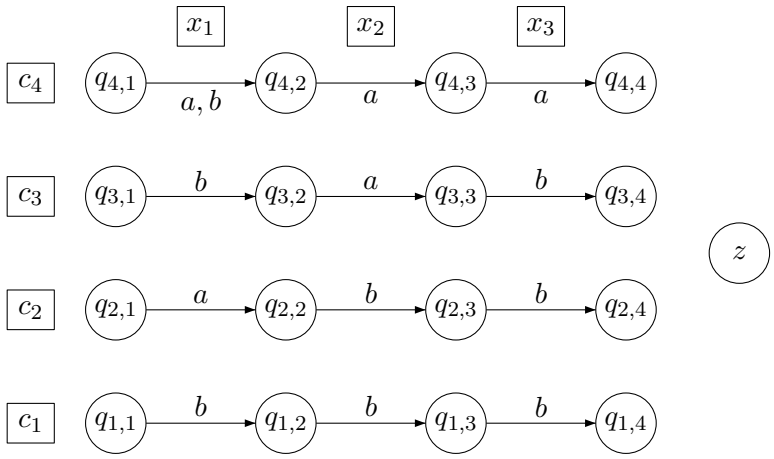

Figure 3: The automaton $\mathscr{A}(\psi_0)$

z. The transitions are defined by

$$q_{i,j}{\cdot}a = \begin{cases} z \text{ if the literal } x_j \text{ occurs in } c_i, \\ q_{i,j+1} \text{ otherwise} \end{cases} \qquad \text{for } 1 \le i \le m,\ 1 \le j \le n;$$

$$q_{i,j}{\cdot}b = \begin{cases} z \text{ if the literal } \neg x_j \text{ occurs in } c_i, \\ q_{i,j+1} \text{ otherwise} \end{cases} \qquad \text{for } 1 \le i \le m,\ 1 \le j \le n;$$

$$q_{i,n+1}{\cdot}a = q_{i,n+1}{\cdot}b = z{\cdot}a = z \cdot b = z \qquad \text{for } 1 \le i \le m.$$

Fig. 3 shows an automaton of the form $\mathscr{A}(\psi)$ build for the QSAT instance

$$\psi_0 = \{x_1 \vee x_2 \vee x_3,\ \neg x_1 \vee x_2 \vee x_3,\ x_1 \vee \neg x_2 \vee x_3,\ \neg x_2 \vee \neg x_3\}.$$

If at some state $q \in Q$ in Fig. 3 there is no outgoing edge labelled $c \in \{a, b\}$, the edge $q \xrightarrow{c} z$ is assumed (those edges are omitted to improve readability).

Observe that Alice wins on ψ_0: she may start with letting $x_1 = 1$ thus ensuring that the first and the third clause become true. Now if Bob responds by letting $x_2 = 0$, then the fourth clause becomes true and Alice wins by letting $x_3 = 1$ which makes also the second clause be true. If Bob responds by letting $x_2 = 1$, then the second clause becomes true and Alice wins by letting $x_3 = 0$.

This winning strategy precisely corresponds to the following winning strategy for Alice in the synchronization game on $\mathscr{A}(\psi_0)$: Alice starts with

the move a and if Bob responds with b (respectively a), she wins by using a (respectively b).

In general, if Alice has a winning strategy for an instance ψ of QSAT with n variables $x_1, \ldots, x_n$ and m clauses $c_1, \ldots, c_m$, she can imitate this strategy in the synchronization game on $\mathscr{A}(\psi)$ using the move a whenever she lets some variable to be true and using the move a whenever she lets some variable to be false. Thus Alice wins the synchronization game in at most n moves. Conversely, if Alice has a strategy that allows her to win the synchronization game on $\mathscr{A}(\psi)$ in at most n moves, she can imitate this strategy in the game on ψ assigning to the current variable values 1 or 0 according to whether her current move in the game on $\mathscr{A}(\psi)$ should be a or b. To justify this claim, it suffices to observe that a truth assignment $\tau : \{x_1, \ldots, x_n\} \to \{0, 1\}$ enforces $c_i(\tau(x_1), \ldots, \tau(x_n)) = 1$ if and only if the word $a_1 a_2 \cdots a_n$ of length n defined by

$$a_j = \begin{cases} a & \text{if } \tau(x_j) = 1, \\ b & \text{if } \tau(x_j) = 0 \end{cases}$$

sends all states $q_{i,1}, \ldots, q_{i,n+1}$ of the automaton $\mathscr{A}(\psi)$ to the state z.

Thus, the answer to an instance ψ of QSAT is 'YES' if and only if so is the answer to the instance $(\mathscr{A}(\psi), n)$ of Short SynchroGame where n is the number of variables of ψ. This reduces QSAT to Short SynchroGame. □

Though Corollary 2 and Theorems 3 and 4 are worth being registered (as they answer to the most natural questions related to synchronization games), the reader acquainted with the theory of synchronizing automata immediately realizes that these results closely follow some more or less standard patterns. Now we proceed with a more original contribution.

Suppose that Alice has a winning strategy in a synchronization game on an n-state DFA. Corollary 2 provides an cubic upper bound for the number of moves in the game. What about lower bounds? Our next result provides a transparent construction from which we can extract a quadratic lower bound.

Theorem 5. *Let $\mathscr{A} = (Q, \Sigma)$ be a synchronizing automaton with $|Q| = n$, $|\Sigma| \geq 2$ and let ℓ be the minimum length of reset words for $\mathscr{A}$. There exists a DFA $\mathscr{D}$ with $2n$ states such that Alice wins in the synchronization game on $\mathscr{D}$ but needs at least ℓ moves for this.*

Proof. We fix a letter $b \in \Sigma$ and a state $q_0 \in Q$. Now let $\mathscr{D} = (Q \times \{0,1\}, \Sigma)$ where for each $q \in Q$ the action of an arbitrary letter $a \in \Sigma$ is defined as follows:

$$(q,0)\cdot a = (q\cdot a, 1), \qquad (q,1)\cdot a = \begin{cases} (q,0) & \text{if } a = b, \\ (q_0,1) & \text{otherwise.} \end{cases}$$

We call $\mathscr{D}$ the *duplication* of $\mathscr{A}$. Fig. 4 shows the duplication of the Černý automaton $\mathscr{C}_n$ from Fig. 2 (with the state 0 in the role of q_0).

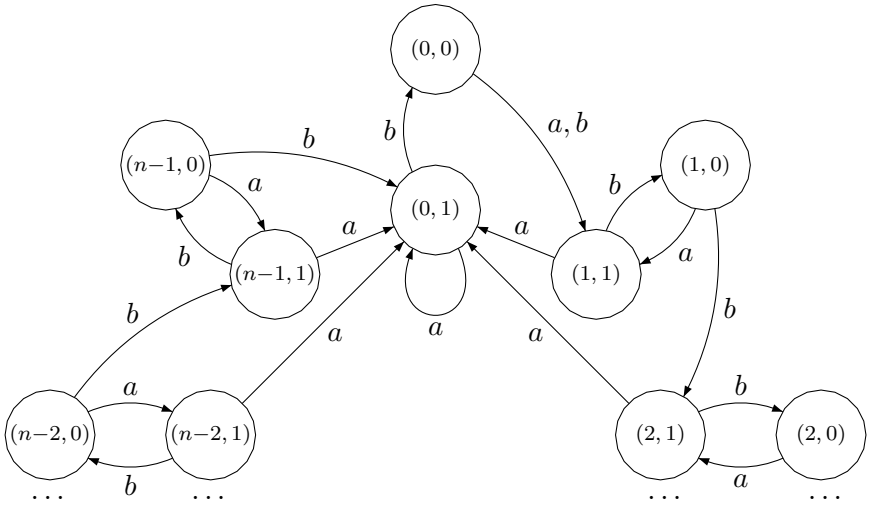

Figure 4: The duplication of the automaton $\mathscr{C}_n$

Suppose that Alice opens the game by selecting a letter $a \neq b$. After that only states of the form $(q,1)$ hold coins. Bob must reply with the move b since he loses immediately otherwise. After that coins cover the states $(q,0)$ with $q \in Q\cdot a \cup \{q_0\}$. Now if Alice spells out a reset word for $\mathscr{A}$, she wins. Indeed, as soon as Bob selects a letter different from b, he loses immediately, and if he replies with b to all Alice's moves, each pair (Alice's move, Bob's move) has the same effect as applying the letter selected by Alice in the DFA $\mathscr{A}$.

On the other hand, Alice needs at least ℓ moves to win if Bob replies with b to each of her moves. Indeed, if Bob plays this way and a winning sequence of Alice's moves forms a word $w \in \Sigma^*$, then after the last move of the sequence every state $(q,1)$ with $q \in Q\cdot w$ still holds a coin. Thus, for Alice to win, w must be a reset word for $\mathscr{A}$, whence the length of w is at least ℓ. □

We denote by $\mathscr{D}_n$ the duplication of the Černý automaton $\mathscr{C}_n$. Combining Theorem 5 and the fact that the minimum length of reset words for $\mathscr{C}_n$ is $(n-1)^2$, we obtain that Alice needs at least $(n-1)^2$ moves to win on $\mathscr{D}_n$. (In fact, the exact number of moves needed is easily seen to be $(n-1)^2+1$.) Thus, we have found a series of k-state DFAs ($k = 2n$ is even) on which Alice's win requires a quadratic in k number of moves. A similar series can be constructed for odd k: we can just add an extra state to $\mathscr{D}_n$ and let both a and b send this added state to the state $(q_0, 1)$.

We notice that the duplication of an arbitrary DFA belongs to a very special class of synchronizing automata as it can be reset by a word of length 2. A somewhat unexpected though immediate consequence of Theorem 5 is that a progress in understanding synchronization games within this specific class may lead to a solution of a major problem in the theory of synchronizing automata.

Corollary 6. *If for every n-state synchronizing automaton with a reset word of length 2 on which Alice can win, she has a winning strategy with $O(n^2)$ moves, then every n-state synchronizing automaton has a reset word of length $O(n^2)$.*

Recall that all known results on synchronization of n-state DFAs (see [12] and [16] for the best bounds) guarantee only the existence of reset words of length $\Omega(n^3)$.

3 Paying for synchronization

Let $\mathscr{A} = (Q, \Sigma, \gamma)$ be a DWA, where $\gamma : Q \times \Sigma \to \mathbb{Z}_+$ is a cost function. For $w = a_1 \cdots a_k \in \Sigma^*$ and $q \in Q$, the cost of applying w at q is

$$\gamma(q, w) = \sum_{i=0}^{k-1} \gamma\big(q \cdot (a_1 \cdots a_i), a_{i+1}\big).$$

If $\mathscr{A}$ is a synchronizing automaton and w is its reset word, then the cost of synchronizing $\mathscr{A}$ by w is defined as $\gamma(w) = \max_{q \in Q} \gamma(q, w)$. The intuition for this choice of $\gamma(w)$ is as follows: we use w to bring $\mathscr{A}$ to a certain state from an unknown state, and therefore, we have to take the most costly case into account. (Of course, in some situations other definitions of the cost of synchronization may make sense. For instance, if we treat synchronization in the flavor of Section 2, that is, as the process of moving coins initially placed on all states in Q to a certain state, it is more natural to define the

cost of the process as $\sum_{q \in Q} \gamma(q, w)$. The results that follow can be adapted to this setting mutatis mutandis.)

Fig. 5 shows a DWA (transition costs are included in the labels) and illustrates the difference between two optimization problems: minimizing synchronization cost and minimizing the length of reset words. The shortest reset word for the DWA is b^3 but the cost of synchronizing by b^3 is 48. On the other hand, the longer word a^2baba^2 manages to avoid the 'expensive' loop at the state 3 whence the cost of synchronizing by a^2baba^2 is only 7.

We study in the computational complexity of the following decision problem:

Synchronizing on Budget: *Given a DWA $\mathscr{A} = (Q, \Sigma, \gamma)$ and a positive integer B, is it true that $\mathscr{A}$ has a reset word w with $\gamma(w) \leq B$?*

Here we assume that the values of γ and the number B are given in binary. (The unary version of Synchronizing on Budget can be easily shown to be NP-complete on the basis of the NP-completeness of the problem Short Reset Word [13, 6]: given a DFA $\mathscr{A}$ and a positive integer ℓ, is it true that $\mathscr{A}$ has a reset word of length ℓ?)

Theorem 7. Synchronizing on Budget *is PSPACE-complete.*

Proof. By Savitch's theorem (see Section 4.3 in [10]), in order to show that Synchronizing on Budget lies in the class PSPACE, it suffices to solve this problem in polynomial space by a non-deterministic algorithm. A small difficulty is that for some instances $(Q, \Sigma, \gamma; B)$ of Synchronizing on Budget, every reset word w satisfying $\gamma(w) \leq B$ may be exponentially long in $|Q|$ and so even if our algorithm correctly guesses such a w, it would not have enough space to store its guess. To bypass the difficulty, the algorithm should guess w letter by letter. It guesses the first letter of w (say, a), applies a at every state $q \in Q$ and saves two arrays: $\{q \cdot a\}$ and $\{\gamma(q, a)\}$. Each

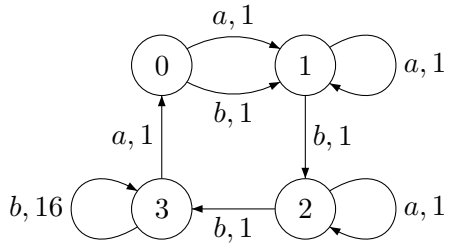

Figure 5: A deterministic weighted automaton

of the arrays clearly requires only polynomial space. Then the algorithm guesses the second letter of w and updates both arrays, etc. At the end of the guessing steps the algorithm check whether all entries of the first array are equal (if so, then w is indeed a reset word for (Q,Σ)) and whether the maximum number in the second array is less than or equal to B (if so, then synchronization is indeed achieved within the budget B).

To show that Synchronizing on Budget is PSPACE-complete, we use a reduction from a problem concerning partial automata. A *partial* finite automaton (PFA) is a pair $\mathscr{A} = (Q,\Sigma)$, where Q is the state set and Σ is the input alphabet whose letters act on Q as partial transformations. Such a PFA is said to be *carefully synchronizing* if there exists $w = a_1 \cdots a_\ell$ with $a_1,\dots,a_\ell \in \Sigma$ such that $q{\cdot}a_i$ with $1 \le i \le \ell$ is defined for all $q \in Q{\cdot}(a_1 \cdots a_{i-1})$ and $|Q{\cdot}w| = 1$. Every word w with these properties is called a *careful reset word* for $\mathscr{P}$. Informally, a careful reset word synchronizes $\mathscr{A}$ and manages to avoid any undefined transition.

The second author [9] has recently proved that the following problem is PSPACE-complete:

Careful Synchronization: *Is a given PFA carefully synchronizing?*

It is the problem that we reduce to Synchronizing on Budget. Our reduction relies on a known fact whose proof is included for the reader's convenience.

Lemma 8. *The minimum length of careful reset words for carefully synchronizing PFAs with n states does not exceed $2^n - n - 1$.*

Proof. Given a PFA $\mathscr{A} = (Q,\Sigma)$ with $|Q| = n$, consider the set of the non-empty subsets of Q and let each $a \in \Sigma$ act on $P \subseteq Q$ as follows:

$$P{\cdot}a = \begin{cases} \{q{\cdot}a \mid q \in P\} & \text{provided } q{\cdot}a \text{ is defined for all } q \in P, \\ \text{undefined} & \text{otherwise.} \end{cases}$$

We obtain a new PFA $\mathscr{P}$, and it is clear that $w \in \Sigma^*$ is a careful reset word for $\mathscr{A}$ if and only if w labels a path in $\mathscr{P}$ starting at Q and ending at a singleton. A path of minimum length does not visit any state of $\mathscr{P}$ twice and stops as soon as it reaches a singleton. Hence the length of the path does not exceed the number of non-empty and non-singleton subsets of Q, that is, $2^n - n - 1$. □

Now take an arbitrary instance of Careful Synchronization, that is, a PFA $\mathscr{A} = (Q,\Sigma)$. We assign to $\mathscr{A}$ an instance of Synchronizing on

BUDGET as follows. First, extend the action of each letter $a \in \Sigma$ to the whole set Q letting

$$q \odot a = \begin{cases} q{\cdot}a & \text{if } q{\cdot}a \text{ is defined in } \mathscr{A}, \\ q & \text{otherwise.} \end{cases}$$

These extended actions give rise to a DFA $\mathscr{A}'$ with the same state set Q and input alphabet Σ. Further, let $|Q| = n$, and define $\gamma : Q \times \Sigma \to \mathbb{Z}_+$ by the rule:

$$\gamma(q, a) = \begin{cases} 1 & \text{if } q{\cdot}a \text{ is defined in } \mathscr{A}, \\ 2^n & \text{otherwise.} \end{cases}$$

This makes $\mathscr{A}'$ a DWA. The construction is illustrated by Fig. 6. Finally, let $B = 2^n - 1$. Observe that the binary presentations of B and of the values of γ are of a linear in n size so that the construction requires only polynomial time in the size of the PFA $\mathscr{A}$.

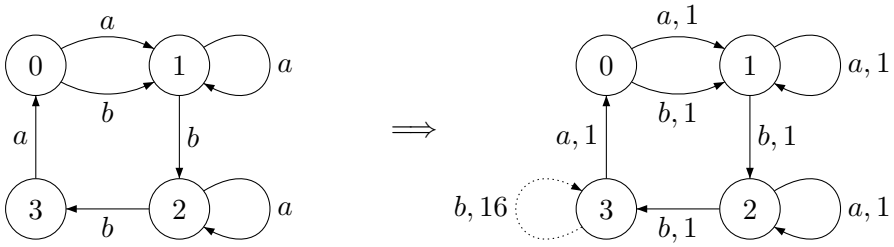

Figure 6: Transforming a partial automaton into a weighted automaton

We aim to show that the PFA $\mathscr{A}$ is carefully synchronizing if and only if the DWA $\mathscr{A}'$ can be synchronized within the budget B. Indeed, if w is a careful reset word for $\mathscr{A}$, then w can be applied to every state in $\mathscr{A}$. This implies that w labels the same paths in $\mathscr{A}'$ as it does in $\mathscr{A}$ whence w synchronizes $\mathscr{A}'$ and involves only transitions with cost 1. Therefore $\gamma(q, w)$ is equal to the length of w for each $q \in Q$ and so is $\gamma(w) = \max_{q \in Q} \gamma(q, w)$. By Lemma 8 w can be chosen to be of length at most $2^n - n - 1$, whence $\gamma(w) \leq 2^n - n - 1 < 2^n - 1 = B$. Conversely, if w is a reset word for $\mathscr{A}'$ with $\gamma(w) \leq B$, then $\gamma(q, w) \leq 2^n - 1$ for each $q \in Q$, whence no path labelled w and starting at q involves any transition with cost 2^n. This means every transition in such a path is induced by a transition with the same effect defined in $\mathscr{A}$. Therefore w can be applied to every state in $\mathscr{A}$. Since all paths labelled w are coterminal in $\mathscr{A}'$, they have the same property in $\mathscr{A}$ and w is a careful reset word for $\mathscr{A}$. □

4 Open problems

Due to the space constraint we restrict ourselves to just two interesting problems.

Road Coloring games. A digraph G in which each vertex has the same out-degree k is called a *digraph of out-degree k*. If we take an alphabet Σ of size k, then we can label the edges of such G by letters of Σ such that the resulting automaton will be complete and deterministic. Any DFA obtained this way is referred to as a *coloring* of G.

The famous Road Coloring Problem asked for necessary and sufficient conditions on a digraph G to admit a synchronizing coloring. The problem has been recently solved by Trahtman [15] and the solution implies that if G has a synchronizing coloring, then such a coloring can be found in $O(n^2k)$ time where n is the number of vertices and k is the out-degree of G, see [2].

Now consider the following *Rood Coloring game*. Alice and Bob alternately label the edges of a given digraph G of out-degree k by letters from an alphabet Σ of size k (observing the rule that no edges leaving the same vertex may get the same label) until G becomes a DFA. Alice who plays first wins if the resulting DFA is synchronizing, and Bob wins otherwise.

Problem 1. *Is there an algorithm that, given a digraph G of constant out-degree, decides in polynomial in the size of G time which player has a winning strategy in the Road Coloring game on G?*

Observe that there are digraphs on which Alice wins by making random moves (for instance, the underlying digraphs of the automata in the Černý series can be shown to have this property); on the other hand, Bob can win on some digraphs admitting synchronizing colorings, see Fig. 7 for a simple example.

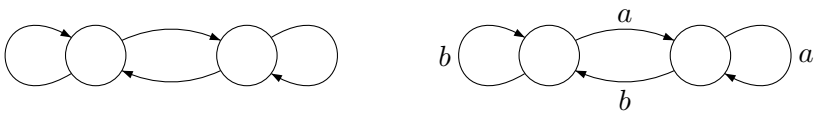

Figure 7: A digraph on which Bob wins the Road Coloring game and its synchronizing coloring

Synchronization games on weighted automata. As a synthesis of the two topics of this paper, one can consider synchronization games on DWAs where the aim of Alice is to minimize synchronization costs while

Bob aims to prevent synchronization or at least to maximize synchronization costs. In particular, we suggest to investigate the following problem that can be viewed as a common generalization of Short SynchroGame and Synchronizing on Budget.

SynchroGame on Budget: *Given a DWA $\mathscr{A} = (Q, \Sigma, \gamma)$ and a positive integer B, is it true that Alice can win the synchronization game on $\mathscr{A}$ with a sequence w of moves satisfying $\gamma(w) \leq B$?*

Problem 2. *Find the computational complexity of* SynchroGame on Budget.

Clearly, the results of the present paper imply that SynchroGame on Budget is PSPACE-hard.

References

[1] Ananichev, D.S., Volkov, M.V., Zaks, Yu.I.: Synchronizing automata with a letter of deficiency 2. Theor. Comput. Sci. 376, 30–41 (2007)

[2] Béal, M.-P., Perrin, D.: A quadratic algorithm for road coloring. Technical report, Université Paris-Est, 2008. Available under `http://arxiv.org/abs/0803.0726`

[3] Blass, A., Gurevich, Yu., Nachmanson, L., Veanes, M.: Play to test. In: Grieskamp, W., Weise, C. (eds.), Formal Approaches to Software Testing, Lect. Notes Comput. Sci., vol. 3997, pp. 32–46. Springer (2006)

[4] Černý, J.: Poznámka k homogénnym eksperimentom s konečnými automatami. Matematicko-fyzikalny Časopis Slovensk. Akad. Vied 14(3), 208–216 (1964) (in Slovak)

[5] Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C.: Introduction to Algorithms. 3rd ed. MIT Press, Cambridge and McGraw-Hill (2009)

[6] Eppstein, D.: Reset sequences for monotonic automata. SIAM J. Comput. 19, 500–510 (1990)

[7] Fominykh, F.M., Volkov, M.V.: P(l)aying for synchronization. In: Moreira, N., Reis, R. (eds.), Implementation and Application of Automata, Lect. Notes Comput. Sci., vol. 7381, pp. 159–170. Springer (2012)

[8] Jungers, R.M.: The synchronizing probability function of an automaton. SIAM J. Discrete Math. 26, 177–192 (2012)

[9] Martyugin, P.V.: Complexity of problems concerning carefully synchronizing words for PFA and directing words for NFA. In: Ablayev, F., Mayr, E.W. (eds.), Computer Science – Theory and Applications, Lect. Notes Comput. Sci., vol. 6072, pp. 288–302. Springer (2010)

[10] Papadimitriou, C.H.: Computational Complexity. Addison-Wesley (1994)

[11] Perles, M., Rabin, M.O., Shamir, E.: The theory of definite automata. IEEE Trans. Electronic Comput. 12, 233–243 (1963)

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

[13] Rystsov, I.K.: On minimizing length of synchronizing words for finite automata. In: Theory of Designing of Computing Systems, pp. 75–82. Institute of Cybernetics of Ukrainian Acad. Sci. (1980) (in Russian)

[14] Sandberg, S.: Homing and synchronizing sequences. In: Broy, M. et al. (eds.), Model-Based Testing of Reactive Systems. Lect. Notes Comput. Sci., vol. 3472, pp. 5–33. Springer (2005)

[15] Trahtman, A.: The Road Coloring Problem. Israel J. Math. 172(1), 51–60 (2009)

[16] Trahtman, A.N.: Modifying the upper bound on the length of minimal synchronizing word. In: Owe O., Steffen M., Telle J.A. (eds.), Fundamentals of Computation Theory, Lect. Notes Comput. Sci, vol. 6914, pp. 173–180. Springer (2011)

[17] Volkov, M.V.: Synchronizing automata and the Černý conjecture. In: Martín-Vide, C., Otto, F., Fernau, H. (eds.), Languages and Automata: Theory and Applications. Lect. Notes Comput. Sci., vol. 5196, pp. 11–27. Springer (2008)