Synchronization and Diversity of Solutions

Emmanuel Arrighi[1], **Henning Fernau**[2], **Mateus de Oliveira Oliveira**[1,3], **Petra Wolf** [1,2]

[1] University of Bergen, Bergen, Norway
[2] University of Trier, Trier, Germany
[3] Stockholm University, Stockholm, Sweden
emmanuel.arrighi@uib.no, oliveira@dsv.su.se, mail@wolfp.net, fernau@uni-trier.de

Abstract

A central computational problem in the realm of automata theory is the problem of determining whether a finite automaton A has a synchronizing word. This problem has found applications in a variety of subfields of artificial intelligence, including planning, robotics, and multi-agent systems. In this work, we study this problem within the framework of diversity of solutions, an up-and-coming trend in the field of artificial intelligence where the goal is to compute a set of solutions that are sufficiently distinct from one another.

We define a notion of diversity of solutions that is suitable for contexts were solutions are strings that may have distinct lengths. Using our notion of diversity, we show that for each fixed $r \in \mathbb{N}$, each fixed finite automaton A, and each finite automaton B given at the input, the problem of determining the existence of a diverse set $\{w_1, w_2, \ldots, w_r\} \subseteq L(B)$ of words that are synchronizing for A can be solved in polynomial time. Finally, we generalize this result to the realm of conformant planning, where the goal is to devise plans that achieve a goal irrespectively of initial conditions and of nondeterminism that may occur during their execution.

1 Introduction

A word w is said to be synchronizing for a deterministic finite automaton (DFA) A if there is some state q of A such that any state q' is sent to q by w. This concept has found numerous applications across several subfields of computer science and artificial intelligence, such as circuit testing (Hennie 1964; Kohavi and Jha 2009; Sandberg 2005), multi-agent systems (Chapman and Mesbahi 2014; Chevalier, Hendrickx, and Jungers 2015), robotics (Natarajan 1986), game theory (Maubert 2009), among others.

The most central problem in the field of synchronization is the one to determine whether a given DFA has a synchronizing word. Note that this problem can be decided in polynomial time (Starke 1966). Nevertheless, in several applications, one is interested in finding a synchronizing word satisfying certain additional constraints (Fernau et al. 2019; Wolf 2020; Türker and Yenigün 2015). Here, the complexity landscape of the problem changes drastically: even the problem of determining the existence of a synchronizing word satisfying certain additional regularity constraints is NP-hard. For instance, it is NP-hard to determine whether a given DFA A has a synchronizing word that belongs to the regular language ab^*a (Fernau et al. 2019), or whose length is bounded by a given integer (Rystsov 1980; Eppstein 1990).

Diversity of solutions is a trend that has been calling substantial attention of the artificial intelligence community during the past years (Hebrard et al. 2005; Baste et al. 2019; Petit and Trapp 2019; Baste et al. 2020; Fomin et al. 2020; Ingmar et al. 2020; Arrighi et al. 2021). The goal is to find not a single solution to a given combinatorial problem, but rather a *small* set of solutions being sufficiently diverse from each other. One of the motivations for this framework is that it can be applied in situations where certain side constraints are difficult, or even impossible to formalize. In this case, the user can select a solution that she deems to be best for her application at hand. Intuitively, the diversity requirement tells us that the solutions that are given as an output are a reasonable representative of the space of solutions.[1] Also, small sets of solutions make sense because one does not want to overwhelm the user with an excessive number of solutions.

While for many combinatorial problems, notions of solution diversity based on the Hamming distance between pairs of solutions are sufficient, in the context of synchronization, these notions are not so appropriate. First, distinct solutions may have distinct length, and it is not clear how positions should be aligned in order to define an appropriate notion of Hamming distance. Additionally, even strings of the same size that are very similar to each other may have very large Hamming distance. For instance, the Hamming distance between the string $w = abab...ab = (ab)^n$ and the string $w' = baba...ba = (ba)^n$ is $2n$, while w can be transformed into w' with two modifications: first delete the first symbol of w and then append the symbol a to the resulting string.

We circumvent this issue by basing our diversity measure on the notion of string edit distance (Wagner and Fischer 1974; Levenshtein 1966). This is a well studied metric for strings with nice properties and applications in a wide variety of fields (Bunke and Csirik 1995; Lyras, Sgarbas, and Fakotakis 2007; Chua, Marsland, and Guesgen 2011).

Another issue is that sets of solutions in which any two of them are far apart from each other may still not capture solu-

[1]See (Sayin 2000; Vaz et al. 2015; Demirovic and Schwind 2020) for further discussions.

tion diversity in the context of synchronization. The problem is that if w is a synchronizing word, then any superword of w is also synchronizing. Therefore, any sequence of superwords $w_1, w_2, \ldots, w_k$ of w of substantially distinct lengths would have large diversity if only edit distance were taken into consideration. To circumvent this issue, we require that each word in the set of solutions is subsequence-minimal with respect to the synchronization requirement. The subsequence minimality requirement combined with edit distance not only guarantees that solutions in any given subset are genuinely distinct, but also provides a way of tackling diverse synchronization problems using the machinery of finite automata theory. On the one hand, Higman's lemma (Higman 1952) (see Lemma 6), a classical tool in automata theory, implies that the set of subsequence-minimal synchronizing words in the language of an automaton is always finite. On the other hand, the computation of the edit distance between two words is a process that can be simulated using finite automata. More specifically, it is possible to construct finite automata accepting a suitable encoding of pairs of words that are far apart from each other.

Note that subsequence-minimal synchronization problems involving a single DFA A are already hard. First, subsequence-minimal synchronizing words for a DFA A may have exponential length on the number of states of A (Proposition 20). Second, determining if a given word w is subsequence-minimal among all synchronizing words in the language of a DFA A is coNP-hard (Theorem 19). Third, determining if a DFA A has two distinct subsequence-minimal synchronizing words is NP-hard (Theorem 23). Finally, the problem of enumerating the set of subsequence-minimal synchronizing words is $\#\mathsf{P}$-hard (Theorem 24).

In order to cope with the inherent intractability of synchronization problems, we leverage on the framework of parameterized complexity theory (Downey and Fellows 1999). In particular, we show that for each fixed value of r, interesting computational problems requiring a diverse set with r subsequence-minimal synchronizing words can be solved in time that is fixed-parameter tractable with respect to the size of the synchronizing automaton A. Previously, algorithms with an FPT dependence in $|A|$ were unknown even for $r = 2$! Using our approach, we also show that given a DFA A with state set Q over an alphabet Σ and a word $w \in \Sigma^*$, one can determine in time $O(f(|\Sigma|, |Q|) \cdot |w|)$, for some function f, if some subsequence-minimal synchronizing word for A is a subsequence of w, and we can construct such a subsequence in case the answer is affirmative (Theorem 15). As mentioned above, the unparameterized version of this problems is already coNP-hard. Our main result (Theorem 16) states that given numbers $r, k \in \mathbb{N}$, a DFA A, and an possibly nondeterministic finite automaton B over an alphabet Σ, the problem of computing a subset $\{w_1, \ldots, w_r\} \subseteq L(B)$ of subsequence-minimal synchronizing words for A, with pairwise edit distance of at least k, can be solved in time $O(f_A(r,k) \cdot |B|^r \log(|B|))$ for some suitable function f depending only on A, r and k. Intuitively, the automaton A is a specification of a system which we want to synchronize (or reset), and B is a specification of the set of words that are allowed to be used as synchronizing sequences. As stated in the beginning of this section, the unparameterized version of this problem is NP-hard even if we are interested in finding a single solution and the language of the automaton B is as simple as ab^*a. As a consequence of our main result, given a word $w \in \Sigma^*$, the problem of determining whether there exist r subsequence-minimal synchronizing words for A that are subsequences of w and that are at least k apart from each other can be solved in time $O(f_A(r,k) \cdot |w|^r \log(|w|))$ (Corollary 17).

It turns out that our notion of diversity of solutions is general enough to be applied in other contexts where solutions are strings whose sizes may have to vary. In particular, we generalize our framework to the realm of conformant planning (Theorem 28), where the goal is to design plans that achieve goals irrespectively of initial conditions and of nondeterminism that may occur during the execution of these plans (Anders 2019; Bonet 2010; Cimatti, Roveri, and Bertoli 2004; Palacios and Geffner 2009).

Further Applications. In the full version of this work (see https://doi.org/10.5281/zenodo.7384348), we discuss applications of our main results to several subdomains of artificial intelligence, such as multi-agent systems, robotics, secret sharing, and others. The intuition is that synchronizing words may be viewed as a way of resetting agents whose behavior are modeled by finite automata, while diversity may be interpreted as a way of achieving resilience (if one synchronizing word fails, we may still use another one), distributivity (multiple synchronizing words may be distributed among distinct parties), or security (multiple synchronizing words may be used to define secret-sharing protocols).

2 Preliminaries

Let $\mathbb{N}$ denote the set of non-negative integers, while $\mathbb{N}_{>0}$ is the set of positive integers. For an integer $k \in \mathbb{N}$, we denote by $[k]$ the set $\{1, 2, \ldots, k\}$. Hence, $[0] = \emptyset$. For $n \in \mathbb{N}$, Σ^n denotes the set of all words of length n. We denote $\Sigma^+ = \bigcup_{n \in \mathbb{N}_{>0}} \Sigma^n$ and $\Sigma^* = \Sigma^+ \cup \{\varepsilon\}$, where ε denotes the *empty word*. Hence, Σ^* is the groundset of the free monoid generated by Σ; its binary operation is known as *concatenation*, often denoted by juxtaposition, but sometimes highlighted by explicitly writing $\cdot$. Given a word $w \in \Sigma^*$, we let $|w|$ denote the *length* of w. For each $i \in 1, \ldots, |w|$, we let $w[i]$ denote the i-th symbol of w. For $i, j \in 1, \ldots, |w|$, we let $w[i..j]$ denote the *infix* $w[i]w[i+1] \ldots w[j]$.

A *deterministic finite automaton* (DFA) A is a tuple $A = (Q, \Sigma, \delta, q_0, F)$, where Q is a finite set of *states*, Σ is a finite *alphabet*, $\delta \colon Q \times \Sigma \to Q$ is a total function called the *transition function* of A, $q_0 \in Q$ is the *start state*, and $F \subseteq Q$ is a set of *final states*. For synchronization problems, we sometimes omit start and final states of DFAs. When speaking about the complexity of algorithms involving a DFA A, we sometimes use $|A|$ to denote the number of bits needed to specify A; for convenience, let $|A| = |Q|\,|\Sigma|\,\log(|Q|)$ (for DFAs). Often, $|Q|$ is sufficient as a size estimate. We generalize δ to words by setting $\delta(q, \varepsilon) = q$ and $\delta(q, w) = \delta(\delta(q, w[1]), w[2..|w|])$. We further generalize δ to sets of states $S \subseteq Q$ and sets of input letters $\Gamma \subseteq \Sigma$

as $\delta(S,\Gamma) = \{\delta(s,\gamma) \mid s \in S, \gamma \in \Gamma\}$, or to words w as $\delta(S,w) = \{\delta(s,w) \mid s \in S\}$. We say a word $w \in \Sigma^*$ is a *synchronizing word* for A if $|\delta(Q,w)| = 1$.

We assume basic knowledge of automata theory on side of the reader. In particular, the product automata construction for finite automata should be known. Also, we make use of nondeterministic finite automata (or NFA for short) without a precise formal definition. Recall that with NFAs, the transition function is replaced by a transition relation δ. Due to the well-known powerset automaton construction, we can (still) view δ as mapping (S,w) to T, where S,T are sets of states, and w is a word over the input alphabet.

We will discuss several partial orderings on Σ^* now. See (Birget 1993) for a review of these and other concepts. In particular, x is a *prefix* of y, written $x \sqsubseteq y$, if $y \in x\Sigma^*$, x is a *suffix* of y if $y \in \Sigma^* x$ and x is an *infix* of y, written $x \preceq y$, if $y \in \Sigma^* x \Sigma^*$. We say that x is a *subsequence* of y, written $x \leq y$,[2] if there exists a sequence of indices $i_1 < i_2 < ... < i_{|x|}$ such that for each $j \in [|x|]$, $y_{i_j} = x_j$. If we add the word *proper*, we exclude the possibility that $x = y$ in each of the previous definitions. In our notations, we then write $\sqsubset$, $\prec$, or $<$, respectively. A non-empty word x can be split into the prefix $x[1]$ of length 1 and the suffix of length $|x|-1$ that we call (reminiscent of Prolog) the *tail* of x, denoting it by $\mathrm{tail}(x)$. Hence, $x = x[1]\mathrm{tail}(x)$.

Let Σ be some alphabet not containing the symbol $\Box$ that we will now use as a blank symbol. For $k \in \mathbb{N}_{>0}$, define $(\Sigma \cup \{\Box\})^{\times k} = (\Sigma \cup \{\Box\})^k \setminus \{(\Box,\dots,\Box)\}$ as a new alphabet with $(|\Sigma|+1)^k - 1$ many symbols that we also call *compound characters*, consisting of k letters. Of particular importance will be the case $k = 2$. For this, we now define a specific construction, called *convolution*, that takes two words $u, v \in \Sigma^*$ and constructs a unique word $w = u \otimes v$ over the alphabet $(\Sigma \cup \{\Box\})^{\times 2}$, in a recursive fashion as follows.

$$u \otimes v = \begin{cases} \varepsilon, & \text{if } u = v = \varepsilon \\ (u[1],\Box) \cdot (\mathrm{tail}(u) \otimes \varepsilon), & \text{if } u \neq \varepsilon, v = \varepsilon \\ (\Box, v[1]) \cdot (\varepsilon \otimes \mathrm{tail}(v)), & \text{if } u = \varepsilon, v \neq \varepsilon \\ (u[1], v[1]) \cdot (\mathrm{tail}(u) \otimes \mathrm{tail}(v)), & \text{if } u \neq \varepsilon, v \neq \varepsilon \end{cases}$$

For instance, the convolution of word $u = ababa$ with $v = abb$ is the word $u \otimes v = (a,a)(b,b)(a,b)(b,\Box)(a,\Box)$. This operation can be generalized to the convolution of $k > 2$ words in a straightforward manner.

Lemma 1. *The language of all words over the alphabet $(\Sigma \cup \{\Box\})^{\times k}$ that can be obtained as the convolution of k words over Σ can be accepted by a DFA with 2^k many states.*

The previous construction can be integrated into the classical product automaton construction to give the following result.

Corollary 2. *Let $A = (Q,\Sigma,\delta,q_0,F)$ be a DFA. Let A^k be a DFA accepting the set of all strings of the form $u_1 \otimes u_2 \otimes \cdots \otimes u_k$, where for each $l \in [k]$, $u_l \in L(A)$. Then, A^k needs to have no more than $(2|Q|)^k$ many states. A similar statement holds for NFAs.*

[2]In the literature, the naming of these relations is rather confusing. Infixes are known as factors or subwords, subsequences are also called (scattered) subwords.

3 Diversity in Synchronization

We will base our notion of diversity on the notion of edit distance between strings, as it provides a suitable way of measuring dissimilarity between strings with possibly distinct lengths. Let $u = u[1]u[2]\dots u[n]$ be a string in Σ. We define the following elementary edit operations on u; see (Levenshtein 1966; Wagner and Fischer 1974).

1. Insertion: for $i \in \{0,\dots,n\}$, insert letter $a \in \Sigma$ after position i, yielding $u[1..i]\,a\,u[i+1..n]$.
2. Deletion: for $i \in \{1,\dots,n\}$, delete the entry $u[i]$ from u, resulting in $u[1..i-1]\,u[i+1..n]$.
3. Replacement: for $i \in \{1,\dots,n\}$, replace $u[i]$ with some $a \in \Sigma \setminus \{u[i]\}$, getting $u[1..i-1]\,a\,u[i+1..n]$.

We say that a string $u \in \Sigma^*$ can be edited to a string $w \in \Sigma^*$ in s steps if there is some sequence $u_0, u_1, \dots, u_s$ of strings in Σ^* such that for each $j \in [r]$, u_j is obtained from u_{j-1} by the application of one of the three elementary edit operations above and $u = u_0$, $w = u_s$.

Definition 3 (Edit Distance)**.** *The edit distance between u and w, denoted by $\Delta(u,w)$, is defined as the minimum s such that u can be edited to w in s steps.*

Intuitively, the edit distance between u and w is the minimum number of insertions, deletions and replacements necessary to transform u into w. By classical results, this distance (and related ones) can be computed efficiently; see (Wagner and Fischer 1974; Wagner 1975).

As discussed in the introduction, each word containing a synchronizing word as an infix is also synchronizing. Hence, in order to get a diverse set of synchronizing words, we need to demand minimality with respect to the subsequence order.[3] A word w is a *subsequence-minimal synchronizing word* for a DFA A if w is synchronizing for A and no proper subsequence w' of w is synchronizing for A. On the one hand, one can ensure that the set of subsequence-minimal synchronizing words for a DFA is finite (see Sec. 4). On the other hand, subsequence minimality imposes a higher level of dissimilarity between two words in a prospective subset W of solutions, increasing the representativeness of W.

Definition 4 (Synchronization Diversity)**.** *Let $A = (Q,\Sigma,\delta)$ be a DFA. We say that a set of words $\{w_1,\dots,w_r\}$ has* synchronization diversity k *if the following holds.*

1. *For each $i, j \in \{1,...,r\}$ with $i \neq j$, $\Delta(w_i,w_j) \geq k$.*
2. *For each $i \in \{1,...,r\}$, w_i is a subsequence-minimal synchronizing word for A.*

4 Subsequence Minimality

The next well-known lemma, whose proof can be found in (Gruber, Holzer, and Kutrib 2007, Corollary 18), states that, given an NFA A, one can construct a finite automaton $\mathrm{Subseq}(A)$ accepting precisely the subsequences of words in $L(A)$. As shown in (Gruber, Holzer, and Kutrib 2007, Lemma 19), this bound cannot be improved in general.

[3]One is tempted to require infix minimality instead, but as the infix order is not a well-ordering, the set of infix-minimal synchronizing words of an automaton can be infinite, which make it difficult to find good representatives for the space of solutions.

Lemma 5 (Subsequence Automaton)**.** *Let $A = (Q, \Sigma, \delta)$ be an NFA. One can construct in time $O(|A|)$ an NFA* $\mathrm{Subseq}(A)$ *with $|Q|$ states with $L(\mathrm{Subseq}(A)) = \{u : \exists w \in L(A),\ u \leq w\}$. If A has a trap state,* $\mathrm{Subseq}(A)$ *has $|Q| - 1$ states.*

A classic result in partial order theory, known as Higman's Lemma, states that for each finite set Σ, the set Σ^* of finite words over Σ is well-ordered under the subsequence order (Higman 1952). An interesting consequence of this lemma is the fact that for each language $L \subseteq \Sigma^*$, the set of subsequence-minimal words in L is finite. For a more language-theoretic treatment, refer to (Haines 1969).

Lemma 6 (Higman's Lemma)**.** *Let Σ be an alphabet and $L \subseteq \Sigma^*$. There is a unique finite set $S \subseteq L$ satisfying the following properties.*

1. *For each word $w \in L$, some word $u \in S$ is a subsequence of w.*
2. *Each word $u \in S$ is subsequence-minimal for L.*

It is worth noting that Lemma 6 holds with respect to any language $L \subseteq \Sigma^*$, irrespectively of whether L is regular or not. Nevertheless, if we are given a DFA A accepting L, then we can construct a DFA $\mathrm{MinSubseq}(A)$ accepting the set of subsequence-minimal words for L.

Lemma 7 (Subsequence-Minimal Words, Theorem 2.1(3) of (Birget 1993))**.** *Let $A = (Q, \Sigma, \delta)$ be a DFA. One can construct in time $O(2^{|Q|} \cdot |A|)$ a DFA* $\mathrm{MinSubseq}(A)$ *with at most $|Q| \cdot 2^{|Q|}$ many states accepting the language*

$$L(\mathrm{MinSubseq}(A)) = \{u : u \in L(A) \wedge \forall w \in L(A), w \not< u\}.$$

Interestingly, the exponential blow-up in the number of states of the input automaton A is unavoidable, even if one allows $\mathrm{MinSubset}(A)$ to be an NFA. Here, we refer to Example 3.18, Facts 3.19 and 3.20 in (Birget 1993).

The next lemma states that given an automaton A, one can construct a DFA $\mathrm{Sync}(A)$ accepting precisely the words in Σ^* that are synchronizing for A. This is shown by using the classical subset construction, now on a DFA A; see (Sandberg 2005; Volkov 2008).

Lemma 8 (Synchronizing Words)**.** *Let $A = (Q, \Sigma, \delta)$ be a DFA. One can construct in time $O(2^{|Q|} \cdot |A|)$ a DFA* $\mathrm{Sync}(A)$ *with at most $2^{|Q|}$ states such that*

$$L(\mathrm{Sync}(A)) = \{u : u \text{ is synchronizing for } A\}.$$

This exponential blow-up is necessary, as recently proven in (Hoffmann 2021). By combining Lemma 8 with Lemma 7, we have the following corollary stating that, given a DFA A, one can construct a DFA $\mathrm{SyncMinSubseq}(A)$ of state complexity at most double-exponential in the state complexity of A accepting the set of subsequence-minimal synchronizing words for A. It is an interesting open question if this double-exponential blow-up is necessary.

Corollary 9. *Let $A = (Q, \Sigma, \delta)$ be a DFA. One can construct in time $O(2^{2^{|Q|}} \cdot 2^{|A|})$ a DFA* $\mathrm{SyncMinSubseq}(A)$ *with at most $2^{2^{|Q|} + |Q|}$ many states accepting the language*

$$L(\mathrm{SyncMinSubseq}(A)) = \{u : u \text{ is a subsequence-minimal synchronizing word for } A\}.$$

5 Edit Distance vs Diversity of Solutions

In (Wagner and Fischer 1974), a dynamic-programming algorithm is given that efficiently computes the edit distance between two given strings. Edit distance questions are an active area of study, as exemplified by (Bringmann et al. 2019). Ignoring technicalities, the next lemma can be seen as an automata-theoretic counterpart of such computations.

Lemma 10. *Let Σ be an alphabet and $k \in \mathbb{N}_{>0}$. There is an NFA* $\mathrm{Edit}^{<}(\Sigma, k)$ *with $2^{O(k \log |\Sigma|)}$ many states that accepts*

$$L(\mathrm{Edit}^{<}(\Sigma, k)) = \{u \otimes w : \Delta(u, w) < k\}. \qquad (1)$$

As a corollary of Lemma 10, by first determinizing the automaton of the previous lemma and then (basically) complementing final states, plus checking (by a product automaton construction) that words come from convolutions of words, using Lemma 1, we get the following result.

Lemma 11. *Let Σ be an alphabet and $k \in \mathbb{N}_{>0}$. There is a DFA* $\mathrm{Edit}^{\geq}(\Sigma, k)$ *with $2^{2^{O(k \log |\Sigma|)}}$ states accepting*

$$L(\mathrm{Edit}^{\geq}(\Sigma, k)) = \{u \otimes w : \Delta(u, w) \geq k\}. \qquad (2)$$

Let $W \subseteq \Sigma^*$ be a finite set of strings. The *min-diversity* of W, denoted by $\mathrm{MinDiv}(W)$ is defined as the minimum edit distance among any pair of strings in W.

$$\mathrm{MinDiv}(W) = \min_{u,w \in W} \Delta(u, w) \qquad (3)$$

Next, we state that given DFA A, one can construct a DFA $\mathrm{MinDiv}(A, r, k)$ whose language is a suitable encoding of all r-tuples of words from $L(A)$ with diversity at least k.

Lemma 12. *Let $A = (Q, \Sigma, \delta, q_0, F)$ be DFA over Σ. For each $r, k \in \mathbb{N}^+$, one can construct a DFA* $\mathrm{MinDiv}(A, r, k)$ *with $2^{r^2 \cdot 2^{O(k \cdot \log |\Sigma|)}} \cdot |Q|^r$ many states for the language of all compound words $u_1 \otimes \cdots \otimes u_r \in (\Sigma \cup \{\square\})^{\times r}$ that satisfy*

$$\forall i \in [r](u_i \in L(A)) \wedge \mathrm{MinDiv}(\{u_1, \ldots, u_r\}) \geq k.$$

Its construction takes $O\left(2^{r^2 \cdot 2^{O(k \log |\Sigma|)}} \cdot |A|^r\right)$ time.

Proof. Let $\binom{[r]}{2} = \{\{i, j\} : i, j \in [r], i \neq j\}$. For each pair $\{i, j\} \in \binom{[r]}{2}$, let $\mathrm{Edit}^{\geq}_{i,j}(\Sigma, k)$ be the automaton that accepts all strings of the form $u_1 \otimes u_2 \otimes \cdots \otimes u_r$ where for each $l \in [r]$, $u_l \in \Sigma^+$ and $\Delta(u_i, u_j) \geq k$, based on Lemma 11. Let A^r be the automaton accepting the set of all strings of the form $u_1 \otimes u_2 \otimes \cdots \otimes u_r$ where for each $l \in [r]$, $u_l \in L(A)$. Then, $\mathrm{MinDiv}(A, r, k)$ can be defined as the automaton that accepts the intersection of the languages $L(A^r)$ with the languages $L(\mathrm{Edit}^{\geq}_{i,j}(\Sigma, k))$ for $\{i, j\} \in \binom{[r]}{2}$. This implies that the number of states in $\mathrm{MinDiv}(A, r, k)$ is at most the product of the number of states of A^r with the number of states of $\mathrm{Edit}^{\geq}_{i,j}(\Sigma, k))$ for $\{i, j\} \in \binom{[r]}{2}$. According to Corollary 2, the DFA A^r has at most $(2|Q|)^r + 1$ states. Its construction costs $O(|A|^r)$ time. We claim below that one can define $\mathrm{Edit}^{\geq}_{i,j}(\Sigma, k))$ in such a way that it has at most $2^{2^{O(k \log |\Sigma|)}}$ states. As a consequence, one can construct $\mathrm{MinDiv}(A, r, k)$ in such a way that it has $2^{r^2 \cdot 2^{O(k \log |\Sigma|)}} \cdot |Q|^r$ states. Its construction takes $O(2^{r^2 \cdot 2^{O(k \log |\Sigma|)}} \cdot |A|^r)$ time.

Let $\rho_{i,j} : \Sigma^{\times k} \to \Sigma^{\times 2}$ be the map that sends each tuple $(a_1, a_2, \ldots, a_r)$ to the pair (a_i, a_j). Then, the automaton $\text{Edit}^{\geq}_{i,j}(\Sigma, k)$ can be defined as the automaton that accepts the inverse homomorphic image of $L(\text{Edit}^{\geq}(\Sigma, k))$ under $\rho_{i,j}$. More specifically, the automaton $\text{Edit}^{\geq}_{i,j}(\Sigma, k)$ is constructed from $\text{Edit}^{\geq}(\Sigma, k)$ by replacing each transition $(q, (a, b), q')$ with the set of transitions $\{(q, (a_1, \ldots, a_r), q') : (a_1, \ldots, a_r) \in \Sigma^{\times r},\ a_i = a,\ a_j = b\}$. Note that since no new states are created, the number of states in $\text{Edit}^{\geq}_{i,j}(\Sigma, k)$ is equal to the number of states in $\text{Edit}^{\geq}(\Sigma, k)$, that is, $2^{2^{O(k \log |\Sigma|)}}$ (Lemma 11). $\square$

6 Algorithmic Results

In this section, we state and prove our main results, starting with two preparatory observations.

Proposition 13 (Single-Word Automaton). *Let $w \in \Sigma^*$ and $A(w)$ be the minimal deterministic finite automaton accepting $\{w\}$. Then, $A(w)$ has $|w| + 2$ states.*

In combination with Lemma 5, we obtain the following.

Corollary 14. *Let $w \in \Sigma^*$. Then, the NFA $\text{Subseq}(A(w))$ has $|w| + 1$ states.*

Let us move on to questions that are more directly related to our discussions on diversity in synchronization. The next algorithmic result is interesting, as we prove intractability of the corresponding decision problem in Theorem 19 below.

Theorem 15. *Let $A = (Q, \Sigma, \delta, q_0, F)$ be a DFA. Given a word $w \in \Sigma^+$, one can determine in $O(2^{|Q|} \cdot |w| \cdot (|A| + \log |w|))$ time whether w has a subsequence that is synchronizing for A. If yes, in time $O(2^{|Q|} \cdot |w|^2 \cdot (|A| + \log |w|))$, or alternatively, in time $O(2^{2^{|Q|}} \cdot 2^{|A|} \cdot |w| \cdot \log(|w|))$, one can even construct a subsequence-minimal synchronizing word for A that is a subsequence of w.*

Proof. Let $\text{Subseq}(A(w))$ be the NFA of Corollary 14 with $|w| + 1$ states accepting all subsequences of w. Let $\text{Sync}(A)$ be the automaton accepting all words that are synchronizing for A. By Lemma 8, this DFA has $O(2^{|Q|})$ states and can be constructed in time $O(2^{|Q|}|A|)$. Then, w has some subsequence that is synchronizing for A if and only if $L(\text{Subseq}(A(w))) \cap L(\text{Sync}(A)) \neq \emptyset$, a condition that can be verified in time $O(2^{|Q|}|w|(|A| + \log |w|))$ by first constructing a product automaton (here, an NFA), and then by performing a reachability test. In the same time bound, we can find some $u \in L(\text{Subseq}(A(w))) \cap L(\text{Sync}(A))$ (if it exists).

For the last part, suppose that the intersection above is non-empty. Then, w also contains a subsequence-minimal synchronizing word for A. Using Lemma 7, we can construct an automaton $\text{MinSync}(A)$ that accepts the language of all synchronizing words of A that do not contain any subsequence that is also synchronizing for A. This construction takes time $O(2^{2^{|Q|}} \cdot |\text{Sync}(A)|) = O(2^{2^{|Q|}} \cdot 2^{|A|})$. Observe that $\text{MinSync}(A)$ has at most $O(2^{2^{|Q|}} \cdot 2^{|Q|})$ states. Hence, it is enough to output any word in the language $L(\text{MinSync}(A)) \cap L(\text{Subseq}(A(w)))$. Since $\text{MinSync}(A)$ has $O(2^{2^{|Q|}} \cdot 2^{|Q|})$ states and $A(w)$ has $|w| + 1$ states, we have that one can find a word in the intersection of the languages accepted by these automata in time $O(2^{2^{|Q|}} \cdot 2^{|Q|} \cdot |w| \log(|w|))$.

Alternatively, we can obtain an algorithm running in time $O(2^{|Q|} \cdot |w|^2 \cdot (|A| + \log |w|))$ as follows. Above, we described how to test whether $L(\text{Subseq}(A(w))) \cap L(\text{Sync}(A))$ is non-empty. If the intersection is empty, then w has no subsequence that is synchronizing for A. In the other case, such a subsequence exists. Let $w^{\downarrow i} = w[1..i-1]w[i+1..|w|]$. If $L(\text{Subseq}(A(w^{\downarrow i}))) \cap L(\text{Sync}(A)) = \emptyset$ for every $i \in [|w|]$, then we know that w is a subsequence-minimal synchronizing word for A. Otherwise, if this intersection is non-empty for some i, we know that $w^{\downarrow i}$ contains a subsequence-minimal synchronizing word for A. We then update w to $w^{\downarrow i}$, and repeat the process described in this paragraph. The algorithm runs in time $O(2^{|Q|} \cdot |w|^2 \cdot (|A| + \log |w|))$, since we need to delete at most $|w|$ letters, and at each deletion, one needs $O(2^{|Q|} \cdot |w| \cdot (|A| + \log |w|))$ steps to determine if $L(\text{Subseq}(A(w^{\downarrow i}))) \cap L(\text{Sync}(A)) \neq \emptyset$. $\square$

Next, we deal with the problem of finding a sufficiently diverse set W of subsequence-minimal synchronizing words for an automaton A that satisfy an additional constraint. More specifically, we require that each word in W belongs to the language of an automaton B given at the input. In this setting, A may be regarded as the specification of a system (say a robot) that interacts with an environment. Here, B models the set of all sequences of actions that are legal in the environment. The requirement $W \subseteq L(B)$ ensures that the synchronizing words under consideration correspond to sequences of actions that are legal in the environment. Note that it makes sense to assume that A is fixed (say, a robot of a certain model), and that the environment may vary (e.g., the environment where the robot will be deployed). The next theorem analyzes the computational complexity of this problem parameterized by the diversity parameters r and k. As A is assumed to be fixed, in the statement of the theorem we hide the dependencies on the DFA A in the function f_A.

Theorem 16. *Let $A = (Q, \Sigma, \delta)$ be a DFA and $B = (Q', \Sigma, \delta', Q'_0, F')$ be an NFA. One can determine in time $O(f_A(r, k) \cdot |Q'|^r \log(|Q'|))$ if there is a set $W \subseteq L(B)$ with r strings such that each word in W is subsequence-minimal synchronizing for A and $\text{MinDiv}(W) \geq k$.*

Proof. By combining Corollary 9 with Lemma 12, we can construct a DFA A' accepting the language of all compound words $u_1 \otimes \cdots \otimes u_r \in (\Sigma \cup \{\square\})^{\times r}$ such that for each $i \in [r]$, u_i is a subsequence-minimal synchronizing word for A, and $\text{MinDiv}(\{u_1, \ldots, u_r\}) \geq k$. The DFA A' has $2^{r^2 \cdot 2^{O(k \cdot \log |\Sigma|)}} \cdot (2^{|Q|})^r$ many states and can be constructed in time $2^{r^2 \cdot 2^{O(k \cdot \log |\Sigma|)}} \cdot (2^{|Q|} \cdot |\Sigma| \cdot |Q|)^r$. Conversely, again by an $(r + 1)$-fold product automaton construction, also using Lemma 1 and Corollary 2, an NFA B' with $(2|Q'|)^r + 1$ many states can be constructed that accepts the language

$$\{u_1 \otimes u_2 \cdots \otimes u_r : \forall i \in [r](u_i \in L(B))\}.$$

Checking if the product automaton C of A' and B' accepts any compound words solves the proposed problem.

This final check takes time linear in the size of C, which is $O\left(2^{r^2\cdot\left(|Q|+2^{O(k\cdot\log|\Sigma|)}\right)}\cdot|Q'|^r\log(|Q'|)\right)$. □

The previous result can be viewed as a diversity result concerning *synchronization under regular constraints*, introduced in (Fernau et al. 2019). This variation of the classical synchronization theme comes with a constraint automaton B; with $L(B)=\Sigma^*$, we get back to the classical theme.

Furthermore, as a direct consequence of Theorem 16 and Corollary 14 in combination with Lemma 5, we have that the problem of finding a set of sufficiently diverse subsequences of a word w that are subsequence-minimal synchronizing words for A can be solved by an algorithm with an FPT dependency on the parameters A (i.e., $|Q|$ and $|\Sigma|$) and k and an XP dependency on the parameter r.

Corollary 17. *Let $A=(Q,\Sigma,\delta)$ be a DFA and w be a word in Σ^*. One can determine in time $O(f_A(r,k)\cdot|w|^r\log(|w|))$ whether there is a set $W=\{w_1,\ldots,w_r\}$ of subsequences of w such that each word in W is subsequence-minimal synchronizing for A and* $\mathrm{MinDiv}(W)\geq k$.

7 Hardness Results

Below we show that the very problem of determining whether a given word is subsequence-minimal synchronizing for a given DFA A is coNP-hard.

Definition 18 (MIN-SUBSEQUENCE-SW).
***Given**: DFA $A=(Q,\Sigma,\delta)$ and $w\in\Sigma^*$ synchronizing A.*
***Question**: Is w a minimal synchronizing word with respect to the subsequence order?*

Theorem 19. MIN-SUBSEQUENCE-SW *is coNP-complete, even for DFAs over a binary input alphabet.*

This hardness result (and even more the hardness result for the counting class #P, our third main result of this section below in Theorem 24) explains why we have to develop exponential-time algorithms for the suggested diversity problems. We prove the hardness by a reduction from HITTING SET (Karp 1972) to the complementary problem of MIN-SUBSEQUENCE-SW which asks for the existence of a proper synchronizing subsequence of w.

Recall that above, in Theorem 15 we proved that MIN-SUBSEQUENCE-SW, parameterized by the number of states of the input DFA, belongs to FPT.

The following result explains also why the problems that we consider are computationally hard ones. Note that in the classical setting, length-minimal synchronizing words (if existent at all) are of polynomial length only (Volkov 2008). Requiring subsequence-minimality instead of length-minimality changes the picture drastically: with this property, some synchronizing words can be exponentially long.

Proposition 20. *Some subsequence-minimal synchronizing words can be of exponential length, even for DFAs with a ternary input alphabet.*

For binary input alphabets, we do not have such an example, giving an open question. For unary alphabets, subsequence-minimality equals length-minimality: lengths of shortest synchronizing words are polynomially bounded.

Next, we define two further combinatorial questions, which are in fact the central problems of our study. Above, we showed some algorithmic results that can be viewed as exponential-time algorithms for these problems. The fact that these algorithms exceed polynomial time is (with hindsight) justified by the hardness results proved next.

Definition 21 (DIVERSITY-SYNC).
***Given**: DFA $A=(Q,\Sigma,\delta)$ and $k\in\mathbb{N}$, encoded in binary.*
***Question**: Is there a set $W=\{w_1,w_2,\ldots,w_r\}$ of subsequence-minimal synchronizing words for A such that $\sum_{1\leq i<j\leq r}\Delta(w_i,w_j)\geq k$?*

A natural variant of the problem above is obtained when we ask for a set of subsequence-minimal synchronizing words of some minimal *cardinality* and neglect the diversity of the set. Note that subsequence-minimality already implies that the words have a pairwise distance greater than one.

Definition 22 (CARD-SYNC).
***Given**: DFA $A=(Q,\Sigma,\delta)$ and $r\in\mathbb{N}$, encoded in binary.*
***Question**: Is there a set $W=\{w_1,w_2,\ldots,w_r\}$ of subsequence-minimal synchronizing words for A?*

With a slight adaption of the proof of Theorem 19, we can show computational hardness for these problems.

Theorem 23. *The problems* DIVERSITY-SYNC *and* CARD-SYNC *are NP-hard.*

As we have shown in Theorem 19, verifying the subsequence minimality of a synchronizing word is coNP-hard and hence the problem is unlikely to be contained in NP, because we cannot use a guess & check approach. Further, for binary encoded parameters k and r, we obtain coNP-hardness from the reduction in Theorem 19 and hence should at least climb to Δ_2^P in the polynomial-time hierarchy for a membership attempt. Note that the NP-hardness shown below does not require binary parameters k and r, whereas coNP-hardness does (so far).

Finally, we show in the next theorem that it is #P-hard to compute the maximal diversity of the set of subsequence-minimal synchronizing words for a given DFA A.

Theorem 24. *Let A be a DFA. Then, on input A, computing the diversity of the set of all subsequence-minimal synchronizing words of A is #P-hard.*

8 Conformant Planning

Conformant planning (Goldman and Boddy 1996; Smith and Weld 1998) is the task of finding a sequence of actions for a planning problem that ensures that the goal will be achieved regardless of the initial state and of the non-determinism of the planning domain. The essence of many planning problems can be abstracted using the framework of automata theory (Cimatti and Roveri 2000; Cimatti, Roveri, and Bertoli 2004). We will use this point of view to define a suitable notion of diversity of solutions in the context of conformant planning. Apart from some notational differences, our terminology corresponds to (Cimatti and Roveri 2000).

A *planning domain* can be abstracted as a 4-tuple $D=(Q,\Sigma,\delta,P)$, where Q is a set of *states*, Σ is a set of *actions*, $\delta\subseteq Q\times\Sigma\times Q$ is a transition relation, and P is a function that assigns a set $P(q)$ of *propositions* (or *beliefs*) to each state $q \in Q$. Intuitively, $P(q)$ is the set of beliefs that are known to hold at state q, and a transition $(q, a, q') \in \delta$ indicates that the set of beliefs $P(q)$ should be updated to $P(q')$ if action a is taken from state q. Note that that the relation δ may be nondeterministic, meaning that for some state $q \in Q$ and some action $a \in \Sigma$, there may be two states q' and q'' such that both (q, a, q') and (q, a, q'') belong to δ. In this case, the result of taking the action a at state q is regarded as being undetermined. A *planning problem* is a triple (D, I, G) where $D = (Q, \Sigma, \delta, P)$ is a planning domain, $I \subseteq Q$ is a set of *initial states*, and $G \subseteq Q$ is a set of *goal states*. A word $u \in \Sigma^*$ is called a *plan*. An action a is said to be *applicable* in a state q if there is some state q' such that $(q, a, q') \in \delta$. Such an action a is applicable in a set of states S if it is applicable in *every* state of S. A plan u is said to be *applicable* in a set of states S if either $u = \varepsilon$ and S is non-empty, or $u = au'$ for some action a and some plan $u' \in \Sigma^*$, a is applicable on S and u' is applicable in $\delta(S, a)$. The key notion of a conformant plan is defined next.

Definition 25 (Conformant Plan)**.** *Let (D, I, G) be a planning problem with domain $D = (Q, \Sigma, \delta, P)$. A plan $u \in \Sigma^*$ is* conformant *for (D, I, G) if the following conditions are satisfied: (1) u is applicable in I, and (2) $\delta(I, u) \subseteq G$.*

Intuitively, the two conditions guarantee that a conformant plan u achieves a goal regardless of the initial state, and of the nondeterministic actions that may occur during the execution of u. The following lemma, an analogue of Lemma 8 in the context of conformant planning, gives an upper bound on the number of states in a DFA accepting all conformant words for a given planning problem (D, I, G).

Lemma 26 (Conformant Words)**.** *Let (D, I, G) be a planning problem with $D = (Q, \Sigma, \delta, P)$. One can build in time $|\Sigma| \cdot 2^{O(|Q|)}$ a DFA $\mathrm{Conf}(D, I, G)$ with $2^{|Q|}$ states such that*

$$L(\mathrm{Conf}(D, I, G)) = \{u \ : \ u \textit{ is conformant for } (D, I, G)\}.$$

Proof. Let $\mathrm{Conf}(D, I, G)$ be the DFA $\mathcal{A}$ whose set of states $\mathcal{Q}$ is the set of all subsets of Q, I is the unique initial state, and the set of final states $\mathcal{F}$ is the set of all non-empty subsets of G. The state $\emptyset$ acts as a trap state of $\mathrm{Conf}(D, I, G)$. The transition function $\Delta : \mathcal{Q} \times \Sigma \to \mathcal{Q}$ sends each pair $(S, a) \in \mathcal{Q} \times \Sigma$ to the state $\{q' \ : \ \exists q \in S, (q, a, q') \in \delta\}$ if a is applicable in every state of S, and to the state $\emptyset$ if a is not applicable from some state of S. The DFA $\mathrm{Conf}(D, I, G)$ has $2^{|Q|}$ states and can be constructed in time $|\Sigma| \cdot 2^{O(|Q|)}$. Additionally, by construction, we have that a plan $u = a_1 a_2 \ldots a_n$ is accepted by $\mathrm{Conf}(D, I, G)$ if and only if the plan u is applicable in I and $\delta(I, u) \subseteq G$, or, in other words, if and only if u is conformant for (D, I, G). $\square$

Let (D, I, G) be a planning problem and u be a conformant plan for (D, I, G); u is subsequence-minimal if no proper subsequence of u is a conformant plan for (D, I, G). The following corollary, which is an analogue of Corollary 9 in the realm of conformant planning, states that one can construct an automaton accepting all subsequence-minimal conformant plans for a given planning problem (D, I, G). The proof of this corollary follows directly by plugging the DFA $\mathrm{Conf}(D, I, G)$ of Lemma 26 into Lemma 7.

Corollary 27. *Let (D, I, G) be a planning problem with $D = (Q, \Sigma, \delta, P)$. In time $O(|\Sigma| \cdot 2^{2^{|Q|}+|Q|})$, we can build a DFA $\mathrm{ConfMinSubseq}(A)$ with $\leq 2^{2^{|Q|}+|Q|}$ many states for*

$$L(\mathrm{ConfMinSubseq}(A)) = \{u : u \textit{ is subsequence-minimal conformant for } (D, I, G)\}.$$

The next theorem is an analogue of Theorem 16; we consider the problem of computing a sufficiently diverse set W of subsequence-minimal conformant plans for a planning problem (D, I, G), with the property that each word in W belongs to the language of an NFA B given at the input. We assume that B is a specification of all legal plans that could occur in a given environment independent of (D, I, G). The proof of this theorem is identical to the proof of Theorem 16: only Corollary 27 is used instead of Corollary 9.

Theorem 28. *Let (D, I, G) be a planning problem with planning domain $D = (Q, \Sigma, \delta, P)$. Given an NFA $B = (Q', \Sigma, \delta', Q'_0, F')$, one can determine in time $O(f_D(r, k) \cdot |Q'|^r \log(|Q'|))$ whether there is a set $W \subseteq L(B)$ with r plans such that each plan in W is subsequence-minimal conformant for (D, I, G) and $\mathrm{MinDiv}(W) \geq k$.*

Diversity in conformant planning is relevant in the setting where we want to obtain a small set of sufficiently diverse plans for solving a given task. In settings where resilience is important, having two plans that are sufficiently distinct from each other may decrease the chances that both plans fail at the same time. In settings where variety is important, such as entertainment, diversity can be used to ensure that the alternative plans are not simply small variations of each other.

9 Conclusion

In this work, we have introduced a suitable notion of diversity of solutions in the context of the theory of synchronizing automata. Using this framework, we showed that for each $r, k \in \mathbb{N}$, each DFA A, and each NFA B over an alphabet Σ, the problem of computing a subset $\{w_1, \ldots, w_r\} \subseteq L(B)$ of subsequence-minimal synchronizing words for A, with pairwise edit distance of at least k, can be solved in time $O(f_A(r, k) \cdot |B|^r \log(|B|))$ for some function f depending only on A, r and k. Our algorithm has a fixed-parameter tractable dependency on the parameters $|A|$ and k and an XP dependency on the parameter r. Therefore, for each fixed r, our algorithm runs in FPT time when parameterized by $|A|$ and k. The existence of such FPT-algorithms was not clear even when the number r of solutions was fixed to 2. We leave the problem of determining whether one can obtain an algorithm of the form $O(f_A(r, k) \cdot |B|^{O(1)})$ as an interesting direction of further research. We have also shown that similar results hold in the context of conformant planning.

For some simplified versions of our problems, we proved NP- or coNP-hardness results. Possibly, our main problems are hard for higher levels of the polynomial hierarchy. Here, we point to recent papers (Burjons et al. 2019; Frei, Hemaspaandra, and Rothe 2020) that study hardness and membership of combinatorial problems with a flavor of minimality, placing them on low levels of this hierarchy.

Acknowledgements

Emmanuel Arrighi was supported by the Research Council of Norway (projects no. 274526 and no. 329745) and IS-DAAD (project no. 309319). Henning Fernau was supported by DAAD PPP (project no. 57525246). Mateus de Oliveira Oliveira was supported by the Research Council of Norway (projects no. 288761 and no. 326537) and by the Sigma2 Network (project no. NN9535K). Petra Wolf was supported by the DFG (project no. FE 560/9-1) and DAAD PPP (project no. 57525246).

References

Anders, A. S. 2019. *Reliably arranging objects: a conformant planning approach to robot manipulation*. Ph.D. thesis, Massachusetts Institute of Technology, USA.

Arrighi, E.; Fernau, H.; Lokshtanov, D.; de Oliveira Oliveira, M.; and Wolf, P. 2021. Diversity in Kemeny rank aggregation: a parameterized approach. In *International Joint Conference on Artificial Intelligence, 30th IJCAI*, 10–16. ijcai.org.

Baste, J.; Fellows, M. R.; Jaffke, L.; Masaík, T.; de Oliveira Oliveira, M.; Philip, G.; and Rosamond, F. A. 2020. Diversity of solutions: an exploration through the lens of fixed-parameter tractability theory. In *International Joint Conference on Artificial Intelligence, 29th IJCAI*, 1119–1125. ijcai.org.

Baste, J.; Jaffke, L.; Masařík, T.; Philip, G.; and Rote, G. 2019. FPT algorithms for diverse collections of hitting sets. *Algorithms*, 12: 254.

Birget, J.-C. 1993. Partial orders on words, minimal elements of regular languages and state complexity. *Theoretical Computer Science*, 119(2): 267–291.

Bonet, B. 2010. Conformant plans and beyond: principles and complexity. *Artificial Intelligence*, 174(3-4): 245–269.

Bringmann, K.; Grandoni, F.; Saha, B.; and Williams, V. V. 2019. Truly subcubic algorithms for language edit distance and RNA folding via fast bounded-difference min-plus product. *SIAM Journal on Computing*, 48(2): 481–512.

Bunke, H.; and Csirik, J. 1995. Parametric string edit distance and its application to pattern recognition. *IEEE Transactions on Systems, Man, and Cybernetics*, 25(1): 202–206.

Burjons, E.; Frei, F.; Hemaspaandra, E.; Komm, D.; and Wehner, D. 2019. Finding optimal solutions with neighborly help. In Rossmanith, P.; Heggernes, P.; and Katoen, J., eds., *International Symposium on Mathematical Foundations of Computer Science, 44th MFCS*, volume 138 of *LIPIcs*, 78:1–78:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik.

Chapman, A.; and Mesbahi, M. 2014. On symmetry and controllability of multi-agent systems. In *IEEE Conference on Decision and Control, 53rd CDC*, 625–630. IEEE.

Chevalier, P.-Y.; Hendrickx, J. M.; and Jungers, R. M. 2015. Reachability of consensus and synchronizing automata. In *IEEE Conference on Decision and Control, 54th CDC*, 4139–4144. IEEE.

Chua, S.-L.; Marsland, S.; and Guesgen, H. W. 2011. Unsupervised learning of patterns in data streams using compression and edit distance. In Walsh, T., ed., *International Joint Conference on Artificial Intelligence, 22nd IJCAI*, 1231–1236. IJCAI/AAAI.

Cimatti, A.; and Roveri, M. 2000. Conformant planning via symbolic model checking. *Journal of Artificial Intelligence Research*, 13: 305–338.

Cimatti, A.; Roveri, M.; and Bertoli, P. 2004. Conformant planning via symbolic model checking and heuristic search. *Artificial Intelligence*, 159(1-2): 127–206.

Demirovic, E.; and Schwind, N. 2020. Representative Solutions for Bi-Objective Optimisation. In *National Conference on Artificial Intelligence, 34th AAAI*, 1436–1443. AAAI Press.

Downey, R. G.; and Fellows, M. R. 1999. *Parameterized Complexity*. Monographs in Computer Science. Springer.

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

Fernau, H.; Gusev, V. V.; Hoffmann, S.; Holzer, M.; Volkov, M. V.; and Wolf, P. 2019. Computational complexity of synchronization under regular constraints. In Rossmanith, P.; Heggernes, P.; and Katoen, J., eds., *International Symposium on Mathematical Foundations of Computer Science, 44th MFCS*, volume 138 of *LIPIcs*, 63:1–63:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik.

Fomin, F. V.; Golovach, P. A.; Jaffke, L.; Philip, G.; and Sagunov, D. 2020. Diverse pairs of matchings. In Cao, Y.; Cheng, S.; and Li, M., eds., *International Symposium on Algorithms and Computation, 31st ISAAC*, volume 181 of *LIPIcs*, 26:1–26:12. Schloss Dagstuhl - Leibniz-Zentrum für Informatik.

Frei, F.; Hemaspaandra, E.; and Rothe, J. 2020. Complexity of Stability. In Cao, Y.; Cheng, S.; and Li, M., eds., *International Symposium on Algorithms and Computation, 31st ISAAC*, volume 181 of *LIPIcs*, 19:1–19:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik.

Goldman, R. P.; and Boddy, M. S. 1996. Expressive planning and explicit knowledge. In Drabble, B., ed., *International Conference on Artificial Intelligence Planning Systems, 3rd AIPS*, 110–117. AAAI.

Gruber, H.; Holzer, M.; and Kutrib, M. 2007. The size of Higman-Haines sets. *Theoretical Computer Science*, 387(2): 167–176.

Haines, L. H. 1969. On free monoids partially ordered by embedding. *Journal of Combinatorial Theory*, 6(1): 94–98.

Hebrard, E.; Hnich, B.; O'Sullivan, B.; and Walsh, T. 2005. Finding Diverse and Similar Solutions in Constraint Programming. In *National Conference on Artificial Intelligence, 20th AAAI*, 372–377.

Hennie, F. C. 1964. Fault detecting experiments for sequential circuits. In *SWCT (FOCS)*, 95–110. IEEE Computer Society.

Higman, G. 1952. Ordering by divisibility in abstract algebras. *Proceedings of the London Mathematical Society, Series 3*, 2(7): 326–336.

Hoffmann, S. 2021. Completely reachable automata, primitive groups and the state complexity of the set of synchronizing words. In Leporati, A.; Martín-Vide, C.; Shapira, D.; and Zandron, C., eds., *Language and Automata Theory and Applications - 15th International Conference, LATA*, volume 12638 of *LNCS*, 305–317.

Ingmar, L.; de la Banda, M. G.; Stuckey, P. J.; and Tack, G. 2020. Modelling diversity of solutions. In *National Conference on Artificial Intelligence, 34th AAAI*, 1528–1535. AAAI Press.

Karp, R. M. 1972. Reducibility Among Combinatorial Problems. In *Proceedings of a symposium on the Complexity of Computer Computations*, The IBM Research Symposia Series, 85–103.

Kohavi, Z.; and Jha, N. K. 2009. *Switching and Finite Automata Theory*. Cambridge University Press, 3rd edition.

Levenshtein, V. I. 1966. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics Doklady*, 10(8): 707–710.

Lyras, D. P.; Sgarbas, K. N.; and Fakotakis, N. D. 2007. Using the Levenshtein edit distance for automatic lemmatization: A case study for modern Greek and English. In *IEEE International Conference on Tools with Artificial Intelligence, 19th ICTAI*, volume 2, 428–435. IEEE Computer Society.

Maubert, B. 2009. *Synchronizing automata and their applications to games with imperfect information*. Research Master's Thesis, Computer Science, Université Rennes, France.

Natarajan, B. K. 1986. An algorithmic approach to the automated design of parts orienters. In *Annual Symposium on Foundations of Computer Science, 27th FOCS*, 132–142. IEEE Computer Society.

Palacios, H.; and Geffner, H. 2009. Compiling uncertainty away in conformant planning problems with bounded width. *Journal of Artificial Intelligence Research*, 35: 623–675.

Petit, T.; and Trapp, A. C. 2019. Enriching solutions to combinatorial problems via solution engineering. *INFORMS Journal on Computing*, 31(3): 429–444.

Rystsov, I. K. 1980. On minimizing the length of synchronizing words for finite automata. In *Theory of Designing of Computing Systems*, 75–82. Institute of Cybernetics of the Ukrainian Academy of Sciences. (in Russian).

Sandberg, S. 2005. Homing and synchronizing sequences. In Broy, M.; Jonsson, B.; Katoen, J.; Leucker, M.; and Pretschner, A., eds., *Model-Based Testing of Reactive Systems, Advanced Lectures*, volume 3472 of *LNCS*, 5–33. Springer.

Sayin, S. 2000. Measuring the quality of discrete representations of efficient sets in multiple objective mathematical programming. *Mathematical Programmming*, 87(3): 543–560.

Smith, D. E.; and Weld, D. S. 1998. Conformant graphplan. In Mostow, J.; and Rich, C., eds., *National Conference on Artificial Intelligence and Innovative Applications of Artificial Intelligence Conference, 15th AAAI & 10th IAAI*, 889–896. AAAI Press / The MIT Press.

Starke, P. H. 1966. Eine Bemerkung über homogene Experimente. *Elektronische Informationsverarbeitung und Kybernetik*, 2(4): 257–259.

Türker, U. C.; and Yenigün, H. 2015. Complexities of some problems related to synchronizing, non-synchronizing and monotonic automata. *International Journal on Foundations of Computer Science*, 26(1): 99–122.

Vaz, D.; Paquete, L.; Fonseca, C. M.; Klamroth, K.; and Stiglmayr, M. 2015. Representation of the non-dominated set in biobjective discrete optimization. *Computers & Operations Research*, 63: 172–186.

Volkov, M. V. 2008. Synchronizing automata and the Černý conjecture. In Martín-Vide, C.; Otto, F.; and Fernau, H., eds., *Language and Automata Theory and Applications, 2nd International Conference, LATA*, volume 5196 of *LNCS*, 11–27. Springer.

Wagner, R. A. 1975. On the complexity of the Extended String-to-String Correction Problem. In Rounds, W. C.; Martin, N.; Carlyle, J. W.; and Harrison, M. A., eds., *Annual ACM Symposium on Theory of Computing, 7th STOC*, 218–223. ACM.

Wagner, R. A.; and Fischer, M. J. 1974. The string-to-string correction problem. *Journal of the ACM*, 21(1): 168–173.

Wolf, P. 2020. Synchronization Under Dynamic Constraints. In Saxena, N.; and Simon, S., eds., *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, 40th FSTTCS*, volume 182 of *LIPIcs*, 58:1–58:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik.