RAIRO-Theor. Inf. Appl. 55 (2021) 7
https://doi.org/ 10.1051/ita/2021005

RAIRO - Theoretical Informatics and Applications
www.rairo-ita.org

SYNCHRONIZING SERIES-PARALLEL DETERMINISTIC FINITE AUTOMATA WITH LOOPS AND RELATED PROBLEMS

Jens Bruchertseifer and Henning Fernau*

Abstract. We study the problem DFA-SW of determining if a given deterministic finite automaton A possesses a synchronizing word of length at most k for automata whose (multi-)graphs are TTSPL, *i.e.*, series-parallel, plus allowing some self-loops. While DFA-SW remains NP-complete on TTSPL automata, we also find (further) restrictions with efficient (parameterized) algorithms. We also study the (parameterized) complexity of related problems, for instance, extension variants of the synchronizing word problem, or the problem of finding smallest alphabet-induced synchronizable sub-automata.

Mathematics Subject Classification. 68Q25, 68Q45.

Received January 2, 2020. Accepted March 25, 2021.

1. Introduction

Černý's conjecture is arguably the most famous open combinatorial problem concerning deterministic finite automata (DFA), somehow dating back to [9]. Recently, a particular Special Issue was dedicated to this conjecture being around for more than five decades; see [51]. This Special Issue also contains an English translation of Černý's paper [10]. The key notion is that of a synchronizing word. A word x is called *synchronizing* for a DFA A, if there is a state s_f, also called the *synchronizing state* of A, such that if A reads x starting in any state, it will end up in s_f. Good reviews of this and related notions can be also found in [43, 49]. The Černý conjecture states that every n-state DFA can be synchronized by a word of length $(n-1)^2$ if it can be synchronized at all [8]. Although this upper bound was proven for several classes of finite-state automata, the general case is still widely open. The currently best upper bound is cubic, and only very little progress has been made, basically improving on the multiplicative constant factor in front of the cubic term, see [22, 37, 44, 46].

The notion of a synchronizing word is not only important from a mathematical perspective, offering a nice combinatorial question, but it is quite important in a number of application areas, simply because synchronization is an important concept for many applied areas: parallel and distributed programming, system and protocol testing, information coding, robotics, etc. Therefore, it is also interesting to compute a shortest synchronizing word. Unfortunately, as it was shown by Rystsov and Eppstein in [15, 40], the corresponding decision problem DFA-SW is NP-complete. We only refer to [30] for explaining several possible applications of this problem. The problem has also been considered from the viewpoint of approximation [2] and parameterized complexity [19, 36]. Let us now define this problem central to our paper.

Keywords and phrases: Synchronizing words, series-parallel graphs, parameterized complexity.

Theoretische Informatik, Abteilung Informatikwissenschaften CIRT, Fachbereich 4, Universität Trier, Germany.

* Corresponding author: fernau@uni-trier.de

DFA-Synchronizing Word (DFA-SW)

Input: DFA A, $k \in \mathbb{N}$

Problem: Is there a synchronizing word w for A with $|w| \leq k$?

The practical importance of this question can be also seen by the fact that since decades, heuristics for solving this problem have been published; we only mention one publication of Rho *et al.* [39] at one of the major circuit design events. Also, its importance in relation to testing is seen by publications like [11].

It is important to note at this point that we are working with complete DFAs in our definition, *i.e.*, a DFA can be specified as $A = (Q, \Sigma, \delta, q_0, F)$, where Q is the finite state alphabet, Σ is the finite input alphabet, $\delta : Q \times \Sigma \to Q$ is the complete transition function, and $q_0 \in Q$ and $F \subseteq Q$ specify the start state and the final state set that are actually inessential for the task of synchronization. While DFA-SW is NP-complete, the complexity picture actually changes if we allow undefined transitions (*i.e.*, if we move over to partial DFAs); the corresponding synchronization problem then becomes PSPACE-complete; see [34].

Several papers deal with subregular classes of languages and discuss both combinatorial and computational aspects of synchronizing words with respect to these classes; the papers [1, 15, 24, 29, 32, 33, 36, 38, 41, 45, 52] form only an incomplete list of such investigations. Restrictions to automata based on graph-theoretic properties are less frequently considered compared to other possibilities for defining subregular classes. Here, we study series-parallel automata graphs and related concepts. One of our motivations was that the inductive definition of series-parallel graphs often allows for efficient algorithms for problems on such graphs that are NP-hard on general graphs. The situation is quite diverse (and hence interesting) when considering DFA-SW restricted to such-defined automata classes, as we will see.

Throughout this paper, we assume the reader to be familiar with basic notions related to automata theory, graph theory and complexity theory. Yet, the crucial concepts of this paper will be introduced below whenever needed in order to keep this paper self-contained. Having said this, we highlight the main results.

- We introduce two-terminal series-parallel automata (graphs) with loops (TTSPL) and give several characterizations and properties thereof.
- We show that DFA-SW remains NP-complete on TTSPL automata.
- Several parameterizations of DFA-SW are studied (also with a focus on TTSPL automata), the most interesting one arguably being the length upper bound k, where besides W[2]-hardness also membership in WNL is shown and discussed, linking to other problems studied in the literature. This could also raise further interest in the parameterized complexity class WNL, defined by Guillemot in [25], as mostly formal language problems are known to be complete for this class.
- In another paper [5], we studied other aspects of parameterized complexity in connection with DFA-SW, parameterized by a length upper bound k on the synchronizing word. For instance, we could prove FPT equivalence to Monoid Factorization (as defined in [7]: Given n mappings $f_i : X \to X$ and a target mapping $g : X \to X$, can g be represented by composing at most k mappings, *i.e.*, is $g = f_{i_1} \circ \cdots \circ f_{i_{k'}}$ for some $k' \leq k$?) and to Bounded DFA-intersection (Given a finite set of DFAs, is there a word of length k accepted by all of them?). This might yield an interesting (new?) parameterized complexity class between W[2] and WNL which we suggest to call W[sync] in [5]. Without going into details here, we mention that we could prove the further inclusions W[sync] $\subseteq$ W[P] and W[sync] $\subseteq$ A[2], which also suggests that W[sync] should be considered as a complexity class on its own.
- Supplementing [20], we show some more results on extension variants of DFA-SW. In particular, with respect to some length-lexicographical ordering, the extension variant of DFA-SW, parameterized by the length of the string that is to be extended, is hard for co-W[sync] and contained in co-WNL, but it is unclear if it is contained co-W[sync]. With respect to the subsequence ordering, we obtain a W[3]-hardness results for the extension variants, even on TTSPL graphs. This is interesting, as few natural problems are known to be hard for higher levels of the W-hierarchy of parameterized complexity.

– Finally, we turn to the Minimum Synchronizable Sub-automaton problem introduced in [47]. We again look at parameterized complexity aspects and prove that its natural parameterization leads again to a problem between W[2] and WNL, where we do not know its exact position, nor its relation to the previously considered problems in a similar situation.

Only (most of) the results mentioned in the first three items have been contained in the conference version of this paper, presented at NCMA 2019 in Valencia. Further results (on parameterized complexity mentioned in this paper) are presented with more details in [5]. The paper concludes with listing the most interesting open problems mentioned throughout the text.

2. Series-parallel automata graphs and generalizations

The study of series-parallel multi-graphs as automata multi-graphs was mainly undertaken by Gulan [26, 27], extending them to so-called SPL multi-graphs. In this section, we will first see that this concept does not make good sense in a classical treatment of synchronizability. Yet, a slight variation, namely, allowing self-loops in automata multi-graphs, in particular to the source and sink nodes, helps overcome this problem. In fact, we show that automata with such multi-graphs always have relatively short synchronizing words.

We first introduce some general terminology for directed multi-graphs. A *directed multi-graph* is specified as $G = (V, A)$, where V is the finite set of *vertices* and A is a finite multi-set of *arcs*. Each arc is an element (u, v) of $V \times V$, where u is called the *tail* and v is called the *head* vertex. An arc of the form (x, x) is called a *loop*. If A is in fact a set (all elements occur once), then G is a *directed graph*. Two directed multi-graphs $G_1 = (V_1, A_1)$ and $G_2 = (V_2, A_2)$ are called *disjoint* if $V_1 \cap V_2 = \emptyset$. The *union* of the two disjoint multigraphs G_1, G_2 yields the multi-graph $G = (V_1 \cup V_2, A_1 \cup A_2)$. The two composition operations defined next can be thought of first producing the union and then identifying certain vertices.

Now, we are ready to introduce a class of directed multi-graphs that has been studied for quite some time, mostly under algorithmic aspects. Many graph problems that are hard on general graphs can be efficiently solved on such recursively defined graph classes; refer to [16] for further discussions.

Definition 2.1. A directed multi-graph G is *two-terminal series-parallel*, or TTSP for short,[1] with terminals s (the *source*) and t (the *sink*), if it can be produced by a sequence of the following operations:

1. Create a new graph, with two vertices s and t and a single arc directed from the source s to the sink t.
2. Given two disjoint TTSP multi-graphs X and Y, with sources s_X and s_Y, respectively, and sinks t_X and t_Y, respectively, form a new multi-graph $G = P(X, Y)$, with source s and sink t, by identifying $s = s_X = s_Y$ and $t = t_X = t_Y$ in the union of X and Y. This is known as the *parallel composition* of X and Y.
3. Given two disjoint TTSP multi-graphs X and Y, with sources s_X and s_Y, respectively, and sinks t_X and t_Y, respectively, form a new multi-graph $G = S(X, Y)$, with source s and sink t, by identifying $s = s_X$, $t_X = s_Y$ and $t = t_Y$ in the union of X and Y. This is known as the *series composition* of X and Y.

As an aside, let us mention that there is an equivalent characterization of TTSP multi-graphs by an internal inductive definition, starting again with $G = (\{s, t\}, \{(s, t)\})$ and allowing to either subdivide any arc or to add parallel arcs to existing ones. Gulan rather follows this definition in his works.

An automaton (multi-graph) is called TTSP if the directed multi-graph obtained by removing all arc labels from an automaton graph is TTSP. Observe that TTSP automata have a sink state which should be the synchronizing state (if the automaton would be synchronizable at all), but as there is no loop attached to the sink state, synchronization is not possible. Therefore, we consider the following slight generalization of TTSP multi-graphs:

Definition 2.2. A directed multi-graph $G = (V, A)$ is *two-terminal series-parallel with loops*, or TTSPL for short, with terminals s (the *source*) and t (the *sink*), if it can be produced from a TTSP multi-graph $G' = (V, A')$ by adding any number of loops (x, x) to G', *i.e.*, $A \setminus A'$ has loops only.

[1] Valdes, Tarjan and Lawler called such directed multi-graphs *edge series-parallel (ESP) multidigraphs* in [48].

Notice that Gulan defined so-called SPL multi-graphs by allowing another arc operation, namely the contraction of an arc to form a loop, paying special care not to produce loops at the source or sink states. Hence, SPL multi-graphs and TTSPL multi-graphs differ in two ways:

- TTSPL multi-graphs may possess loops at terminal nodes, while SPL multi-graphs may not.
- SPL multi-graphs may possess bigger cycles than just loops, because arcs added as loops can be further processed, for instance, they may be subdivided, in further inductive steps.

There is an alternative way of introducing loops into TTSP multi-graphs, giving a characterization of TTSPL multi-graphs, following the inductive definition of TTSP multi-graphs:

Definition 2.3. A directed multi-graph G is *two-terminal series-parallel with loops*, or TTSP with loops for short, with two dedicated vertices, called terminals s (the *source*) and t (the *sink*), if it can be produced by a sequence of the following operations:

1. $G = (\{s,t\}, \{(s,t)\})$ (create a new graph).
2. Build $G = P(X,Y)$ (parallel composition) or
3. build $G = S(X,Y)$ (series composition) from two given TTSP multi-graphs X, Y with loops. Notice that when identifying vertices, we collect all possibly existing loops attached to the original vertices, which is possible as we deal with multi-graphs.
4. Add a loop to a terminal node of a given TTSP multi-graph with loops.

TTSP multi-graphs with loops have no cycles but loops. By induction, the next result follows.

Proposition 2.4. *A directed multi-graph G is a TTSPL multi-graph if and only if it is a TTSP multi-graph with loops.*

Notice that the special nodes s and t of TTSP multi-graphs, characterized by having in-degree 0 and out-degree 0, respectively, may lose these properties when building TTSPL multi-graphs out of them; yet, we will still maintain these two designated vertices s and t for TTSPL multi-graphs, as well. The following combinatorial characterization is rather easily seen by induction.

Lemma 2.5. *A directed multi-graph $G = (V, A)$ with two dedicated vertices s, t is a TTSPL multi-graph with source s and sink t if and only if the multi-graph $G' = (V, A')$ obtained from G by removing all loops is a TTSP multi-graph with source s and sink t.*

This lemma is important, as it shows that we can recognize if a given directed multi-graph G is a TTSPL multi-graph in linear time, based on the well-known linear-time algorithm for recognizing TTSP multi-graphs ([48], Sect. 3.3).

Recall that we can obtain a directed multi-graph from the automaton graph of a finite automaton by dropping the arc labels (typically signifying single letters). If we start out with a complete DFA, then it is clear that the resulting multi-graph is out-regular, or more precisely r-out-regular for some positive integer r, meaning that every vertex is the tail vertex of exactly r many arcs. Namely, take r as the size of the input alphabet Σ. The converse is also true by an arbitrary Σ-labeling f of the arcs that satisfies the property that f, restricted to the set A_v of arcs emanating from any vertex v, is a bijection. This reasoning shows in particular the following consequence.

Corollary 2.6. *It is decidable in linear time if a given directed multi-graph is both a TTSPL multi-graph and resulting from the automaton multi-graph of a complete deterministic finite automaton by dropping arc labels.*

Another interesting combinatorial observation comes from the fact that TTSPL automata have a single sink state, namely, t, and hence they possess a synchronizing word of length upper-bounded by a quadratic bound, see [50], because they are clearly synchronizable if they arise from deterministic finite automata. We can strengthen this consequence as follows.

Proposition 2.7. *Let A be an n-state DFA with input alphabet Σ whose multi-graph is TTSPL. Then, A possesses a synchronizing word of length upper-bounded by $(n-1)\cdot\min\{\frac{n}{2},|\Sigma|\}$.*

Proof. Assume that the DFA A has a shortest synchronizing word of length k. The bound $k \leq \frac{(n-1)n}{2}$ follows from [50]. Let w_Σ contain every symbol from the input alphabet Σ exactly once, in any order. Apart from the unique sink state t, for every other state q, there is a state $q' \neq q$ that can be reached from q by reading one symbol. As loops are the only cycles in TTSPL multi-graphs, w_Σ^{n-1} is synchronizing. □

Considering automata whose automata multi-graphs are acyclic (DAGs) after removing all loops gives a natural generalization of TTSPL automata. Notice that this type of generalization has been considered before by Ryzhikov and others in the context of synchronization before [42]. Ryzhikov termed these automata *weakly acyclic*, but other names have been also used for them in the literature. For instance, they correspond to so-called partially ordered automata as studied in [6].

Remark 2.8. In view of the inductive definition of TTSPL automata, it is tempting to introduce a class $\mathcal{R}_{SPL}$ of regular expressions (over Σ) as follows. (a) Each letter a is in $\mathcal{R}_{SPL}$. (b) If $r_1, r_2 \in \mathcal{R}_{SPL}$, then $r_1 r_2$, $(r_1 + r_2) \in \mathcal{R}_{SPL}$. (c) If $r \in \mathcal{R}_{SPL}$ and $a \in \Sigma$, then $ra^* \in \mathcal{R}_{SPL}$. However, this (only) describes a superclass of languages recognizable by TTSPL automata because of missing determinism and completeness conditions. Consider for example $a^*(a+b)$. Adding such conditions to the definition of $\mathcal{R}_{SPL}$ would make it look more clumsy.

So, in a combinatorial sense, DFAs whose multi-graph is TTSPL behave quite nicely. However, this impression changes a bit when turning towards the natural complexity questions related to synchronizing words, as we will do in the next section. In order to properly speak about certain aspects of TTSPL, we introduce some further notions now.

A vertex-induced sub-multi-graph $G' = (V', A')$ of a TTSPL multi-graph $G = (V, A)$ with source s and sink t is called a *strand* if $s, t \in V'$ and $G' - s$ is also a TTSPL multi-graph. This implies that there is a vertex $s' \in V'$ that has only one predecessor, *i.e.*, if $(x, s') \in A$, then $x = s$. Notice that in general a TTSPL multi-graph possesses several strands; they will always share (only) the vertices s and t and a possible arc from s to t. If there is a path of length at least two from u to v in G, but also an arc a from u to v, then a is called a *shortcut* in G.

3. Classical complexity aspects

As the main result of this section, we prove that it is NP-hard to determine if a DFA with a TTSPL automaton multi-graph has a synchronizing word of length at most k.

The aforementioned proof of Rystsov and Eppstein uses very simple multi-graphs, which is adapted as follows in order to produce TTSPL multi-graphs. Let φ be a Boolean formula in conjunctive normal form with n variables $x_1, x_2, \ldots, x_n$ and m clauses. The corresponding automaton is constructed as follows (refer to Fig. 1): The state set Q contains $mn + n$ states $q_{i,j}$ for $1 \leq i \leq m$ and $1 \leq j \leq n+1$, a state q_t and in addition to the original construction, a state q_s. The alphabet Σ, which originally only contains σ_1 and σ_2, is extended with m new symbols $\sigma_3, \sigma_4, \ldots, \sigma_{m+2}$, whereas the transition function δ is characterized by the following properties:

- It encompasses $\delta(q_{i,n+1}, \sigma_p) = q_t$ for $p \in \{1, 2\}$.
- If the i-th clause of φ contains the literal x_j (*i.e.*, if variable x_j is contained as a positive literal in this clause), there will be a transition $\delta(q_{i,j}, \sigma_1) = q_t$, otherwise $\delta(q_{i,j}, \sigma_1) = q_{i,j+1}$. In case that it contains $\neg x_j$ (*i.e.*, if variable x_j is contained as a negative literal in this clause), there will be a transition $\delta(q_{i,j}, \sigma_2) = q_t$, or else $\delta(q_{i,j}, \sigma_2) = q_{i,j+1}$.
- In addition to the original construction, there will be transitions
  - $\delta(q_s, \sigma_k) = q_s$ for $k \in \{1, 2\}$,
  - $\delta(q_s, \sigma_k) = q_{k-2,1}$ for $3 \leq k \leq m+2$,
  - $\delta(q_{i,j}, \sigma_k) = q_{i,j}$ for all $q_{i,j}$ and $3 \leq k \leq m+2$ as well as

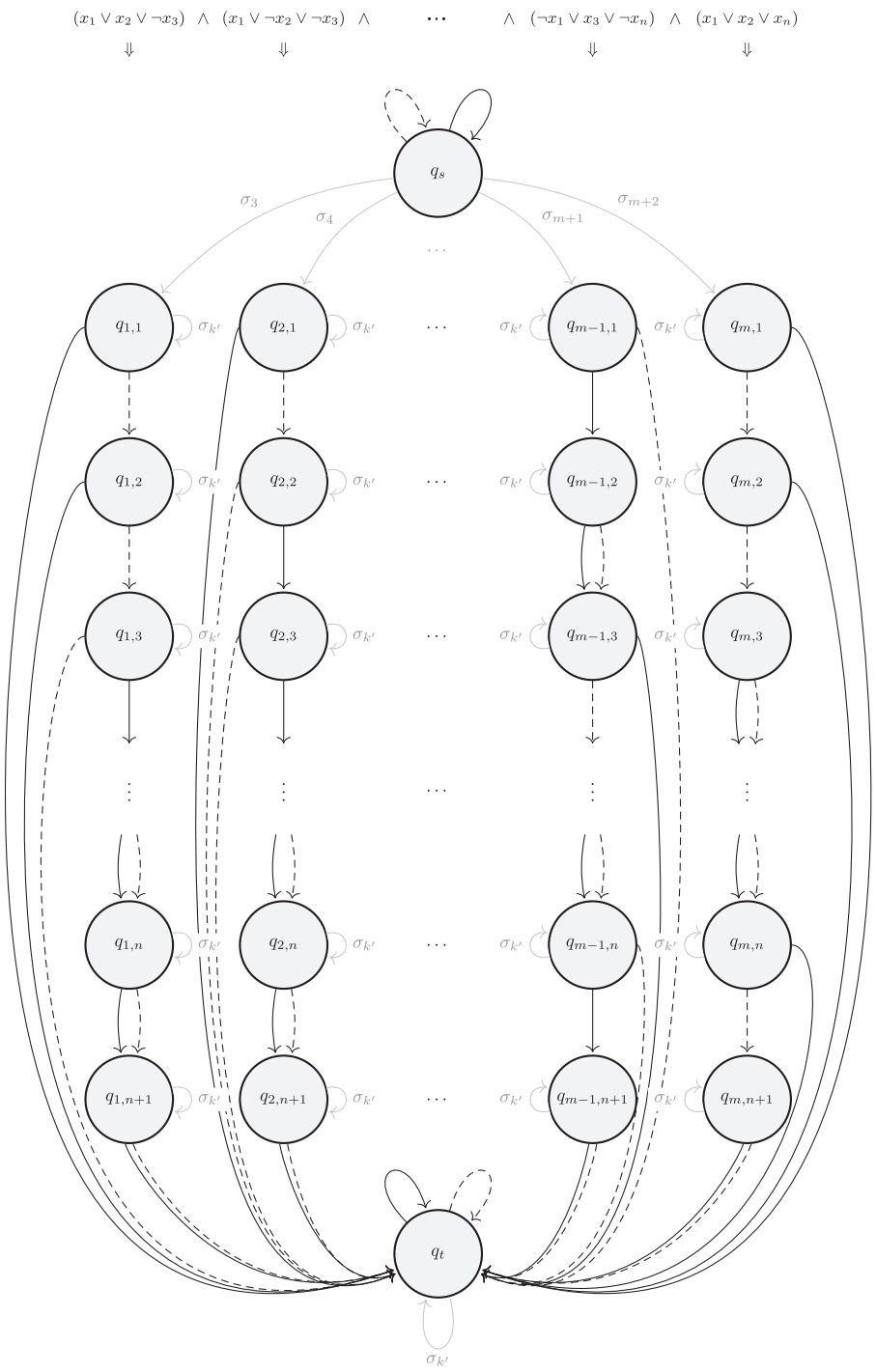

FIGURE 1. Structure of the extended Rystsov-Eppstein graph: Transitions with σ_1 and σ_2 are labeled by black solid and dashed lines, respectively. Edges created by new input symbols are colored gray and labeled with the corresponding symbol. The loops classified with "$\sigma_{k'}$" represent all loops which result from the symbols σ_k for $3 \leq k \leq m+2$.

- $\delta(q_t, \sigma_k) = q_t$ for $3 \leq k \leq m+2$.

The initial state and the final states are of no relevance for DFA-SW, so we omit them here. We call the resulting multi-graphs *Rystsov-Eppstein*. The following is quite obvious:

Lemma 3.1. *The extended Rystsov-Eppstein multi-graphs meet the definition of TTSPL multi-graphs.*

With the extended construction, it is possible to state the following result.

Theorem 3.2. DFA-SW *remains NP-complete on DFAs with a TTSPL automaton multi-graph.*

Proof. As in the construction of Rystsov and Eppstein, we will present a reduction from 3-SAT to DFA-SW, but this time the synchronizing word is exactly one symbol longer than before. Let φ be some Boolean formula with n variables $x_1, x_2, \ldots, x_n$ as well as m clauses and $A = (Q, \Sigma, \delta, q_0, F)$ the automaton built with the extended Rystsov-Eppstein construction. There is always a synchronizing word of length $n+2$; for all states except s_t, there exists even a (partially) synchronizing word of length $n+1$. So a synchronizing word for A, of length of $n+1$ or less, has to use at least one shortcut before arriving at q_t. Due to the construction, the synchronizing word has to start with σ_k for $3 \leq k \leq m+2$, followed by a sequence of σ_1 and σ_2. This requires that at least one path from $q_{i,1}$ for each $1 \leq i \leq m$ needs to take a shortcut if there is a synchronizing word of length $n+1$. The strands going through each $q_{i,1}$ represent the clauses again, while the states within a strand depict the variables. Therefore, after removal of the first letter we can read off the corresponding assignment. We give some more details in the following.

If the assignment $\alpha : \{x_1, \ldots, x_n\} \to \{0, 1\}$ evaluates φ to true, there has to be a literal in every clause which is evaluated to true. The synchronizing word is $\sigma_k w_\alpha$, where σ_k for $3 \leq k \leq m+2$ is an arbitrary input symbol and $w_\alpha \in \{\sigma_1, \sigma_2\}^n$ corresponds to the truth assignment α as explained in the following. Let $w_\alpha = a_1 a_2 \cdots a_n$. Let $a_j = \sigma_1$ if $\varphi(x_j) = 1$ and $a_j = \sigma_2$ if $\varphi(x_j) = 0$. Consider clause c_i. Let x_j be the first variable (with respect to the ordering $x_1 < x_2 < \cdots < x_n$ on the set of variables) that appears in c_i (either positive or negative) and that is set by the assignment α such that c_i becomes true. Then, $\delta(q_{i,1}, \sigma_{i+2}\sigma_\alpha) = \delta(q_{i,1}, a_1 a_2 \cdots a_n) = \delta(q_{i,2}, a_2 \cdots a_n) = \cdots = \delta(q_{i,j}, a_j \cdots a_n) = \delta(q_{i,j}, a_j) = q_t$. The assumption that x_j is the first variable that makes the clause c_i to evaluate to true means that $\delta(q_{i,\ell}, a_\ell) = q_{i,\ell+1}$ by our construction for $\ell = 1, \ldots, j-1$. Moreover, if x_j appears in a positive literal, then $a_j = \sigma_1$ and $\delta(q_{i,j}, a_j) = q_t$ by construction of δ, while if x_j appears in a negative literal, then $a_j = \sigma_2$ and again $\delta(q_{i,j}, a_j) = q_t$ by construction of δ. The self-loops at q_t guarantee that $\delta(q_{i,j}, a_j \cdots a_n) = \delta(q_{i,j}, a_j) = q_t$. Because of the connections from $q_{i,n+1}$ to q_t, it is clear that for $\ell > 1$, $\delta(q_{i,\ell}, \sigma_k w_\alpha) = q_t$, because $|w_\alpha| = n$. Finally, as $\delta(q_s, \sigma_k w_\alpha) = \delta(q_{k-2,1}, w_\alpha)$, this again leads to q_t by our discussion on starting in $q_{i,1}$.

Similarly, a truth assignment α_w satisfying φ can be read off from any word w synchronizing A that is of length $n+1$ or less. Namely, as $\delta(q_s, w) = q_t$, by construction we can assume that w starts with σ_k for some $3 \leq k \leq m+2$. As any further occurrences of symbols from $\Sigma \setminus \{\sigma_1, \sigma_2\}$ would lead to loops within states from $Q \setminus \{q_s\}$, we can assume that $w \in \{\sigma_k \mid 3 \leq k \leq m+2\} \cdot \{\sigma_1, \sigma_2\}^*$. Then, we can observe that in each strand S_i given by $\{q_s, q_t\} \cup \{q_{i,\ell} \mid 1 \leq \ell \leq m\}$, a word w from $\{\sigma_1, \sigma_2\}^*$ would synchronize all states in $S_i \setminus \{q_s\}$ iff $|w| > n$ or there is some $\ell_i < n$ such that w drives the automaton through $q_{i,1}$, $q_{i,2}$, …, q_{i,ℓ_i}, q_t. By construction, a switch from q_{i,ℓ_i} to q_t is possible only if the assignment α_w defined next satisfies the clause c_i that corresponds to strand S_i. First observe that we can exclude long synchronizing words by setting the upper bound accordingly. Then, for any clause c_i, by construction we find that the ℓ_i-th letter of w is σ_1 if x_{ℓ_i} occurs as a positive literal in c_i, and the ℓ_i-th letter of w is σ_2 if x_{ℓ_i} occurs as a negative literal in c_i. This setting is consistent as the automaton is deterministic. Then, we set the variable x_{ℓ_i} to true if the ℓ_i-th letter of w is σ_1, and we set the variable x_{ℓ_i} to false if the ℓ_i-th letter of w is σ_2. For variables not set to true or false in this way, we can (arbitrarily) decide to set them true. This defines an assignment α_w that satisfies the given formula φ, because each clause c_i is satisfied by construction.

The NP-completeness follows, as a simple guess-and-check approach would work in polynomial time. □

When dealing with (directed multi-) graphs, sometimes further structural properties are useful, as they might lower the computational complexity of certain computational problems. One such structural (or rather

topological) aspect is planarity. However, this is not very helpful because of the following observation that can be easily deduced by induction.

Proposition 3.3. *TTSPL multi-graphs are planar.*

Hence, in particular the extended Rystsov-Eppstein multi-graphs are planar, as can be also seen by looking at the drawing in Figure 1. Our studies hence fit into the considerations on planar instances of automata multi-graphs undertaken in [36] in the context of DFA-SW.

4. Aspects of parameterized complexity

In view of our previous observations, let us now try to find parameterizations that allow us to characterize easy aspects of finding short synchronizing words in TTSPL automata. For reasons of space, we only refer to recently published textbooks [12, 14], in particular for the discussion of parameterized complexity classes that we will make use of in the following. Appropriate for parameterized algorithms is using the $\mathcal{O}^*$-notation; for instance, a problem solvable in time $\mathcal{O}^*(f(k))$ refers to an algorithm running in time $\mathcal{O}(f(k)p(n))$, where n is the input size, p is some polynomial, f is some arbitrary (computable) function and k is the so-called parameter, some secondary measurement of the input. A problem (with parameter k) is in FPT (or, *fixed-parameter tractable*) if it can be solved in time $\mathcal{O}^*(f(k))$. We will also present lower bound results based on the (Strong) Exponential Time Hypothesis, or (S)ETH for short, which basically assumes that Satisfiability cannot be solved (much) faster than currently known. For details, we again refer to the mentioned textbooks. A brief introduction into this area, more tailored towards the theory of formal languages, can be also found in [17].

In [19], several natural parameterizations have been studied for arbitrary automata graphs. We will discuss these results next and then see how they translate to our more restricted scenario.

- As DFA-SW remains NP-hard for binary input alphabets, $|\Sigma|$ is not an interesting parameter on its own.
- An upper bound k on the length of a synchronizing word as the sole parameter is known to lead to some hardness results. More specifically, it was first shown in [19] that this problem is W[2]-hard.
- Combining both parameters, one arrives at a rather trivial FPT-result by enumerating and testing all words up to length k. However, assuming SETH, it was observed in [19] that there is no $\mathcal{O}^*((|\Sigma| - \varepsilon)^k)$ algorithm for any $\varepsilon > 0$. This proof is based on the Rystsov-Eppstein construction, so that it carries over to our restricted scenario. By the same type of argument, it could be shown that there is no polynomial-size kernel for our problem, parameterized by $|\Sigma|$ and k, unless an unlikely collapse of the polynomial-time hierarchy occurs. Again the argument builds upon Rystsov-Eppstein graphs and can be hence carried over. We will make this explicit below.
- Another natural parameter choice is the number of states $|Q|$ of the automaton. Basically, by following the well-known subset automaton construction, also see [43], it is clear that a shortest synchronizing word can be found in time $\mathcal{O}^*(2^{|Q|})$, and in Lemma 11 from [19], a modification of the Rystsov-Eppstein construction was used to prove a matching lower bound, based on ETH. We will see below that the same lower bound still holds for TTSPL automata graphs.

4.1. More on the length parameter k: an excursion into parameterized complexity

Let A be some DFA with state set Q, input alphabet Σ, transition function $\delta : Q \times \Sigma \to Q$. A word $w \in \Sigma^*$ is called P-synchronizing for a state subset $P \subseteq Q$ if w drives P into a single state $s_{P,f}$. Obviously, a Q-synchronizing word is just a synchronizing word. Montoya and Nolasco considered in [36] the following problem variant:

DFA-Subset-sbSW (where sb indicates a size bound)

Input: DFA A with state set Q as well as $k, m \in \mathbb{N}$ and subset $P \subseteq Q$ with $|P| = k$

Problem: Is there a P-synchronizing word w for A with $|w| \leq m$?

Notice that if m is given in binary and the restriction $|P| = k$ is missing, then this problem is even PSPACE-complete, because subset synchronization in itself (without the length bound) is already PSPACE-complete; see [18] for more discussions. However, if m is presented in unary, as implicit in the problem definition, because the input DFA comes with a list of its states and this list contains more elements than k, then the problem is 'only' NP-complete, as it also follows from [36], but this also follows when considering $P = Q$. However, Montoya and Nolasco focused on the parameterized complexity of DFA-Subset-sbSW and could prove the following result:

Theorem 4.1. DFA-Subset-sbSW*, parameterized with the set size parameter k, is complete for the class WNL, even when restricted to planar graph instances.*

Recall that the parameterized complexity class WNL was introduced by Guillemot in [25].[2] Consider the following problem:

Nondeterministic Turing Machine Computation (NTMC)	
Input:	Nondeterministic single-tape Turing machine M, $k, q \in \mathbb{N}$, where k and q are given in unary
Problem:	Does M accept the empty string in at most q steps, visiting at most k tape cells?

With the parameter q, this problem is known to be W[1]-complete. Guillemot defined WNL as the class of parameterized problems that can be reduced (by a parameterized many-one reduction) to NTMC with parameter k. Interesting formal language problems complete for WNL include Bounded DFA-Intersection – given k DFAs, with parameter k, plus the length of the searched word that should be accepted by all k automata – or Longest Common Subsequence, parameterized by the number of given strings. WNL is situated above all levels of the W-hierarchy, because the last two mentioned problems are known to be hard for W[t] for any $t \geq 1$, see [4, 54]. Each level of the W-hierarchy can be characterized by specific weighted satisfiability problems; we refrain from giving details here but refer to textbooks like [12, 14]. Furthermore, if a parameterized problem Π can be solved in time $n^{f(k)}$ for instances (x, k) of size n for some function f, then $\Pi \in$ XP. Let us now return to our problems related to synchronizing words. Notice that for general graphs, the WNL-hardness result from Theorem 4.1 also holds for binary input alphabets (on general graphs) or for four-letter input alphabets (on planar graphs). As we are more interested in the parameterized complexity of the basic problem DFA-SW, we are now going to prove, supplementing the work of Montoya and Nolasco:

Theorem 4.2. DFA-SW*, parameterized with the length parameter k, is contained in the class WNL.*

Proof. Notice that membership in WNL does not follow with the arguments presented in [36], because that argument makes explicit use of the fact that the given set P of states contains k elements. However, we now describe a nondeterministic Turing machine that visits at most $f(k)$ many cells (ever), providing the required reduction. Given a DFA A with state set Q and input alphabet Σ, where, w.l.o.g., $Q \cap \Sigma = \emptyset$, together with a bound k on the length of a synchronizing word, a Turing machine M is constructed that works as follows:

1. M writes a word of length at most k over the alphabet Σ on its tape, followed by some letter from the alphabet Q; this means that M starts with making (at most) $k + 1$ nondeterministic guesses.
2. For each $q \in Q$ (this information can be hard-coded in the finite-state memory of M), M first moves its head to the left end of its tape and then starts reading the tape content from left to right. Each time a symbol $a \in \Sigma$ is read, M updates the current state it stores according to the transition function of A. Finally, M will read a symbol from Q, and it will only continue working if this symbol equals the current state stored in the finite memory of M. Notice that (2) works deterministically.
3. Only if M has completely processed the loop described in (2) (without abort), M will accept. This verifies that the guessed word over Σ is indeed synchronizing, always leading into the state that was also previously

[2] Let us remark that WNL is one of the less known parameterized complexity classes, but we have the impression that this class is quite interesting in particular for classifying parameterized problems dealing with strings, automata and formal languages.

guessed. Hence, M will accept the empty word if and only if there is a possibility to guess a synchronizing word of length at most k. It is also clear that the Turing machine makes at most $(|Q|+1)(2k+1)$ many steps.[3]

□

Unfortunately, we do not know if our problem is WNL-hard.[4]

4.2. Discussing TTSPL automata

Let us now try to transfer the reported hardness results (lower bounds) from the case of general (planar) automata graphs to the much more restricted scenario of TTSPL automata graphs. Let us start with the alphabet size as a parameter.

Proposition 4.3. DFA-SW *is NP-complete on TTSPL automata with four input symbols.*

Proof. We can modify the construction of Theorem 3.2 by replacing the forking into m successor nodes from q_s by a binary tree, using only two new letters, say, τ_1 and τ_2, on top of σ_1 and σ_2. This will introduce (only) m additional nodes into the construction. We add loops with the original symbols σ_1 and σ_2 to the internal nodes of this tree. To nodes where previously loops were added on these m new symbols, now we add (only) loops with τ_1 and τ_2. It can be seen that the resulting automaton graph is TTSPL. □

This means that the alphabet size on its own is not a very useful parameterization. More technically speaking, this proves that there is no XP-algorithm (with parameter input alphabet size) for DFA-SW on TTSPL automata, unless P = NP.

In the following proposition we make use of the parameterized problem Hitting Set that is defined as follows.

Hitting Set

Input: A family $\mathcal{F}$ of sets over a universe $\mathcal{U}$ and an integer k, where k is considered as a parameter.

Problem: Does there exist a set $\mathcal{S} \subset \mathcal{U}$ such that

1. $|\mathcal{S}| \leq k$ and
2. $F \cap \mathcal{S} \neq \emptyset$ for each $F \in \mathcal{F}$?

A set $\mathcal{S}$ satisfying (2) is also called a *hitting set*. Hitting Set is W[2]-complete with respect to parameter k [14].

Proposition 4.4. DFA-SW*, parameterized by an upper bound k on the length of a synchronizing word, is W[2]-hard, even when restricted to DFAs with TTSPL automata multi-graphs.*

Proof. Adapting a construction from [19], we can create a DFA-SW instance from a given Hitting Set instance with universe $\mathcal{U} = \{e_1, e_2, \ldots, e_n\}$, set family $\mathcal{F} = \{F_1, F_2, \ldots, F_m\}$ and k as an upper bound on the size of the hitting set $\mathcal{S}$. We take $k+1$ as the upper bound on the length of a shortest synchronizing word. The state set Q of the DFA A consists of

- *set family states* f_i for $1 \leq j \leq m$, denoting in which set F_i an element of $\mathcal{U}$ is located, and
- *terminal states* q_s and q_t, the latter one being a trap state.

[3] Here, some definitorial details might change the constants, e.g., how 'fast' can a Turing machine detect that it is at the tape end and turn back?

[4] This was claimed to be the case in the conference version of this paper, but this was due to a mis-interpretation of the literature. In [5], we exhibit quite a number of problems that are equivalent to DFA-SW under parameterized reductions, justifying the introduction of a new parameterized complexity class that we call W[sync]. From that discussion, some indications follow why our problem DFA-SW is probably not WNL-hard.

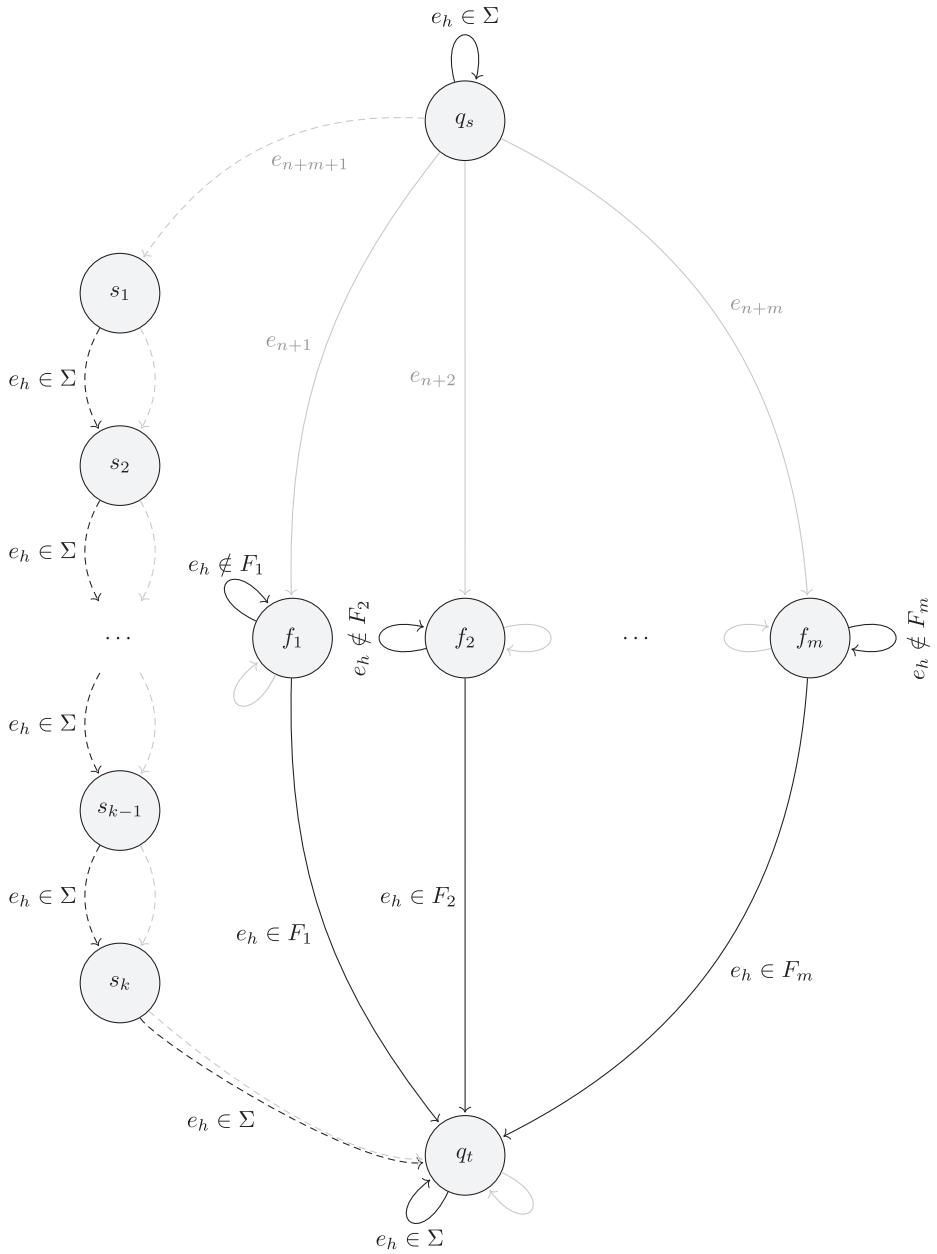

FIGURE 2. An example for the structure of the transition graph from the proof of Proposition 4.4 (containing a second extension with s_i, $1 \leq i \leq k$, as discussed in Prop. 4.10) derived from a HITTING SET instance $((\mathcal{U} = \{e_1, \ldots, e_n\}, \mathcal{F} = \{F_1, \ldots, F_m\}), k)$. Grey edges depict new symbols e_b for $n+1 \leq b \leq n+m$ (resp. $n+m+1$). For space reasons, they are only labeled if the transition is not carried out with each of the added symbols.

Note that the original construction contained one long strand which can be omitted so far and will be of importance only in the further course. The input alphabet $\Sigma = \mathcal{U}$ contained the elements e_h for $1 \leq h \leq n$ in the original construction, to which $e_{n+1}, \ldots, e_{n+m}$ are added to form the input alphabet Σ' of our automaton A. The transition function δ of A is defined as follows.

- $\delta(f_i, e_h) = f_i$ for every $e_h \notin F_i$ and $1 \leq i \leq m$, as well as
  $\delta(f_i, e_h) = q_t$ for each $e_h \in F_i$ and
- $\delta(q_t, e_h) = q_t$ for all $e_h \in \Sigma$;
- $\delta(q_s, e_{n+j}) = f_j$ for $1 \leq j \leq m$ and
- $\delta(q_s, e_h) = q_s$ for all $e_h \in \Sigma$.

The remaining transitions (which are caused by the new symbols, not belonging to Σ) are added as loops. If there is a synchronizing word for A, it has to meet the following conditions:

- Since q_t is a trap state, there can be no other state in which such a word could end.
- In order to synchronize f_i, some symbol $e_h \in F_i$ has to be part of any synchronizing word.

Therefore, a shortest synchronizing word has to be of the form $w = \sigma_0\sigma_1 \cdots \sigma_o$, where $\sigma_0 \notin \Sigma$ and $\sigma_p \in \Sigma$ for $1 \leq p \leq o$ with $o \leq k$. Considering the reasons listed above, we can read off a hitting set of size at most k from such a synchronizing word after removing the first symbol of w if it is no longer than $k+1$. Notice that $\sigma_i = \sigma_j$ is possible for $1 \leq i < j \leq k$.

For the other direction, we assume that $\mathcal{S} = \{\sigma_1, \sigma_2, \ldots, \sigma_i\}$ with $\sigma_i \in \Sigma$ for $1 \leq i \leq k$ is a hitting set with $|\mathcal{S}| \leq k$. Let $\sigma_0 \notin \Sigma$. Consider the word $\sigma_0\sigma_1 \cdots \sigma_k$ and observe that it is a synchronizing word.

With the reduction above and Hitting Set being W[2]-hard, the claim follows. □

We do not know if DFA-SW, parameterized by an upper bound k on the length of a synchronizing word, restricted to DFAs with TTSPL automata multi-graphs is as hard as the problem is on general graphs. As shown by Möhring in [35], there are quite close connections between TTSP(L) graphs and so-called series-parallel partial orders. Without going into too much detail here, observe that the mappings $Q \to Q$ that can be associated to input letters are monotone with respect to the series-parallel partial order corresponding to the TTSPL automaton multi-graph. This allows us to conclude the following result with the help of our constructions and observations above. In this result, we refer to a restricted variant of the problem that we discuss in more detail in [5].

Monoid Factorization (see [7])

Input: A finite set Q, a collection $F = \{f_0, f_1, \ldots, f_m\}$ of mappings $f_i : Q \to Q$, $k \in \mathbb{N}$

Problem: Is there a selection of at most k mappings $f_{i_1}, \ldots, f_{i_{k'}}$, $k' \leq k$, with $i_j \in \{1, \ldots, m\}$ for $j = 1, \ldots, k'$, such that $f_0 = f_{i_1} \circ f_{i_2} \circ \cdots \circ f_{i_{k'}}$?

Corollary 4.5. DFA-SW*, parameterized by an upper bound on the length of a synchronizing word, restricted to DFAs with TTSPL automata multi-graphs, is polynomial-time equivalent to* Monoid Factorization*, parameterized by an upper bound k on the generator length, restricted to collections of mappings F that are monotone with respect to a given series-parallel partial order on the finite ground set Q.*

In other words, these problems might belong to yet another class of parameterized problems, lying between W[2] and WNL. We summarize some further parameterized results below, where the hardness result follows with the proof of Proposition 4.3.

Corollary 4.6. DFA-SW*, parameterized by input alphabet size $|\Sigma|$ and an upper bound k on the length of the synchronizing word, belongs to FPT, but it cannot be solved in time $\mathcal{O}^*((((|\Sigma| - 2) - \varepsilon)^k)$ for any $\varepsilon > 0$ under SETH, and there does not exist a polynomial-size kernel, unless the polynomial-time hierarchy collapses to the third level, even if we restrict the instances to DFAs with TTSPL automata multi-graphs.*

Recall that a parameterized problem is in FPT if and only if it allows for a kernelization, *i.e.*, a parameterized self-reduction whose output has size upper-bounded by a function in the parameter. However, this bounding function could be arbitrarily bad. Of particular practical interest are polynomial-size kernels. The question of the existence of such small kernels has been one of the driving forces of the research in Parameterized Complexity

in the last decade. Typically, the non-existence of polynomial-size kernels is based on the assumption that NP is not a subset of the advice class co-NP/poly. By famous results of Yap [55], this non-existence of polynomial-size kernels could be also based on the assumption that the third level Σ_3^p of the polynomial-time hierarchy is different from PSPACE, which is putting the assumption on more common ground. More details can be also found in [12, 14].

Observe that the SETH lower bound is weaker here compared to the case of general graphs, because we seem to need four letters in our hardness constructions. We leave it as an open question if DFA-SW is NP-complete also on TTSPL multi-graphs with binary input alphabets. Notice that we cannot simply re-use the two letters σ_1 and σ_2 in the forking process described in the proof of Proposition 4.3, *e.g.*, by setting $\tau_1 = \sigma_1$ and $\tau_2 = \sigma_2$, because we have to ensure that each of the start nodes of the strands which correspond to clauses are synchronized by reading the same word that is actually a suffix of the synchronizing word of the whole TTSPL automaton.

Proposition 4.7. *Let φ be a CNF formula with n variables and m clauses. It is possible to construct from φ a DFA $A = (Q, \Sigma, \delta, q_0, F)$ with a TTSPL transition graph in time $\mathcal{O}(mn)$, such that $|Q| = n + m + 2$ and $|\Sigma| = 2n + m + 1$, which possesses a synchronizing word of length $(n + 1)$ if and only if φ is satisfiable.*

Proof. The construction from [19] can be extended as follows: Let φ be a given CNF formula, where $V = \{x_1, \ldots, x_n\}$ denotes the set of its variables and $C = \{c_1, \ldots, c_m\}$ the set of its clauses. Without loss of generality, it is assumed that no variable occurs twice in any clause, whereby every c_i can be viewed as a subset of the set of literals $L = \{x_1, \ldots, x_n\} \cup \{\neg x_1, \ldots, \neg x_n\}$. From this formula, a DFA $A = (Q, \Sigma, \delta, q_0, F)$ is constructed (as seen in Fig. 3): The set of input symbols Σ contains $2n$ symbols x_i and $\neg x_i$ for $1 \leq i \leq n$ (one of these always corresponds to a literal, so they will be referred to as l) and additionally $m + 1$ symbols y_b for $1 \leq b \leq m + 1$. The state set Q consists of $n + m + 2$ states listed below:

- *variable states* q_i for $1 \leq i \leq n$,
- *clause states* c_j for $1 \leq j \leq m$ and
- *terminal states* q_s and q_t, the latter one being a trap state.

The transition function δ is defined as follows:

- $\delta(q_i, x_i) = \delta(q_i, \neg x_i) = q_{i+1}$ with $q_{n+1} = q_t$,
  $\delta(q_i, x_{i'}) = \delta(q_i, \neg x_{i'}) = q_i$ for $i \neq i'$,
- $\delta(c_j, l) = c_j$ for every literal $l \notin c_j$,
  $\delta(c_j, l) = q_t$ for every $l \in c_j$,
- $\delta(q_t, l) = q_t$ for all literals. Compared to the original construction, this one is extended by
- $\delta(q_s, y_b) = c_b$ for $1 \leq b \leq m$,
- $\delta(q_s, y_{m+1}) = q_1$ as well as
- $\delta(q_s, x_i) = \delta(q_s, \neg x_i) = q_s$ for $1 \leq i \leq n$.
- The remaining transitions (which result from symbols y_b) are loops.

Since q_t does not have any outgoing transitions, there is no other state in which a synchronizing word could end. Due to the strand containing $q_1, \ldots, q_n$, a synchronizing word must be at least of length n and of the form

$$(y_1|y_2|\ldots|y_{m+1})(x_1|\neg x_1)^+(x_2|\neg x_2)^+\cdots(x_n|\neg x_n)^+(x_1|\neg x_1|x_2|\neg x_2|\ldots|x_n|\neg x_n)^*.$$

Such a synchronizing word represents the assignment of the variables $\Phi\colon V \to \{0, 1\}$ if the first letter is cut off. After this removal it can be generated by the assignment function

$$\Phi(x_i) = \begin{cases} 1, & \text{if } l_i = x_i \\ 0, & \text{if } l_i = \neg x_i. \end{cases}$$

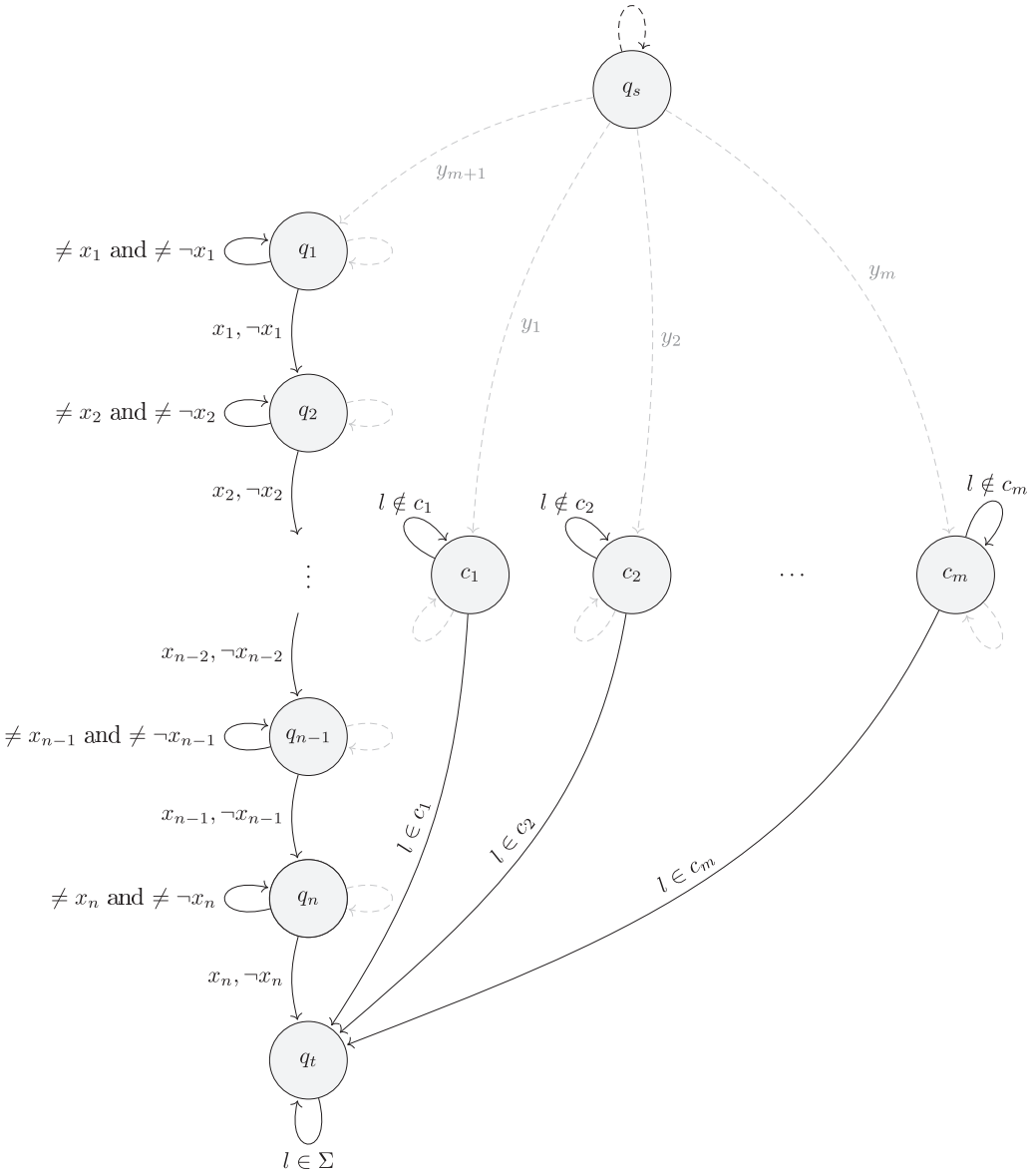

FIGURE 3. Structure of the automata graphs from Proposition 4.7: Solid edges indicate edges from the initial construction, dashed edges are added with the new construction. Grey edges result from one of the added input symbols y_b, black edges from one of the already existing symbols l. If there is no label on an edge, this transition is performed by all of the symbols y_b or l, respectively.

Considering that this word leads into q_t from every state c_j and that this transition is not possible with y_b for $1 \leq b \leq m+1$, it is clear that every clause and therefore the formula φ itself is satisfied.

For the reversed direction of this proof it should be observed that it is possible to create a word $w = l_1 l_2 \cdots l_n$ with

$$l_i = \begin{cases} x_i, & \text{if } \Phi(x_i) = 1 \\ \neg x_i, & \text{if } \Phi(x_i) = 0. \end{cases}$$

Let $y_b \in \{y_1, \ldots, y_{m+1}\}$ now be chosen arbitrarily but fixed. Obviously, the word $w' = y_b l_1 l_2 \cdots l_n$ leads from every state q_i with $1 \leq i \leq n$ into q_t. Since every clause c_j, $1 \leq j \leq m$, is satisfied, the same holds for every state c_j. It should be noted that y_b acts as the identity for q_t as well as for all states q_i. Taking into account that every possible value for y_b leads from q_s into a state that is directed to q_t by $w' = l_1 l_2 \cdots l_n$, it also directs q_s into q_t. Therefore, w' represents indeed a synchronizing word for A. □

With that result it can be stated:

Corollary 4.8. DFA-SW *on automata with a TTSPL transition multi-graph cannot be solved in time* $\mathcal{O}^*(2^{o(|Q|)})$*, unless ETH fails.*

Proof. As proven in Proposition 4.7, an arbitrary 3-SAT instance with n variables and m clauses can be reduced to an instance $A = (Q, \Sigma, \delta, q_0, F)$ of DFA-SW with $|Q| = n + m + 2$, $|\Sigma| = 2n + m + 1$ and $k = n + 1$. With the existence of an algorithm solving DFA-SW in time $\mathcal{O}^*(2^{o(|Q|)})$, 3-SAT would be solvable in time $\mathcal{O}^*(2^{o(n+m)})$, contradicting ETH. □

Unfortunately, we cannot claim any lower bounds on kernel sizes for the parameterization by state set size, because the constructions by Vorel and Roman [53] seem to destroy the TTSPL automaton multi-graph properties that we need.

So far, we looked at the question if parameterizations studied previously for general automata graphs lead to the same results when considering TTSPL multi-graphs only. Now, we discuss several ideas of parameterizations that are genuine for TTSPL. Notice that the theory of width-parameters for directed graphs is much richer than for undirected graphs (see [23, 28, 31]), and we are not aware of any further studies on width parameters for TTSP multi-graphs, something one might have considered not necessary to do, as most classical graph problems are easy on such restricted graph classes. Our studies reveal this necessity when considering automata problems.

One problem when computing a synchronizing word for an automaton with TTSPL multi-graph consists in different lengths of paths. This necessitates having shortcuts which might motivate to bound the complexity by an accordingly created parameterization. As the graph from the Eppstein-Rystsov construction contains at most three shortcuts per strand if we start out with a 3-SAT instance, a simple parameterization with the *number of shortcuts per strand* nevertheless seems quite hopeless:

Corollary 4.9. DFA-SW *remains NP-complete on TTSPL automata graphs where each strand has at most three shortcuts.*

Another feature in all our reductions is that all strands contain loops. Let us be a bit more precise: In our definition of TTSPL multi-graphs, it is pretty clear that the terminal nodes will contain loops. Therefore, when counting the number of loops in a TTSPL multi-graph, we ignore loops at terminal nodes. With a modification of a construction from [19] it is possible to state:

Proposition 4.10. *The problem* DFA-SW *on automata with TTSPL automata graphs remains NP-complete, even if there is one strand which must not contain loops (except for the terminals).*

Proof. We extend the construction from Proposition 4.4 with a strand containing k nodes s_i as well as the mandatory q_s and q_t (*cf.* Fig. 2). There will be a new input symbol e_{n+m+1}. The transitions s_i to s_{i+1} with $1 \leq i \leq k$ and $s_{k+1} = q_t$ are performed on every input symbol. Another added transition will be $\delta(q_s, e_{n+m+1}) = s_1$. The other transitions with e_{n+m+1} are added as loops again in all states except the s_i states and q_s. It is obvious that this automaton is still a valid reduction from Hitting Set. As it contains one strand without loop nodes (not counting the terminals), the claim holds. □

Hence, the number of strands that must not contain loops (except for the terminals) is not an interesting parameter, either, as the previous proposition shows that DFA-SW does not belong to XP for this parameter, unless P = NP.

As strands are also TTSPL multi-graphs by definition, but it is required that the multi-graph obtained from a strand by deleting the source node is again TTSPL, the number of strands that contain at least four nodes with

loops might be an interesting parameter. In some of our reductions, this actually coincides with the (generally larger) parameter *number of long strands*, *i.e.*, strands containing at least four nodes.

Corollary 4.11. *The problem* DFA-SW *on automata with TTSPL automata multi-graphs remains* NP*-complete, even if there is only one long strand.*

Hence, this does not give an interesting parameter, either. Since the connectedness of the graph determines the complexity to some degree, it might be tempting to use it as another parameter. However, a simple parameterization like for example the number of nodes with large indegree similar to the weft in the W hierarchy might not be very promising when taking into account that the extended Eppstein-Rystsov construction uses only one node with indegree > 3.

Corollary 4.12. DFA-SW*, parameterized with the number of nodes with indegree* > 3*, is NP-complete for parameter value 1.*

In view of the reductions we have seen so far, which are promising parameterizations—in the sense of allowing for FPT-results? We discuss some concrete points below in the conclusions.

5. Problem variations

In this section, we consider several variants of our basic problem DFA-SW and study them again for automata with TTSPL automata graphs. We will see that in most cases, the hardness results known for general automata graphs transfer to this case, as well.

5.1. Extension variants

Extension variants of DFA synchronization have been examined in [20]. The notion of extension relies on the chosen partial order on the set of all words:

- $y|w$, meaning that y is a (possibly scattered, but ordered) *subsequence* of w, *i.e.*, there are subwords $y_1, \ldots, y_r$ and $w_0, w_1, \ldots, w_r$ such that $y = y_1 \cdots y_r$ and $w = w_0 y_1 w_1 \cdots y_r w_r$.
- y *sub* w, meaning that y is a *subword* of w, *i.e.*, there are words x, z such that $w = xyz$;
- $y \sqsubseteq w$, meaning that y is a *prefix* of w, *i.e.*, there is a word z such that $w = yz$;
- $y \sqsupseteq w$, meaning that y is a *suffix* of w, *i.e.*, there is a word x such that $w = xy$.
- $v \leq_{lex} w$ (*lexicographical ordering*) means that either $v \sqsubseteq w$ or u is the longest common prefix of v and w and $ua \sqsubseteq v$, $ub \sqsubseteq w$, and $a \leq b$ for some $a, b \in \Sigma$.
- $v \leq_{ll} w$ (*length-lexicographical ordering*) means that either $|v| < |w|$ or that $|v| = |w|$ and $v \leq_{lex} w$.

In the last two cases, we assume that a total order $\leq$ is given on the alphabet Σ. Based on our previous discussions, we are now defining so-called extension problems, depending on the chosen partial order $\prec$ on Σ^*. As we will see, the complexity status of these problems will heavily depend on this choice.

Ext DFA-SW-$\prec$	
Input:	DFA A with input alphabet Σ, $u \in \Sigma^*$
Problem:	Is there a $w \in \Sigma^*$, $u \prec w$, such that w is minimal for the set of synchronizing words for A with respect to $\prec$?

Observe that we automatically face a NO-instance if the given DFA A is not synchronizable. Let us clarify this problem by a small example. Consider the automaton A displayed in Figure 4. Let b be the word given in addition, so the question is if b can be extended to a minimal synchronizing word. If $\prec = \sqsubseteq$, then ba is indeed a synchronizing word that extends b (as $b \sqsubseteq ba$) such that no proper prefix of it is synchronizing. Hence, (A, b) is a YES-instance of Ext DFA-SW-$\sqsubseteq$. Similarly, it can be seen that (A, b) is a YES-instance of Ext DFA-SW-$\sqsupseteq$, because ab is a $\sqsupseteq$-minimal synchronizing word. However, (A, b) is a NO-instance of Ext DFA-SW-$|$ and of Ext

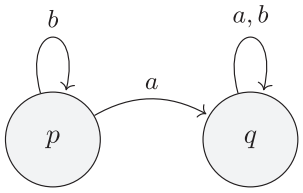

FIGURE 4. Synchronizing a small automaton.

DFA-SW-*sub*, because any synchronizing word w must contain at least one a, and a itself is a synchronizing word. Hence, in particular any synchronizing word w that extends b (with respect to $|$ or *sub*) cannot be minimal, because both $a|w$ and a *sub* w. For the (length-)lexicographic ordering, we observe the following: if $a < b$, then clearly no extension of b (as b itself, or ba, or bb, or bab etc.) can be minimal, as a is always smaller, *i.e.*, we face a NO-instance (A, b); if $b < a$, then ba is indeed a minimal synchronizing word that extends b, so that (A, b) is a YES-instance. This small example already shows that this class of extension problems could be quite diverse and rich.

In [20], the following results on general automaton graphs have been shown. We also highlight open questions for this general case.

- EXT DFA-SW-$\sqsubseteq$ and EXT DFA-SW-$\sqsupseteq$ are solvable in polynomial time.
- EXT DFA-SW-$|$ is NP-hard, allowing arbitrarily large input alphabets. For binary input alphabets (for instance), this question is *open*.
- EXT DFA-SW-*sub* poses an *open question*.
- EXT DFA-SW-$\leq_{lex}$ is solvable in polynomial time.
- EXT DFA-SW-$\leq_{ll}$ is co-NP-hard, even on binary input alphabets.
- Membership of EXT DFA-SW-$|$, EXT DFA-SW-*sub* and EXT DFA-SW-$\leq_{ll}$ in interesting complexity classes is *open*. We will resolve one of these questions below, proving membership of EXT DFA-SW-$\leq_{ll}$ in co-NP.

For the (co-)NP-hard problem variants, it is again interesting to study aspects of parameterized complexity. The most natural parameter appears to be the length of the provided string u. Let us summarize again the findings from [20].

- EXT DFA-SW-$|$, parameterized by $|u|$, is W[3]-hard. Membership in W[3] is open, but let us remark that this is one of the few parameterized problems that might find its home on a higher level of the W-hierarchy; see [3].
- The case of EXT DFA-SW-$\leq_{ll}$, parameterized by $|u|$, was *not discussed*. The construction for showing NP-hardness is not helpful, as it is basically a modification of the construction already presented by Rystsov and Eppstein (and also used throughout this paper), starting out from SATISFIABILITY, so that the length of u is the number of Boolean variables of the original instance.

We are first describing another proof of co-NP-hardness of EXT DFA-SW-$\leq_{ll}$ that also shows some parameterized complexity result; this is the way we state our theorem in the following. However, notice that this reduction is not a parameterized many-one reduction, because the YES- and NO-answers are switched.

Theorem 5.1. EXT DFA-SW-$\leq_{ll}$, *parameterized by* $|u|$, *is contained in co-WNL, but one can reduce* DFA-SW, *parameterized by a length upper bound* k, *to the complement problem of* EXT DFA-SW-$\leq_{ll}$, *parameterized by* $|u|$.

Proof. For membership in co-WNL, we can modify the proof of Theorems 4.2, constructing a nondeterministic Turing machine M as follows, given A and u. As in the previous construction, the machine can first guess a possible word $w \leq_{ll} u$ and verify if it is synchronizing. If such a word is found, then (A, u) is a NO-instance. The

reduction itself checks if A is synchronizable at all, which can be done in polynomial time according to [43, 49]. We also have that if M does not find a synchronizing word $w \leq_{ll} u$, then (A, u) is a YES-instance, because as A is synchronizable, there must be a synchronizing word v. Remind that according to the previous tests, $u \leq_{ll} v$ must hold.

We are now turning to the the second claim. Consider a DFA A on the input alphabet Σ, together with a number k, as an instance of DFA-SW. We can first check in polynomial time if A is synchronizable at all. If A is not synchronizable, then (A, k) (clearly) is a NO-instance of DFA-SW, so our reduction will produce some fixed NO-instance of Ext DFA-SW-$\leq_{ll}$. Hence, we now assume that A is synchronizable. Let $c \notin \Sigma$ be a fresh letter. Consider an arbitrary ordering $<$ on Σ, extended by $c < x$ for all $x \in \Sigma$ towards an ordering on $\hat{\Sigma} = \Sigma \cup \{c\}$. We are going to define the DFA $\hat{A}$ as an extension of A, working on the same state set Q. Let c simply act as the identity on Q. Hence, no word from c^* is synchronizing for $\hat{A}$. As A is synchronizable, $\hat{A}$ is also synchronizable. Consider $\hat{A}$ together with $u = c^{k+1}$ as an instance of Ext DFA-SW-$\leq_{ll}$. If $\hat{A}$ has a synchronizing word w of length at most k, then clearly u is not extendible, as $|w| < |u|$. Otherwise, as $\hat{A}$ is synchronizable, $\hat{A}$ must have some synchronizing word w with $|w| \geq |u|$, and any synchronizing word of $\hat{A}$ is of length at least $|u|$. As u is the smallest of all words in $\hat{\Sigma}^*$ of length at least $|u|$, any synchronizing word will hence extend u. Hence, if $\hat{A}$ has no synchronizing word of length at most k, then u is extendible. □

Notice that the last reduction in the previous proof is not a many-one reduction, as it reverses YES- and NO-answers. Hence, this reduction proves that Ext DFA-SW-$\leq_{ll}$, parameterized by $|u|$, is hard for the parameterized complexity class consisting of all complements of parameterized problems FPT-equivalent to DFA-SW.

It is not so clear if one can solve Ext DFA-SW-$\leq_{ll}$ with the help of DFA-SW. Clearly, given A and u, one can check with the instance $(A, |u| - 1)$ if there are synchronizing words shorter than u, which would automatically imply that u is not extendible. If $(A, |u| - 1)$ is a NO-instance of DFA-SW and if $(A, |u|)$ is a NO-instance of DFA-SW, then (A, u) is a YES-instance of Ext DFA-SW-$\leq_{ll}$ if and only if A is synchronizable (at all), although we are not necessarily in the position of producing a witness of this fact in polynomial time. But, as A is synchronizing but has no synchronizing word of length at most $|u|$, any synchronizing word is an extension of u, and we can freely choose (but not efficiently compute) the length-lexicographically smallest one. What remains is the case when $(A, |u| - 1)$ is a NO-instance of DFA-SW and $(A, |u|)$ is a YES-instance of DFA-SW. We should now check somehow all words w of length $|u|$ that are length-lexicographically smaller than u. If and only if none of them is a synchronizing word, then (A, u) is a YES-instance of Ext DFA-SW-$\leq_{ll}$. However, there could be exponentially (up to $|\Sigma|^k$) many of such strings w; we did not see a way to quickly check all of them. Hence, it remains open if this extension problem is as hard as DFA-SW in the parameterized sense.

Corollary 5.2. Ext DFA-SW-$\leq_{ll}$ *is co-NP-complete, even on binary input alphabets.*

Proof. The co-NP-hardness was shown in [20]. Membership in co-NP follows with the same arguments as the ones from the previous proof, yielding membership in co-WNL. □

For the parameter $|\Sigma|$, *i.e.*, input alphabet size, the situation is different. This is not explicitly stated in [20], but can be easily deduced from the discussions in that paper. Let us make this explicit in the following.

- Ext DFA-SW-$|$, parameterized by $|\Sigma|$, poses an *open* question.
- Ext DFA-SW-$\leq_{ll}$, parameterized by $|\Sigma|$, does not possess an XP-algorithm, unless P = NP.

When we restrict our attention again to TTSPL multi-graphs, the positive algorithmic results immediately transfer. So, the question is if the (co-)NP-hardness (or parameterized hardness) results still hold and if we might tackle some of the open questions in this more restricted setting.

Proposition 5.3. Ext DFA-SW-$\leq_{ll}$ *is co-NP-complete, even for TTSPL automata.*

Proof. Membership in co-NP immediately follows from the more general case stated in Cor. 5.2. For the co-NP-hardness, recall that the proof of the corresponding statement for general automata could be deduced from

the hardness proof presented in Theorem 5.1. This construction modifies a given DFA instance of DFA-SW by adding self-loops. Hence, it preserves the TTSPL property. Therefore, the claim follows with Theorem 3.2. □

As an aside, notice that Theorem 3.2 was not working for fixed input alphabet sizes, and so the previous proposition does not work for fixed input alphabets, either.

Proposition 5.4. Ext DFA-SW-|, *parameterized by* $|u|$, *is* W[3]*-hard, even for TTSPL automata.*

Proof. Bläser *et al.* [3] have shown that Ext Hitting Set, parameterized by the size of the given set U, is hard for W[3]. More formally, instances of this problem are formed by a family $\mathcal{F}$ of sets over a universe $\mathcal{U}$ and some set $U \subseteq \mathcal{U}$, and the question is if there exists a set S with $U \subseteq S \subseteq \mathcal{U}$ that is an inclusion-wise minimal hitting set. Theorem 27 in [20] converted an instance of Ext Hitting Set into an instance of Ext DFA-SW-|, parameterized by $|u|$. This construction did not produce a TTSPL automaton, but it can be modified to do so, in a similar way as in the construction from Proposition 4.4 resulting in an automaton A which would be clearly TTSPL. Moreover, from $U = \{e_{i_1}, \ldots, e_{i_{|U|}}\}$, we construct the word $u = e_{n+1}e_{i_1} \cdots e_{i_{|U|}}$. Finally, observe that the given instance of Ext Hitting Set is a YES-instance if and only if (A, u) is a YES-instance of Ext DFA-SW-|. In particular, the first letter e_{n+1} moves to the node corresponding to the first hyperedge, which must be covered by some $e \in U$ (as any other hyperedge) and hence u will lead from the source state q_s to the target state q_t. □

Unfortunately, again there is no bound on the size of the input alphabet. This is also true for the following result in classical complexity, which is obtained with the same reduction.

Corollary 5.5. Ext DFA-SW-| *is* NP*-hard, even for TTSPL automata.*

Let us conclude by remarking that the situation both of Ext DFA-SW-$\leq_{ll}$ and of Ext DFA-SW-|, parameterized by $|\Sigma|$, is *open* when restricted to TTSPL automata.

5.2. Minimum synchronizable sub-automata

In [47], the authors asked to extract a synchronizable sub-automaton that is as small as possible, obtained by deleting letters from its specification. This notion of a sub-automaton is of particular interest to us, as we are dealing with completely specified deterministic automata, and DFAs are not closed under most other notions of sub-automaton one might come up with, because they tend to produce incomplete automata. Türker and Yenegün formalized this idea as a weighted minimization problem. For our purposes, it is sufficient to consider the following unweighted variant:

DFA-MSS (referring to a minimum synchronizable sub-automaton)

Input: DFA A with input alphabet Σ, $k \in \mathbb{N}$

Problem: Is there a sub-alphabet $\hat{\Sigma} \subseteq \Sigma$, $|\hat{\Sigma}| \leq k$, such that the restriction of A to $\hat{\Sigma}$ is synchonizable?

Interestingly, Türker and Yenegün used nearly the same reduction as Fernau, Heggernes and Villanger in [19] for a different purpose to prove the following result.

Theorem 5.6. DFA-MSS *is* NP*-complete.*

In actual fact, in [47] only NP-hardness is proven, but a simple guess-and-check approach shows membership in NP, as well. Due to the simplicity of the reduction, inapproximability results follow, as already stated in [47], although even stronger results can be obtained by using more recent results from Dinur and Steurer [13]. However, in the context of our study, it is more interesting that we can also deduce the following parameterized complexity result from this reduction.

Corollary 5.7. DFA-MSS*, parameterized by the upper bound* k *on the size of the selected alphabet, is* W[2]*-hard.*

As with the other W[2]-hard problems considered in this paper, membership in W[2] is open. Also in this case, we can prove membership in WNL, although this is not completely trivial this time.

Theorem 5.8. DFA-MSS *is contained in WNL.*

Proof. Let (A, k) be an instance of DFA-MSS. Notice that in a first preprocessing step, we can eliminate letters a' that act the same on the state set Q as another letter a that we decide to keep. Such a rule can be implemented to run in polynomial time, as we simply loop over all pairs of letters, so that we can now assume to face an automaton A with input alphabet Σ and state set Q such that $|\Sigma| \leq |Q|^{|Q|}$. Moreover, we can assume that Σ contains more than $\log(|Q|)$ many symbols, as otherwise we can test all subsets of Σ (for synchronizability) in polynomial time. Finally, we can test in polynomial time if A itself is synchronizable at all [43, 49]. Recall that the algorithm checking synchronizability tests if any pair of states can be synchronized within at most $|Q|$ steps. We adapt this strategy in the following.

Now, we hard-wire the DFA into a Turing machine M as described in the following. In particular, this means that this Turing machine can keep track of pairs of states, say, (q, p) and update this information towards (q', p') in its finite memory (alternatively, on the tape, using state letters) upon reading a symbol $a \in \Sigma$, such that q' is reached from q (in the given DFA) and p' is reached from p upon reading a. We also assume a fixed linear ordering $<$ on the state set Q and moreover, we assume that $Q \cap \Sigma = \emptyset$. Let q_a and q_b be the smallest and second-to-smallest states in Q.

The Turing machine M first guesses a sub-alphabet Σ' by writing the corresponding k input letters on its tape. Moreover, it writes down q_a next to q_b. After reading (and memorizing) the current state pair (q, p), initially $(q, p) = (q_a, q_b)$, the machine (nondeterministically) reads one of the k guessed input letters on its tape and transfers to (q', p') in its internal memory. It continues doing so until it reaches a pair (r, r) for some $r \in Q$. If it reaches such a pair, it is verified that the pair (q, p) written on the tape can be synchronized. It continues by incrementing p on the tape (according to the linear order $<$), and then testing the synchronizability of the new pair of states on the tape with respect to the guessed sub-alphabet. M loops until the largest element of Q (with respect to $<$) is reached. Then, it would increment the first component q of the pair and set the second component of the pair of states to the smallest state larger than the first component's state; then, again the inner loop is entered. Finally, when reaching the largest state in the first component, the procedure terminates, this way verifying that the automaton is indeed synchronizable when restricted to the guessed alphabet.

Obviously, the machine M uses only $k + c$ space for some constant c, depending upon details of the implementation. This proves that DFA-MSS belongs to WNL. □

Although we found similar parameterized complexity results for DFA-MSS as for DFA-SW, we are not aware of further close links between both problems. In particular, it remains unclear how to solve one problem with the help of the other (respecting our choice of parameters). Recall that we have previously used the reduction leading to W[2]-hardness for DFA-MSS also in our studies, see Proposition 4.4. Hence, it is tempting to look at this problem on TTSPL automata multi-graphs. There is one technicality why we cannot simply take the construction from Proposition 4.4: there, quite a number of new symbols were added (from the start state), and we cannot do so, since we like to keep up the TTSPL property and aim at parameterized complexity results in the end. Therefore, we need to add some tree-like gadgets on top (to keep the alphabet extension small), as we detailed in the proof of Proposition 4.3. This means that we only need two additional symbols to maintain the TTSPL property.

To be clear, let us explicitly state the problem and the consequences.

TTSPL-DFA-MSS

Input: TTSPL DFA A with input alphabet Σ, $k \in \mathbb{N}$

Problem: Is there a sub-alphabet $\hat{\Sigma} \subseteq \Sigma$, $|\hat{\Sigma}| \leq k$, such that the restriction of A to $\hat{\Sigma}$ is TTSPL and synchonizable?

TABLE 1. Complexity results for DFA-SW ['ETH'/'SETH'/'PH' ⇔ assuming (Strong) Exponential-Time Hypothesis/Polynomial-time Hierarchy non-collapse].

Parameter(s)	(Parameterized) Complexity	Further Bounds
-	NP-complete	
k	in WNL, W[P] and A[2]; W[2]-hard	SETH: not in $\mathcal{O}((\lvert\Sigma\rvert-\varepsilon)^k)\forall\varepsilon>0$
$\lvert\Sigma\rvert$	NP-complete for $\lvert\Sigma\rvert=2$	
k, $\lvert\Sigma\rvert$	FPT with $\mathcal{O}^*(\lvert\Sigma\rvert^k)$; PH: no polyn. kernel	SETH: not in $\mathcal{O}((\lvert\Sigma\rvert-\varepsilon)^k)\forall\varepsilon>0$
$\lvert Q\rvert$	FPT with $\mathcal{O}^*(2^{\lvert Q\rvert})$; PH: no polyn. kernel	ETH: not in $\mathcal{O}^*(2^{o(\lvert Q\rvert)})$

Corollary 5.9. TTSPL-DFA-MSS, *parameterized by the upper bound k on the size of the selected alphabet, is W[2]-hard. Moreover, viewed as a classical decision problem, it is NP-complete.*

Clearly, we also inherit inapproximability results from the source problem HITTING SET.
Let us allow two side-notes here:

1. It might be interesting to combine the questions from this subsection with that of the previous one, which would lead to questions to extend a given sub-alphabet such that the resulting automaton becomes synchronizable. Again, as we have related this question (previously) to HITTING SET, it is quite straightforward to see that this decision problem is NP-hard, as well as W[3]-hard, when parameterized by the size of the given sub-alphabet.
2. It is not clear to us how difficult it is to find, given some DFA A, a sub-alphabet $\hat{\Sigma}$ such that the restriction of A to this sub-alphabet results in a complete automaton that is TTSPL. This might lead to other interesting problem variants.

6. CONCLUSIONS

We have considered the classical automaton problem DFA-SW restricted to quite constrained types of automata graphs; most constraints still lead to NP-complete decision problems. The reader might have wondered why we did not touch more usual constraints, like bounded treewidth (or other width parameters) or bounded outerplanarity (in the already specialized case of planar graphs). However, observe that both reductions that we started out with (Rystsov-Eppstein or HITTING SET) are already hard on outerplanar graphs and hence on graphs of bounded treewidth, which makes it NP-hard on basically any width measure for directed graphs, as well, see [21, 23].

As an overview, the currently known complexity results from this paper (and from [5], including definitions of A[2] and W[P]) can be found in Table 1. It should be noted that by the constructions presented before, some hardness results for the problem DFA-SW arise for the problem restricted to certain graph classes. By the structure of the graphs created in [15, 40] and [19], the following holds:

Proposition 6.1. *Apart from the kernel lower bound with respect to $|Q|$, the results presented in Table 1 also hold if we restrict the problem* DFA-SW *to bipartite, planar, outerplanar, f-outerplanar and TTSPL multi-graphs as well as multi-graphs of bounded treewidth.*

Please note that the parameterized complexity results do not change if the treewidth or the parameter f from the definition of f-outerplanarity are used as an additional parameter. The construction from [53] seem to destroy some of the nice graph properties we need to preserve, therefore we formulated one exception in the proposition. Further parameterized results for DFA-SW restricted to automata with a TTSPL transition graph can be found in Table 2.

Let us finally mention some open problems and suggestions for further research.

Table 2. Complexity results for DFA-SW on automata with TTSPL graphs.

Parameter(s)	(Parameterized) Complexity
Number of shortcuts per strand	NP-complete for values ≥ 3
Number of loop-free strands	NP-complete for values ≥ 1
Number of long strands	NP-complete for values ≥ 1

1. From the viewpoint of parameterized complexity, it might be very interesting to study problems that are as hard as DFA-SW, parameterized by an upper bound on the length of the shortest synchronizing word. This line of research has been continued in [5]. One of the problems proven to be equivalent to DFA-SW is Monoid Factorization (as defined above).
2. We have shown that the W[2]-hardness proof for the parameter k upper-bounding the length of the shortest synchronizing word carries over to TTSPL multi-graphs, but we do not know if this restricted synchronization problem is as hard as the unrestricted one mentioned in the previous item. Again, this might also form an interesting new complexity class, also hosting a variation of Monoid Factorization, as we have explained.
3. Generalizing the observations mentioned in the previous item further, one could also look at more general (but not completely general) scenarios for automata multi-graphs, for instance, the general situation of being ordered (also sometimes called weakly acyclic), *i.e.*, assuming that the automaton graph is acyclic after removing all loops. In terms of transition monoids, this restriction is tightly linked to R-trivial monoids, see [6]. So, we might even find some connections to basic monoid theory, as exemplified by Green's relation. This might give a particular algebraic taste to the corresponding restrictions of Monoid Factorization.
4. Based on the inductive definition of TTSPL, we can observe: roughly speaking, if automaton A is produced by a series or parallel composition of automata A_1 and A_2 with synchronizing words w_1 and w_2, then w_1w_2 is a synchronizing word for A.[5] Can such ideas lead to good approximations of synchronizing words?
5. What happens if we consider the number of strands or the number of series or the number of parallel compositions needed to construct the automaton graph as a parameter? Observe that on the one hand the gadget graphs presented in this paper have relatively large parameter values for each of these parameters, but on the other hand the problem on graphs consisting only of parallel compositions remains solvable in constant time.
6. It is unclear if the parameterization with the number of loops leads to better results. This also applies to the parameterization counting nodes with loops.
7. We also discussed several extension variants; in particular, with respect to the length-lexicographical ordering, we might find another problem candidate that finds its home in the parameterized complexity class suggested in the first item, but also this is not completely clear, even if we restrict our attention to TTSPL automata.
8. Also for the extension variants, the parameter "input alphabet size" is not well understood and remains to be studied.
9. In the case of DFA-MSS, we can likewise ask about membership in W[2], even when restricted to the setting of TTSPL-DFA-MSS. As discussed above, the (parameterized) reducibility to DFA-SW is unclear.

Acknowledgements. We are grateful to Stefan Hoffmann for comments on the very first versions of this paper.

[5] One has to be careful with loops and completeness conditions; we skip the corresponding technical details.

References

[1] M. Béal, M.V. Berlinkov and D. Perrin, A quadratic upper bound on the size of a synchronizing word in one-cluster automata. *Int. J. Found. Comp. Sci.* **22** (2011) 277–288.

[2] M.V. Berlinkov, Approximating the minimum length of synchronizing words is hard. *Theory Comput. Syst.* **54** (2014) 211–223.

[3] T. Bläsius, T. Friedrich, J. Lischeid, K. Meeks and M. Schirneck, Efficiently enumerating hitting sets of hypergraphs arising in data profiling, in *Algorithm Engineering and Experiments (ALENEX)*. SIAM (2019) 130–143.

[4] H. Bodlaender, R.G. Downey, M.R. Fellows and H.T. Wareham, The parameterized complexity of sequence alignment and consensus. *Theor. Comput. Sci.* **147** (1995) 31–54.

[5] J. Bruchertseifer and H. Fernau, Synchronizing words and monoid factorization: a parameterized perspective, in *Theory and Applications of Models of Computation, 16th International Conference, TAMC*, edited by J. Chen, Q. Feng and J. Xu, eds., vol. 12337 of *LNCS*. Springer (2020) 352–364.

[6] J.A. Brzozowski and F.E. Fich, Languages of $\mathcal{R}$-trivial monoids. *J. Comput. Syst. Sci.* **20** (1980) 32–49.

[7] L. Cai, J. Chen, R. Downey and M. Fellows, On the parameterized complexity of short computation and factorization. *Arch. Math. Logic* **36** (1997) 321–337.

[8] J. Černý, A. Pirická and B. Rosenauerová, On directable automata. *Kybernetika* **7** (1971) 289–298.

[9] J. Černý, Poznámka k homogénnym experimentom s konečnými automatmi. *Matematicko-fyzikálny časopis* **14** (1964) 208–216.

[10] J. Černý, A note on homogeneous experiments with finite automata. *J. Autom. Lang. Combinat.* **24** (2019) 123–132.

[11] H. Cho, S. Jeong, F. Somenzi and C. Pixley, Synchronizing sequences and symbolic traversal techniques in test generation. *J. Electr. Testing* **4** (1993) 19–31.

[12] M. Cygan, F. Fomin, L. Kowalik, D. Lokshtanov, D. Marx, M. Pilipczuk, M. Pilipczuk and S. Saurabh, Parameterized Algorithms. Springer (2015).

[13] I. Dinur and D. Steurer, Analytical approach to parallel repetition, in *Symposium on Theory of Computing, STOC*, edited by D.B. Shmoys. ACM (2014) 624–633.

[14] R.G. Downey and M.R. Fellows, Fundamentals of Parameterized Complexity. Texts in Computer Science. Springer (2013).

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

[16] D. Eppstein, Parallel recognition of series-parallel graphs. *Inf. Comput.* **98** (1992) 41–55.

[17] H. Fernau, Modern aspects of complexity within formal languages, in *Language and Automata Theory and Applications - 13th International Conference, LATA*, edited by C. Martín-Vide, A. Okhotin and D. Shapira. Vol. 11417 of *LNCS*. Springer (2019) 3–30.

[18] H. Fernau, V.V. Gusev, S. Hoffmann, M. Holzer, M.V. Volkov and P. Wolf, Computational complexity of synchronization under regular constraints, in *44th International Symposium on Mathematical Foundations of Computer Science (MFCS 2019)*, edited by P. Rossmanith, P. Heggernes and J.-P. Katoen. Vol. 138 of *Leibniz International Proceedings in Informatics (LIPIcs)*. Schloss Dagstuhl–Leibniz-Zentrum für Informatik (2019) 63:1–63:14.

[19] H. Fernau, P. Heggernes and Y. Villanger, A multi-parameter analysis of hard problems on deterministic finite automata. *J. Comput. Syst. Sci.* **81** (2015) 747–765.

[20] H. Fernau and S. Hoffmann, Extensions to minimal synchronizing words. *J. Autom. Lang. Combin.* **24** (2019) 287–307.

[21] H. Fernau and D. Meister, Digraphs of bounded elimination width. *Discr. Appl. Math.* **168** (2014) 78–87.

[22] P. Frankl, An extremal problem for two families of sets. *Eur. J. Combin.* **3** (1982) 125–127.

[23] R. Ganian, P. Hliněný, J. Kneis, D. Meister, J. Obdrzálek, P. Rossmanith and S. Sikdar, Are there any good digraph width measures? *J. Combin. Theory B* **116** (2016) 250–286.

[24] W. Göhring, Minimal initializing word: a contribution to Černý's conjecture. *J. Autom. Lang. Combin.* **2** (1997) 209–226.

[25] S. Guillemot, Parameterized complexity and approximability of the longest compatible sequence problem. *Discr. Optim.* **8** (2011) 50–60.

[26] S. Gulan, On the Relative Descriptional Complexity of Regular Expressions and Finite Automata, PhD thesis, Fachbereich IV, Universität Trier, Germany (2011).

[27] S. Gulan, Series parallel digraphs with loops—graphs encoded by regular expression. *Theory Comput. Syst.* **53** (2013) 126–158.

[28] F. Gurski and C. Rehs, Comparing linear width parameters for directed graphs. *Theory Comput. Syst.* **63** (2019) 1358–1387.

[29] J. Kari, Synchronizing finite automata on Eulerian digraphs. *Theor. Comput. Sci.* **295** (2003) 223–232.

[30] A. Kisielewicz, J. Kowalski and M. Szykuła, Computing the shortest reset words of synchronizing automata. *J. Combin. Optim.* **29** (2015) 88–124.

[31] S. Kreutzer and O. Kwon, Digraphs of bounded width, in *Classes of Directed Graphs, Springer Monographs in Mathematics*. Springer (2018) 405–466.

[32] P. Martyugin, Complexity of problems concerning reset words for some partial cases of automata. *Acta Cybern.* **19** (2009) 517–536.

[33] P.V. Martyugin, Complexity of problems concerning reset words for cyclic and Eulerian automata. *Theor. Comp. Sci.* **450** (2012) 3–9.

[34] P.V. Martyugin, Computational complexity of certain problems related to carefully synchronizing words for partial automata and directing words for nondeterministic automata. *Theory Comput. Syst.* **54** (2014) 293–304.

[35] R.H. Möhring, Computationally tractable classes of ordered sets, in *Algorithms and Order: Proceedings of the NATO Advanced Study Institute on Algorithms and Order*, edited by I. Rival. Vol. 255 of *NATO Science Series C*. Springer (1989) 105–194.

[36] J.A. Montoya and C. Nolasco, On the synchronization of planar automata, in *Language and Automata Theory and Applications – 12th International Conference, LATA*, edited by S.T. Klein, C. Martín-Vide and D. Shapira. Vol. 10792 of *LNCS*. Springer (2018) 93–104.

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

[38] E.V. Pribavkina and E. Rodaro, Synchronizing automata with finitely many minimal synchronizing words. *Inf. Comput.* **209** (2011) 568–579.

[39] J. Rho, F. Somenzi and C. Pixley, Minimum length synchronizing sequences of finite state machine, in *Proceedings of the 30th Design Automation Conference, DAC*, edited by A.E. Dunlop. ACM Press (1993) 463–468.

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

[41] I.K. Rystsov, Reset words for commutative and solvable automata. *Theor. Comput. Sci.* **172** (1997) 273–279.

[42] A. Ryzhikov, Synchronization problems in automata without non-trivial cycles. *Theor. Comput. Sci.* **787** (2019) 77–88.

[43] S. Sandberg, Homing and synchronizing sequences, in *Model-Based Testing of Reactive Systems*, edited by M. Broy, B. Jonsson, J.-P. Katoen, M. Leucker, and A. Pretschner. Vol. 3472 of *LNCS*. Springer (2005) 5–33.

[44] Y. Shitov, An improvement to a recent upper bound for synchronizing words of finite automata. *J. Autom. Lang. Combinat.* **24** (2019) 367–373.

[45] B. Steinberg, The Černý conjecture for one-cluster automata with prime length cycle. *Theor. Comput. Sci.* **412** (2011) 5487–5491.

[46] M. Szykuła, Improving the upper bound on the length of the shortest reset word, in *35th Symposium on Theoretical Aspects of Computer Science, STACS*, edited by R. Niedermeier and B. Vallée. Vol. 96 of *LIPIcs*. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2018) 1–13.

[47] U.C. Türker and H. Yenigün, Complexities of some problems related to synchronizing, non-synchronizing and monotonic automata. *Int. J. Found. Comput. Sci.* **26** (2015) 99–122.

[48] J. Valdes, R.E. Tarjan and E.L. Lawler, The recognition of series parallel graphs. *SIAM J. Comput.* **11** (1982) 298–313.

[49] M.V. Volkov, Synchronizing automata and the Černý conjecture, in *Language and Automata Theory and Applications, Second International Conference, LATA*, edited by C. Martín-Vide, F. Otto and H. Fernau. Vol. 5196 of *LNCS*. Springer (2008) 11–27.

[50] M.V. Volkov, Synchronizing automata preserving a chain of partial orders. *Theor. Comput. Sci.* **410** (2009) 3513–3519.

[51] M.V. Volkov, Preface: Special issue on the Černý conjecture. *J. Autom. Lang. Comb.* **24** (2019) 119–121.

[52] V. Vorel, Complexity of a problem concerning reset words for Eulerian binary automata. *Inf. Comput.* **253** (2017) 497–509.

[53] V. Vorel and A. Roman, Parameterized complexity of synchronization and road coloring. *Discr. Math. Theor. Comput. Sci.* **17** (2015) 283–306.

[54] H.T. Wareham, The parameterized complexity of intersection and composition operations on sets of finite-state automata, in *Implementation and Application of Automata, 5th CIAA 2000*, edited by S. Yu and A. Păun. Vol. 2088 of *LNCS*. Springer (2001) 302–310.

[55] C. Yap, Some consequences of non-uniform conditions on uniform classes. *Theor. Comput. Sci.* **26** (1983) 287–300.