Computing Equality-Free and Repetitive String Factorisations[☆]

Markus L. Schmid

Trier University, Fachbereich IV – Abteilung Informatikwissenschaften, D-54286 Trier, Germany

Abstract

For a string w, a factorisation is any tuple $(u_1, u_2, \ldots, u_k)$ of strings that satisfies $w = u_1 \cdot u_2 \cdots u_k$. A factorisation is called equality-free if each two factors are different, its size is the number of factors (counting each occurrence of repeating factors) and its width is the maximum length of any factor. To decide, for a string w and a number m, whether w has an equality-free factorisation with a size of at least (or a width of at most) m are NP-complete problems. We further investigate the complexity of these problems and we also study the converse problems of computing a factorisation that is to a large extent not equality-free, i. e., a factorisation of size at least (or width at most) m such that the total number of different factors does not exceed a given bound k.

Keywords: String factorisations, NP-hard string problems, parameterised complexity, multivariate algorithmics
2000 MSC: 68Q17, 68Q25, 68W32

1. Introduction

Many classical hard string problems can be defined in terms of factorisations of strings that satisfy certain properties. For example, the well-known problem of computing the shortest common superstring of given strings $w_1, \ldots, w_k$ (see, e. g., Bulteau et al. [1]) asks whether there exists a short string x that, for every i, $1 \leq i \leq k$, has a factorisation $u_i \cdot w_i \cdot v_i$. Since a string w is a subsequence of a string u if u has a factorisation $v_1 \cdot v_2 \cdots v_k$ and w has a factorisation $v_{i_1} \cdot v_{i_2} \cdots v_{i_n}$ with $1 \leq i_1 < \ldots < i_n \leq k$, the famous Longest Common Subsequence and Shortest Common Supersequence problems can as well be described in terms of factorisations. Another example of a string problem that has recently attracted much attention is the problem to decide for two words x and y and a given k whether they have factorisations $u_1 \cdot u_2 \cdots u_k$ and $v_1 \cdot v_2 \cdots v_k$, respectively, such that $(u_1, \ldots, u_k)$ is a permutation of $(v_1, \ldots, v_k)$,

[☆] A preliminary version [12] of this paper was presented at the conference CiE 2015.
Email address: MSchmid@uni-trier.de (Markus L. Schmid)

i. e., the Minimum Common String Partition problem. See Bulteau et al. [1] for a survey on the multivariate analysis of NP-hard string problems.

In this paper we are concerned with so-called equality-free factorisations, recently introduced by Condon et al. [2, 3]. A factorisation $u_1 \cdot u_2 \cdots u_k$ is *equality-free* if every factor is distinct, i. e., $|\{u_1, u_2, \ldots, u_k\}| = k$. Condon et al. [2, 3] investigate the problem of deciding whether a given string w has an equality-free factorisation of *width* at most m, where the width is the maximum length of any factor. This problem is also mentioned by Bulteau et al. [1]; furthermore, Gagie et al. [6] investigate the hardness of computing an equality-free factorisation with only palindromes as factors. A motivation for this problem comes from gene synthesis. Since it is only possible to artificially produce short pieces of DNA (so-called *oligo fragments*), longer DNA sequences are usually obtained by a self-assembly of many oligos into the desired DNA sequence; thus, the task is to find the right oligos for successful self-assembly. Computing equality-free factorisations with bounded width is an abstraction of this problem: the width bound represents the necessity for short oligos and the equality-freeness models the condition that each two oligos must not be too similar in order to not hybridise with each other (see Condon et al. [2, 3] for more details). This problem is NP-complete, even if the width bound is 2 or the alphabet is binary (see Condon et al. [3]). We revisit this problem and show that it is fixed-parameter tractable if both the width bound and the alphabet size are parameters.

If instead of a small width, we are looking for an equality-free factorisation with a large *size*, i. e., a large number of factors, then we obtain a different NP-complete problem (see Fernau et al. [4]). This variant is motivated by injective pattern matching with variables (which is identical to the special case of solving word equations (see Lothaire [10]), where the left side of the equation does not contain variables and different variables must be replaced by different words), see Fernau et al. [4] for more details. We show that computing equality-free factorisations with large size is fixed-parameter tractable if parameterised by the size bound. However, the question whether the problem remains hard for fixed alphabets is still open.

We also consider the converse of computing equality-free factorsations, i. e., computing factorisations that are to a large extent not equality-free (or *repetitive*). Our measure of repetitiveness is the number of different factors in the factorisation. If this number is small (in comparison to the size or width of the factorisation), then many factors are repeated. This yields an interesting combinatorial question in its own right: how many different words are needed in order to cover a given word? Furthermore, it is motivated by data compression, since a factorisation with many repeated factors can be used in order to compress a word, e. g., by using a dictionary of the different factors. We can show that deciding on whether a word w has a factorisation of width at most m and with at most k different factors is NP-complete, even if $m = 2$. On the other hand, if k or the alphabet size is a constant, then the problem can be solved in polynomial time. In contrast to this, if m is a lower bound on the size of the factorisation, then the problem can be solved in polynomial time if either m, k or the alphabet size is a constant, but it is open, whether the problem is

NP-complete in general.

As a tool for proving some of our main results, we also investigate the problem of deciding whether a given word w has an equality-free factorisation with only factors from a given finite set F of words. It turns out that this problem is NP-complete even for binary alphabets. However, it is in FPT if $|F|$ is a parameter and in P if we drop the equality-freeness condition.

This paper is organised as follows. In Section 2, we define the central concepts of this work, the problems to be investigated, and we make some basic observations. In Section 3, we consider the problem of finding a factorisation that only uses factors from a given set. Then, in Sections 4 and 5, we investigate the problems of computing equality-free factorisations and repetitive factorisations, respectively. Finally, we conclude this work in Section 6, where we categorise the investigated problems in terms of parameterised complexity and discuss open problems.

2. Preliminaries

We start this section with some general and standard notations about words. Then we shall define the central concepts, i. e., equality-free and repetitive factorisations, and the computational problems to be investigated in this work.

Let $\mathbb{N} = \{1, 2, 3, \ldots\}$. By $|A|$, we denote the cardinality of a set A. Let Σ be a finite alphabet of *symbols*. A *word* or *string* (over Σ) is a sequence of symbols from Σ. For any word w over Σ, $|w|$ denotes the length of w and ε denotes the *empty word*, i. e., $|\varepsilon| = 0$. The symbol Σ^+ denotes the set of all non-empty words over Σ and $\Sigma^* = \Sigma^+ \cup \{\varepsilon\}$. For the *concatenation* of two words w_1, w_2 we write $w_1 \cdot w_2$ or simply $w_1 w_2$. For every symbol $a \in \Sigma$, by $|w|_a$ we denote the number of occurrences of symbol a in w. We say that a word $v \in \Sigma^*$ is a *factor* of a word $w \in \Sigma^*$ if there are $u_1, u_2 \in \Sigma^*$ such that $w = u_1 v u_2$. If $u_1 = \varepsilon$ or $u_2 = \varepsilon$, then v is a *prefix* (or a *suffix*, respectively) of w. For every i, $1 \leq i \leq |w|$, by $w[1..n]$ we denote the prefix of w with length n. As a convention, in this work every set of words is always a finite set.

By the term *trie*, we refer to the well-known ordered tree data structure for representing sets of words.

2.1. Factorisations

For any word $w \in \Sigma^+$, a *factorisation of* w is a tuple $p = (u_1, u_2, \ldots, u_\ell) \in (\Sigma^+)^\ell$, $\ell \in \mathbb{N}$, with $w = u_1 u_2 \cdots u_\ell$. Every word u_i, $1 \leq i \leq \ell$, is called a *factor* (*of* p). For the sake of readability, we sometimes represent a factorisation $(u_1, u_2, \ldots, u_\ell)$ in the form $u_1 \mid u_2 \mid \cdots \mid u_\ell$.

Let $p = (u_1, u_2, \ldots, u_\ell)$ be an arbitrary factorisation. We define several parameters of the factorisation p:

- the *set of factors* $\mathsf{sf}(p) = \{u_1, u_2, \ldots, u_\ell\}$,
- the *size* $\mathsf{s}(p) = \ell$,
- the *cardinality* $\mathsf{c}(p) = |\mathsf{sf}(p)|$,

- the *width* $\mathsf{w}(p) = \max\{|u_i| \mid 1 \leq i \leq \ell\}$.

A factorisation p is *equality-free* if $\mathsf{s}(p) = \mathsf{c}(p)$ and if p is not equality-free, then we call it *repetitive*.

2.2. Problems

We now define the different problems to be investigated in this work. The first problem is to check whether a given word can be factorised by an equality-free factorisation that only uses factors from a given set of words.

Equality-Free Factor Cover (EFFC)

Instance: A word w and a set F of words represented as a trie.
Question: Does there exist an equality-free factorisation p of w with $\mathsf{sf}(p) \subseteq F$?

We investigate this problem, since it naturally arises in the context of this work. Furthermore, EFFC and its variant where we are looking for any factorisation instead of an equality-free one (denoted by FC) shall be important later on in Sections 4 and 5.

Next, we define the problem of computing an equality-free factorisation, of which there are two different variants, i. e., either we are looking for an equality-free factorisation with a large size or with a small width (note that asking for a small size or a large width does not make sense, since (w) is always a factorisation of a word w with maximum width and minimum size).

Maximum Equality-Free Factorisation Size (MaxEFF-s)

Instance: A word w and a number α, $1 \leq \alpha \leq |w|$.
Question: Does there exist an equality-free factorisation p of w with $\mathsf{s}(p) \geq \alpha$?

The problem MinEFF-w is identical to MaxEFF-s, but instead of a lower bound α on the size of the factorisation, we get an upper bound β on the width of the factorisation.

We also investigate the problems of computing a factorisation of a word w with as few different factors as possible (and with either a large size or a small width). In a sense, factorisations of this kind are to a large extent repetitive, since if the number of different factors is very small (with respect to the bound on the size or width), then many factors must be repeated.

Maximum Repetitive Factorisation Size (MaxRF-s)

Instance: A word w, numbers α, $1 \leq \alpha \leq |w|$, and k, $1 \leq k \leq |w|$.
Question: Does there exist a factorisation p of w with $\mathsf{s}(p) \geq \alpha$ and $\mathsf{c}(p) \leq k$?

Similar as before, the problem MinRF-w is identical to MaxRF-s, but instead of a lower bound α on the size of the factorisation, we get an upper bound β on the width of the factorisation.

In the remainder of the paper, the symbols α, β and k are exclusively used as bounds on the size, width and cardinality of factorisations, respectively, as introduced in the problem definitions above. For any problem K from above and any fixed alphabet Σ, K_Σ denotes the problem K, where the input word is over Σ.

We shall now illustrate these definitions with an example.

Example 1. Let $p = \mathtt{aab} \mid \mathtt{ba} \mid \mathtt{cba} \mid \mathtt{aab} \mid \mathtt{ba} \mid \mathtt{aab}$ be a factorisation. We note that $\mathsf{sf}(p) = \{\mathtt{aab}, \mathtt{ba}, \mathtt{cba}\}$, $\mathsf{s}(p) = 6$, $\mathsf{c}(p) = 3$ and $\mathsf{w}(p) = 3$. Furthermore, p is not equality-free, since, e. g., the factor $\mathtt{aab}$ has three occurrences.

The word $\mathtt{abbcbaabbc}$ has an equality-free factorisation of size 6, namely $\mathtt{a} \mid \mathtt{bb} \mid \mathtt{c} \mid \mathtt{ba} \mid \mathtt{ab} \mid \mathtt{bc}$, i. e., $(\mathtt{abbcbaabbc}, 6) \in$ MaxEFF-s. On the other hand, $(\mathtt{abbcbaabbc}, 7) \notin$ MaxEFF-s, which can be easily verified by observing that any factorisation of size 7 of $\mathtt{abbcbaabbc}$ needs at least 4 factors of size 1 and therefore cannot be equality-free.

The instance $(\mathtt{aabbccaabbcc}, \beta)$ is a positive MinEFF-w instance if and only if $\beta \geq 2$, whereas $(\mathtt{aabbccaabbccaabbcc}, \beta) \in$ MinEFF-w if and only if $\beta \geq 3$.

2.3. Parameterised Complexity Theory

We assume the reader to be familiar with the basic concepts of complexity theory (for unexplained notions, see Papadimitriou [11]). Since we are also using the framework of parameterised complexity, we shall briefly define the concepts relevant for our results (for detailed explanations on parameterised complexity, the reader is referred to Flum and Grohe [5]).

We consider decision problems as languages over some alphabet Γ. A *parameterisation* (*of* Γ) is a polynomial-time computable mapping $\kappa : \Gamma^* \to \mathbb{N}$ and a *parameterised problem* is a pair (Q, κ), where Q is a problem (over Γ) and κ is a parameterisation of Γ. We usually define κ implicitly by describing which part of the input is the parameter. A parameterised problem (Q, κ) is *fixed-parameter tractable* if there is an *fpt-algorithm* for it, i. e., an algorithm that solves Q on input x in time $f(\kappa(x)) \times p(|x|)$ for recursive f and polynomial p. The class of fixed-parameter tractable problems is denoted by FPT.

Let (Q, κ) be a parameterised problem. Then, for every $\ell \in \mathbb{N}$, we can obtain a classical decision problem $(Q, \kappa)_\ell$ by only considering those instances x of Q that satisfy $\kappa(x) = \ell$, i. e., $(Q, \kappa)_\ell = \{x \mid x \in Q, \kappa(x) = \ell\}$. The class XP is the class of all parameterised problems (Q, κ), such that, for every $\ell \in \mathbb{N}$, $(Q, \kappa)_\ell \in \mathsf{P}$. Equivalently, $(Q, \kappa) \in \mathsf{XP}$ if Q can be solved by an algorithm with running time $\mathcal{O}(|x|^{\kappa(x)})$; note that this is a weaker condition than the existence of an fpt-algorithm (note that $\mathsf{FPT} \subseteq \mathsf{XP}$ obviously holds).

While the membership to FPT (or to XP) of a parameterised problem (Q, κ) is demonstrated by finding an fpt-algorithm (or an algorithm with running time $\mathcal{O}(|x|^{\kappa(x)})$, respectively), the framework of parameterised complexity provides several ways of arguing (under complexity theoretical assumptions) that a parameterised problem is not in FPT, or, in other words, that it is *fixed-parameter intractable*. A rather robust way of doing so is to show that, for at least one $\ell \in \mathbb{N}$, $(Q, \kappa)_\ell$ is NP-hard, since then, under the assumption $\mathsf{P} \neq \mathsf{NP}$, $(Q, \kappa) \notin \mathsf{XP}$, which implies $(Q, \kappa) \notin \mathsf{FPT}$.

We shall present all our results in terms of classical complexity, i. e., we present algorithms with running time estimations in the usual way, as well as NP-hardness reductions. Then, at the end of this paper, in Section 6, we give a detailed interpretation of our results in terms of parameterised complexity.

2.4. Basic Observations

In this section, we make some simple observations that shall be helpful for proving our main results. In this regard, we first note that every equality-free factorisation p can be transformed into an equality-free factorisation of the same word with one less factor (i. e., its size decreases by 1). More precisely, this is achieved by joining one of the longest factors with one of its neighbours. In particular, this means that in order to solve an instance of MaxEFF-s, it is sufficient to check whether there exists an equality-free factorisation of size exactly α.

Observation 1. A word w has an equality-free factorisation p with $\mathsf{s}(p) \geq \alpha$, $\alpha \in \mathbb{N}$, if and only if it has an equality-free factorisation p' with $\mathsf{s}(p') = \alpha$.

Given some factorisation, it is not a hard task to check whether or not it is equality-free: we simply insert all factors in a trie, while checking for each factor if it is already contained in the trie.

Observation 2. Let $w \in \Sigma^+$ and let p be a factorisation of w. It can be decided in time $\mathcal{O}(|w|)$ whether or not p is equality-free.

Similarly, we can check, for a factorisation p and a set F of words given as a trie, whether or not $\mathsf{sf}(p) \subseteq F$.

Observation 3. Let $w \in \Sigma^+$, let p be a factorisation of w and let F be a set of words over Σ represented as a trie. It can be decided in time $\mathcal{O}(|w|)$ whether or not $\mathsf{sf}(p) \subseteq F$.

Any word w has only $f(|w|)$ factorisations, for a function f; thus, the following result is straightforward, although it contributes to our understanding of the complexity of the considered problems.

Proposition 1. *The problems* EFFC, MaxEFF-s, MinEFF-w, MaxRF-s *and* MinRF-w *are in* FPT *with respect to parameter* $|w|$.

3. Computing (Equality-Free) Factorisations From Given Factors

As mentioned in the introduction, the problem EFFC is not our main concern. However, its investigation, as we shall see, yields some valuable insights with respect to equality-free factorisations and we also obtain an algorithm that shall be used later in order to prove tractability results with respect to the problems MaxRF-s and MinRF-w.

We first show that EFFC is NP-complete, even for fixed binary alphabets, by a simple reduction from MinEFF-w.

Theorem 2. *Let* Σ *be an alphabet with* $|\Sigma| = 2$. *Then* EFFC_Σ *is* NP*-complete.*

Proof. Since we can guess a factorisation p and check in polynomial-time both whether it is equality-free and whether $\mathsf{sf}(p) \subseteq F$, EFFC_Σ is in NP. In order to prove hardness, we reduce MinEFF-w_Σ to EFFC_Σ. To this end, let (w, β) be an instance of MinEFF-w_Σ and let F be the set of all factors of w of length

at most β. We note that $|F| \leq \sum_{i=1}^{\beta} |w| - (i-1) \leq \beta \times |w|$ and F can be constructed in time $\mathcal{O}(|F| \times \beta)$. The word w has an equality-free factorisation p with $\mathsf{w}(p) \leq \beta$ (i. e., $(w, \beta) \in$ MinEFF-w_Σ) if and only it has an equality-free factorisation p' with $\mathsf{sf}(p') \subseteq F$ (i. e., $(w, F) \in \mathrm{EFFC}_\Sigma$). Since MinEFF-$\mathsf{w}_\Sigma$ is NP-complete (see Condon et al. [3]), EFFC_Σ is NP-complete as well. □

In addition to the alphabet size, the cardinality of the given set F of factors is another natural parameter and it can be easily seen that the hardness is not preserved if we bound $|F|$ by a constant (in contrast to bounding $|\Sigma|$). More precisely, if there is an equality-free factorisation p of a word w with $\mathsf{sf}(w) \subseteq F$ for a set of factors F, then $\mathsf{s}(p) \leq \ell = \min\{|w|, |F|\}$. Regardless of whether $\ell = |w|$ or $\ell = |F|$, enumerating all factorisations p of w with $\mathsf{s}(p) \leq \ell$ and checking whether $\mathsf{sf}(p) \subseteq F$ (see Observations 3) and whether p is equality-free (see Observations 2) can be done in time $\mathcal{O}(|w|^{|F|+1})$.

However, by dynamic programming, we can conduct another algorithm for EFFC that is also exponential only in $|F|$, but that is even an fpt-algorithm with respect to the parameter $|F|$.

Theorem 3. *The Problem* EFFC *can be solved in time* $\mathcal{O}(|F| \times |w|^2 \times 2^{|F|})$.

Proof. Let $w \in \Sigma^*$ and $F \subseteq \Sigma^*$ be an instance of EFFC. We describe a dynamic programming algorithm that computes, for every i, $1 \leq i \leq |w|$, and every $F' \subseteq F$, whether or not $w[1..i]$ has an equality-free factorisation p with $\mathsf{sf}(p) = F'$.

First, for every $u \in F$, we set $T[|u|, \{u\}] = 1$ if u is a prefix of w and $T[|u|, \{u\}] = 0$ otherwise. Then, for every i, $1 \leq i \leq |w|$, and every $F' \subseteq F$ with $|F'| \geq 2$, we set $T[i, F'] = 1$ if there exists a $u \in F'$ with $T[i - |u|, F' \setminus \{u\}] = 1$ and $w[i-|u|+1..i] = u$, and $T[i, F'] = 0$ otherwise. Any $T[i, F']$ with $1 \leq i \leq |w|$ and $F' \subseteq F$ can be computed in time $\mathcal{O}(|F| \times |w|)$; thus, the overall running time of this procedure is $\mathcal{O}(|F| \times |w|^2 \times 2^{|F|})$. □

Next, we investigate the impact of the equality-freeness condition itself, i. e., we consider the problem FC (recall that FC is identical to EFFC with the only difference that the factorisation p of w with $\mathsf{sf}(p) \subseteq F$ does not need to be equality-free). This problem is similar to the problem Exact Block Cover (recently investigated by Jiang et al. in [8]), which differs from FC only in that instead of a set we are given a sequence of factors and every factor of the sequence has to be used exactly once (in particular, this coincides with the variant of Minimum Common String Partition where the partition of one of the two strings is already fixed). While Exact Block Cover is NP-complete (see Jiang et al. [8]), FC can be solved in polynomial-time by dynamic programming. This demonstrates that it is really the equality-freeness condition that makes EFFC hard and, in addition, we obtain a useful tool to devise algorithms for solving variants of the problems MaxRF-s and MinRF-w later on in Section 5.

Theorem 4. *The problem* FC *can be solved in time* $\mathcal{O}(|F| \times |w|^2)$.

Proof. We define a dynamic programming algorithm. Let w be a word and $F = \{u_1, u_2, \ldots, u_\ell\}$. For every n, m, $1 \leq m \leq n \leq |w|$, let $T[n, m] = 1$ if there exists a factorisation p of size m of $w[1..n]$ with $\mathsf{sf}(p) \subseteq F$ and $T[n, m] = 0$ otherwise. Obviously, (w, F) is a positive instance of FC if and only if $T[|w|, m] = 1$ for some m, $1 \leq m \leq |w|$. We can now solve FC on instance (w, F) by computing all the $T[n, m]$, $1 \leq m \leq n \leq |w|$, in the following way.

In time $\mathcal{O}(|w| \times |F|)$, we first construct a table S with $|w|$ rows and ℓ columns with $S[n, i] = 0$, $1 \leq n \leq |w|$, $1 \leq i \leq \ell$. Then, by using the Knuth-Morris-Pratt algorithm [9], for every i, $1 \leq i \leq \ell$, we set $S[n, i] = 1$ if u_i is a suffix of $w[1..n]$. Since the Knuth-Morris-Pratt algorithm has running time $\mathcal{O}(|w|+|u_i|)$, building up this table can be done in time $\sum_{i=1}^{\ell}(|w| + |u_i|) \leq \sum_{i=1}^{\ell} 2|w| = \mathcal{O}(|F| \times |w|)$. Then, for every n, m, $1 \leq m \leq n \leq |w|$, we initialise $T[n, m] = 0$, which requires time $\mathcal{O}(|w|^2)$, and, for every i, $1 \leq i \leq \ell$, we set $T[|u_i|, 1] = 1$ if $S[|u_i|, i] = 1$, which requires time $\mathcal{O}(|F|)$. We note that, for every n, m with $2 \leq m \leq n \leq |w|$, $T[n, m] = 1$ if and only if there exists a word $u_i \in F$ that is a suffix of $w[1..n]$ (i. e., $S[n, i] = 1$) with $T[n - |u_i|, m - 1] = 1$. Thus, for every n, m, $2 \leq m \leq n \leq |w|$, we can compute $T[n, m]$ in time $\mathcal{O}(|F|)$, provided that all $T[n', m - 1]$, $n' < n$, have already been computed, which is satisfied if we iterate over m, $2 \leq m \leq |w|$, in an outer loop and over n, $m \leq n \leq |w|$, in an inner loop. Hence, all the elements $T[n, m]$, $1 \leq m \leq n \leq |w|$, are computed in time $\mathcal{O}(|F| \times |w|^2)$. □

In the next two sections, we investigate the central problems of this work, i. e., computing equality-free and repetitive factorisations with either a large size or a small width.

4. Computing Equality-Free Factorisations

We now investigate the problems MinEFF-w and MaxEFF-s. Their NP-completeness is established by Condon et al. [2] and by Fernau et al. [4], respectively, but in [3] it is additionally shown that MinEFF-w remains NP-complete even if the bound on the width is 2 or the alphabet is fixed and binary. However, as pointed out by the following result, there is an fpt-algorithm for MinEFF-w with respect to the parameters β and the alphabet size. In particular, this shows that the NP-completeness is not preserved if both β and $|\Sigma|$ are bounded.

Theorem 5. MinEFF-w_Σ *can be solved in time* $\mathcal{O}(|\Sigma|^\beta \times |w|^2 \times 2^{2|\Sigma|^\beta})$.

Proof. Let (w, β) be an instance of MinEFF-w_Σ. We first define $F_{\leq\beta} = \{u \in \Sigma^+ \mid |u| \leq \beta\}$ and note that every equality-free factorisation p of w with $\mathsf{w}(p) \leq \beta$ satisfies $\mathsf{sf}(p) \subseteq F_{\leq\beta}$. Consequently, we can solve MinEFF-w_Σ by running the algorithm from the proof of Theorem 3 on input w and $F_{\leq\beta}$. Since $|F_{\leq\beta}| = \sum_{\ell=1}^{\beta} |\Sigma|^\ell \leq 2 \times |\Sigma|^\beta$, this algorithm has a running time of $\mathcal{O}(|F_{\leq\beta}| \times |w|^2 \times 2^{|F_{\leq\beta}|}) = \mathcal{O}(|\Sigma|^\beta \times |w|^2 \times 2^{2|\Sigma|^\beta})$. □

We wish to point out that it is also possible to express $|w|$ in the running time of Theorem 5 in terms of $|\Sigma|$ and β, i. e., we can assume that $|w| \leq \beta \times 2 \times |\Sigma|^\beta$,

since otherwise clearly no equality-free factorisation with a width of at most β can exist.

For the problem MaxEFF-s, i. e., deciding on the existence of an equality-free factorisation with a size of at least α (instead of a width of at most β), we encounter a slightly different situation. First of all, it is still an open problem whether MaxEFF-s remains NP-complete if the alphabet is fixed:

Open Problem 1. Let Σ be an alphabet. Is MaxEFF-s$_\Sigma$ NP-complete?

From an intuitive point of view, for the problem MinEFF-w, the bound on the width can conveniently be exploited in order to design gadgets for encoding an NP-hard problem (see Condon et al. [3] and also the proof of Theorem 13, Section 5). A lower bound on the size seems to provide fewer possibilities for controlling the structure of the factorisation, which makes it difficult to express another NP-complete problem by MaxEFF-s (especially if we have only a constant number of symbols at our disposal). On the other hand, a constant alphabet does not seem to help in order to find an equality-free factorisation with a size of at least α in polynomial-time.

However, if we consider α as a constant, then the problem is not NP-complete anymore (in fact, it is even fixed-parameter tractable with respect to α):

Theorem 6. *The problem* MaxEFF-s *can be solved in time* $\mathcal{O}((\frac{\alpha^2+\alpha}{2}-1)^\alpha)$.

Proof. Let (w, α) be an instance of MaxEFF-s. If $|w| \geq \Sigma_{i=1}^{\alpha} i = \frac{\alpha^2+\alpha}{2}$, then the factorisation $(u_1, u_2, \ldots, u_\alpha)$ of w with $|u_i| = i$, $1 \leq i \leq \alpha - 1$, and $|u_\alpha| = |w| - |u_1 u_2 \cdots u_{\alpha-1}|$ is equality-free, since each two factors have a different length. If, on the other hand, $|w| \leq \frac{\alpha^2+\alpha}{2} - 1$, then we can enumerate all factorisations of size α of w in time $\mathcal{O}(|w|^{\alpha-1})$ and, by Observation 2, check in time $\mathcal{O}(|w|)$ for each such factorisation whether or not it is equality-free. Since w has an equality-free factorisation of size at least α if and only if it has an equality-free factorisation of size exactly α (see Observation 1), this solves the problem MaxEFF-s in time $\mathcal{O}(|w|^\alpha) = \mathcal{O}((\frac{\alpha^2+\alpha}{2}-1)^\alpha)$. □

5. Computing Repetitive Factorisations

In this section, we investigate the problem of finding a factorisation of a word w with as few different factors as possible, i. e., with cardinality at most k, for a given k (and with a large size or a small width). We first consider the variant, where we want to find a factorisation with a large size.

Theorem 7. *The problem* MaxRF-s *can be solved in time* $\mathcal{O}(k \times |w|^{2k+3})$.

Proof. Let (w, α, k) be an instance of MaxRF-s. Let $F_w = \{u \mid u \text{ is a factor of } w\}$. For every $F \subseteq F_w$ with $|F| \leq k$, we run the algorithm for the problem FC defined in the proof of Theorem 4 on input (w, F). There exists an ℓ, $\alpha \leq \ell \leq |w|$, with $T[|w|, \ell] = 1$ if and only if there is a factorisation p of w with $\mathsf{s}(p) \geq \alpha$, $\mathsf{sf}(p) \subseteq F$ and, since $|F| \leq k$, also $\mathsf{c}(p) \leq k$. To carry out this procedure, we have to enumerate all subsets $F \subseteq F_w$ with $|F| \leq k$. Since $|F_w| \leq |w|^2$,

for every ℓ, $1 \leq \ell \leq k$, there are at most $|F_w|^\ell \leq |w|^{2\ell}$ subsets $F \subseteq F_w$ with $|F| = \ell$. Thus, there are $\sum_{i=1}^{k} |w|^{2i} \leq 2 \times |w|^{2k}$ subsets to investigate. For each subset F, we run the algorithm of the proof of Theorem 4 in time $\mathcal{O}(|F| \times |w|^2)$, and check for every ℓ, $\alpha \leq \ell \leq |w|$, whether or not $T[|w|, \ell] = 1$, which requires time $\mathcal{O}(|w|)$. Hence, the total running time of this procedure is $\mathcal{O}(|w|^{2k} \times k \times |w|^3) = \mathcal{O}(k \times |w|^{2k+3})$. □

From Theorem 7 we can conclude with moderate effort that MaxRF-s can also be solved in time that is exponential only in $|\Sigma|$ or α.

Theorem 8. *Let Σ be an alphabet.*

- *The problem* MaxRF-s_Σ *can be solved in time* $\mathcal{O}(|\Sigma|^2 \times |w|^{2|\Sigma|+1})$.
- *The problem* MaxRF-s *can be solved in time* $\mathcal{O}(\alpha^2 \times |w|^{2\,\alpha+1})$.

Proof. Let (w, α, k) be an instance of MaxRF-s_Σ with $w = b_1 b_2 \cdots b_{|w|}$, $b_i \in \Sigma$, $1 \leq i \leq |w|$. If $|\Sigma| \leq k$, then the factorisation $p = (b_1, b_2, \ldots, b_{|w|})$ satisfies $\mathsf{s}(p) = |w| \geq \alpha$ and $\mathsf{c}(p) = |\Sigma| \leq k$. If, on the other hand, $k < |\Sigma|$, then we can solve MaxRF-s_Σ with the algorithm of the proof of Theorem 7 in time $\mathcal{O}(k^2 \times |w|^{2k+3}) = \mathcal{O}((|\Sigma| - 1)^2 \times |w|^{2(|\Sigma|-1)+3}) = \mathcal{O}(|\Sigma|^2 \times |w|^{2|\Sigma|+1})$.

Analogously, if $\alpha \leq k$, then any factorisation p of w of size α satisfies $\mathsf{c}(p) \leq \alpha \leq k$; thus, (w, α, k) is a positive instance of MaxRF-s. If, on the other hand, $k < \alpha$, then solving MaxRF-s with the algorithm of the proof of Theorem 7 leads to a running time of $\mathcal{O}((\alpha - 1)^2 \times |w|^{2(\alpha-1)+3}) = \mathcal{O}(\alpha^2 \times |w|^{2\,\alpha+1})$. □

Theorems 7 and 8 show that MaxRF-s can be solved in polynomial-time, as long as at least one of k, $|\Sigma|$ or α are bounded by a constant. However, the probably most interesting question, which, unfortunately, is still open is whether the general version of MaxRF-s can also be solved in polynomial-time.

Open Problem 2. Is MaxRF-s NP-complete?

We now turn to the problem MinRF-w, i. e., computing repetitive factorisations with a small width. In an analogous way as done in the proof of Theorem 7, we can show that MinRF-w can be solved in time exponential only in k, too. The only difference is that instead of running the algorithm of Theorem 4 for every subset of the set of all factors of w, it is sufficient to only consider all subsets of the set of all factors of w that have a length of at most β.

Theorem 9. MinRF-w *can be solved in time* $\mathcal{O}(k^2 \times \beta^k \times |w|^{k+3})$.

Proof. We solve MinRF-w on an instance (w, β, k) in an analogous way as we solved MaxRF-s in the proof of Theorem 7. More precisely, instead of the set of all factors of w, we use the set $F_\beta = \{u \mid u \text{ is a factor of } w, |w| \leq \beta\}$. Then, for all its subsets of size at most k, we run the algorithm defined in the proof of Theorem 4 and, since we do not have a lower bound on the size of the factorisation, we have to check for all ℓ, $1 \leq \ell \leq |w|$, whether $T[|w|, \ell] = 1$.

Since $|F_\beta| \leq \sum_{i=1}^{\beta} |w| - (i-1) \leq \beta \times |w|$, for every ℓ, $1 \leq \ell \leq k$, there are at most $|F_\beta|^\ell \leq \beta^\ell \times |w|^\ell$ subsets $F \subseteq F_\beta$ with $|F| = \ell$. Consequently, there are

$\sum_{i=1}^{k} \beta^i \times |w|^i \leq k \times \beta^k \times |w|^k$ subsets to investigate. In the same way as in the proof of Theorem 7, this leads to a total running time of $\mathcal{O}(k \times \beta^k \times |w|^k \times k \times |w|^2 \times |w|) = \mathcal{O}(k^2 \times \beta^k \times |w|^{k+3})$. □

In a similar way as the first part of Theorem 8 follows from Theorem 7, i. e., by bounding k in terms of $|\Sigma|$, we can conclude from Theorem 9 the next result.

Theorem 10. *Let Σ be an alphabet. Then the problem* MinRF-w_Σ *can be solved in time $\mathcal{O}(|\Sigma|^2 \times \beta^{(|\Sigma|-1)} \times |w|^{|\Sigma|+2})$.*

Proof. Let (w, β, k) be an instance of MinRF-w_Σ with $w = b_1 b_2 \cdots b_{|w|}$, $b_i \in \Sigma$, $1 \leq i \leq |w|$. If $|\Sigma| \leq k$, then the factorisation $p = (b_1, b_2, \ldots, b_{|w|})$ satisfies $\mathsf{w}(p) = 1 \leq \beta$ and $\mathsf{c}(p) = |\Sigma| \leq k$. If, on the other hand, $k < |\Sigma|$, then we can solve MinRF-w_Σ with the algorithm of the proof of Theorem 9 in time $\mathcal{O}(k^2 \times \beta^k \times |w|^{k+3}) = \mathcal{O}((|\Sigma|-1)^2 \times \beta^{(|\Sigma|-1)} \times |w|^{(|\Sigma|-1)+3}) = \mathcal{O}(|\Sigma|^2 \times \beta^{(|\Sigma|-1)} \times |w|^{|\Sigma|+2})$. □

While for problem MaxRF-s it was also possible to bound k in terms of α, for MinRF-w, we can only observe that (w, β, k) must be a positive instance if $k \geq \lceil \frac{|w|}{\beta} \rceil$, but in case $k < \lceil \frac{|w|}{\beta} \rceil$, the algorithm of the proof of Theorem 9 has a running time exponential in $|w|$ and it does not seem possible to solely bound k in terms of β. We now justify this intuition by showing that MinRF-w is NP-complete, even if $\beta = 2$.

First, we recall the following well-known problem, from which we shall conduct a reduction to MinRF-w.

Hitting Set (HS)
Instance: $U = \{x_1, \ldots, x_\ell\}$, $S_1, \ldots, S_n \subseteq U$ and $q \in \mathbb{N}$.
Question: Does there exist $T \subseteq U$ with $|T| \leq q$ and $T \cap S_i \neq \emptyset$, $1 \leq i \leq n$?

A set $T \subseteq U$ with the property $T \cap S_i \neq \emptyset$ is called a *hitting set*, since it hits each of the input sets S_i. We now define a reduction from HS to MinRF-w with $\beta = 2$. To this end, let $(U, S_1, \ldots, S_n, q)$ be an instance of HS. We assume that, for every i, j, $1 \leq i < j \leq n$, $|S_i| = |S_j| = r$. This is not a loss of generality, since HS reduces to the variant where all sets S_i have the same cardinality r by adding $r - |S_i|$ new elements to every S_i. For the sake of concreteness, we assume $S_i = \{y_{i,1}, y_{i,2}, \ldots, y_{i,r}\}$, $1 \leq i \leq n$. We define an alphabet $\Sigma = U \cup \{\star_{i,j} \mid 1 \leq i \leq n, 1 \leq j \leq r-1\} \cup \{\diamond\}$ and a word

$$w = \diamond\diamond\, v_1 \diamond v_2 \diamond \cdots \diamond v_n \diamond, \text{ with}$$
$$v_i = y_{i,1} \star_{i,1} y_{i,2} \star_{i,2} \cdots \star_{i,r-1} y_{i,r}, \text{ for every } i, 1 \leq i \leq n\,.$$

We have to formally prove the correctness of this reduction. The following lemma states that a hitting set of size q translates into a factorisation of w with a width of 2 and a cardinality of at most $n(r-1)+q+1$. Intuitively, this is done by grouping each $y_{i,j}$ with a $\star$-neighbour, except the ones that are elements of the hitting set, which are left as individual factors of size 1.

Lemma 11. *If there exists a set $T \subseteq U$ with $|T| \leq q$ and $T \cap S_i \neq \emptyset$, $1 \leq i \leq n$, then w has a factorisation p with $\mathsf{w}(p) \leq 2$ and $\mathsf{c}(p) \leq n(r-1)+q+1$.*

Proof. We assume that there exists a set $T \subseteq U$ with $|T| \leq q$ and $T \cap S_i \neq \emptyset$, $1 \leq i \leq n$. We now construct a factorisation p of w with the desired properties. We let every single occurrence of $\diamond$ be a factor of p; thus, it only remains to split every v_i, $1 \leq i \leq n$, into factors of size at most 2, which is done as follows. For every i, $1 \leq i \leq n$, let j_i, $1 \leq j_i \leq r$, be arbitrarily chosen such that $y_{i,j_i} \in T$ (since $T \cap S_i \neq \emptyset$, $1 \leq i \leq n$, such j_i exist). Then, for every i, $1 \leq i \leq n$, we factorise v_i into

$$y_{i,1} \star_{i,1} \mid y_{i,2} \star_{i,2} \mid \cdots \mid y_{i,j_i-1} \star_{i,j_i-1} \mid y_{i,j_i} \mid \star_{i,j_i}\, y_{i,j_i+1} \mid \cdots \mid \star_{i,r-1}\, y_{i,r}\,.$$

Obviously, this results in a factorisation p of w with $\mathsf{w}(p) \leq 2$. Furthermore, $\mathsf{sf}(p)$ contains the factor $\diamond$, at most $|T|$ factors x with $x \in T$ and, for every i and j with $1 \leq i \leq n$, $1 \leq j \leq r-1$, a distinct factor of length 2 that contains the symbol $\star_{i,j}$ (the distinctness of these factors follows from the fact that each symbol $\star_{i,j}$ has only one occurrence in w). This implies that $\mathsf{c}(p) \leq 1 + |T| + n(r-1) \leq 1 + q + n(r-1)$. □

On the other hand, if w has a factorisation p with $\mathsf{w}(p) \leq 2$ and $\mathsf{c}(p) \leq n(r-1)+q+1$, then we can show that it can be turned into one that is well-formed (i. e., it has the structure of the factorisation constructed in the proof of Lemma 11), from which a hitting set of size at most q can be obtained.

Lemma 12. *If w has a factorisation p with $\mathsf{w}(p) \leq 2$ and $\mathsf{c}(p) \leq n(r-1)+q+1$, then there exists a set $T \subseteq U$ with $|T| \leq q$ and $T \cap S_i \neq \emptyset$, $1 \leq i \leq n$.*

Proof. We assume that there exists a factorisation p of w with $\mathsf{w}(p) \leq 2$ and $\mathsf{c}(p) \leq 1+q+n(r-1)$. We now modify p step by step such that every modification maintains $\mathsf{w}(p) \leq 2$ and $\mathsf{c}(p) \leq 1 + q + n(r-1)$. Since w starts with $\diamond\diamond$ and $\mathsf{w}(p) \leq 2$, we can conclude that $\diamond$ or $\diamond\diamond$ is a factor of p. If $\diamond\diamond$ is a factor of p, then we can split it into $\diamond \mid \diamond$ without increasing $\mathsf{c}(p)$, since the factor $\diamond\diamond$ is then not a factor of p anymore (since it has no other occurrences in w) and we get at most $\diamond$ as a new factor. Then, every factor of p that contains the symbol $\diamond$ is either the factor $\diamond$ or of the form $x\diamond$ or $\diamond x$ for some $x \in U$ (note that $x = \star_{i,j}$ is not possible). If, for such a factor $x\diamond$ or $\diamond x$, we split all its occurrences into two individual factors x and $\diamond$, then we may produce the new factor x (recall that $\diamond$ is already a factor), but we also necessarily lose $x\diamond$ as a factor in p; thus, $\mathsf{c}(p)$ does not increase. If we apply this modification with respect to all $x \in U$ and all factors $x\diamond$ and $\diamond x$, then we obtain a factorisation in which every single occurrence of the symbol $\diamond$ in w is also a factor of p, $\mathsf{w}(p) \leq 2$ and $\mathsf{c}(p) \leq 1 + q + n(r-1)$, i. e., the factorisation has the structure

$$\diamond \mid \diamond \mid v_1 \mid \diamond \mid v_2 \mid \diamond \mid \cdots \mid \diamond \mid v_n \mid \diamond\,,$$

where the v_i are further factorised into factors of size at most 2.

Since, for every i, $1 \leq i \leq n$, v_i has odd length, the factorisation of v_i (according to p) must contain at least one factor of length 1. Furthermore, if

this factor is of the form $\star_{i,j}$, then the parts to the left and to the right of $\star_{i,j}$ have both odd length as well; thus, using the same argument, they have at least one factor of length 1. Inductively, we can conclude that there is a factor of the form y_{i,j_i}, for some j_i, $1 \leq j_i \leq r$. Now if we group the part to the left and the part to the right of y_{i,j_i} (which both have even length) into factors of size 2, then this does not increase $\mathsf{c}(p)$, since each of these factors contains exactly one of the symbols $\star_{i,j}$, which have only one occurrence in w. Consequently, after these modifications, every v_i has the structure

$$y_{i,1} \star_{i,1} | y_{i,2} \star_{i,2} | \cdots | y_{i,j_i-1} \star_{i,j_i-1} | y_{i,j_i} | \star_{i,j_i} y_{i,j_i+1} | \cdots | \star_{i,r-1} y_{i,r} ,$$

for some j_i, $1 \leq j_i \leq r$. We can now define $T = \{y_{i,j_i} \mid 1 \leq i \leq n\}$, which is obviously a hitting set. Furthermore, the set of factors $\mathsf{sf}(p)$ of p contains the factor $\diamond$, all $n(r-1)$ factors containing a symbol $\star_{i,j}$ and, for every $x \in T$, the factor x. Thus, $\mathsf{c}(p) = 1+n(r-1)+|T|$. By assumption, $\mathsf{c}(p) \leq 1+n(r-1)+q$, which implies $|T| \leq q$. Hence, T is a hitting set of size at most q. □

We note that the MinRF-w instance $(w, 2, n(r-1)+q+1)$ can be constructed from the HS instance $(U, S_1, \ldots, S_n, q)$ in polynomial-time and that MinRF-w is in NP (we can guess and verify a factorisation). Hence, from the NP-completeness of HS (see Garey and Johnson [7]) and from Lemmas 11 and 12, we can conclude the following:

Theorem 13. *The problem* MinRF-w *is* NP*-complete even if* $\beta \leq 2$.

6. Conclusions

In this section, we interpret and discuss our results in more detail and we also point out the most relevant open problems. More precisely, we shall now use the algorithmic and hardness results presented in Sections 4 and 5 in order to categorise the investigated problems in terms of parameterised complexity. This will provide us with a more systematic point of view regarding the hardness of computing equality-free and repetitive factorisations.[1] The results with respect to equality-free factorisations are summarised in Table 1, while the results about repetitive factorisations are summarised in Table 2.[2]

We first recall that the fixed-parameter tractability (of all the considered problems) with respect to the parameter $|w|$ follows trivially (see Proposition 1) and is therefore not of much interest. Furthermore, we have also briefly investigated the problem EFFC, which is not in XP with respect to the parameter $|\Sigma|$

[1] This kind of complexity analysis of NP-hard problems, which considers several different parameters and, by applying the framework of parameterised complexity, tries to scrutinise their impact on the hardness, is also called *multivariate analysis* (*multivariate algorithmics*) or *multi-parameter analysis* in the literature.

[2] In these tables, an entry **p** indicates that the label of the corresponding column is treated as a parameter, while an integer entry means that the non-membership to XP is due to the NP-hardness of the problem variant, where the parameter is bounded by the given integer.

Problem	$\lvert\Sigma\rvert$	α/β	Result	Ref.
MinEFF-w	**p**	**p**	$\in$ FPT	Thm. 5
	2	–	$\notin$ XP	[3]
	–	**2**	$\notin$ XP	[3]
MaxEFF-s	–	–	NP-h	[4]
	p	–	Open	–
	–	**p**	$\in$ FPT	Thm. 6

Table 1: Problems MinEFF-w and MaxEFF-s.

Problem	$\lvert\Sigma\rvert$	α/β	k	Result	Ref.
MaxRF-s	–	–	–	Open	–
	p	–	–	$\in$ XP	Thm. 8
	–	**p**	–	$\in$ XP	Thm. 8
	–	–	**p**	$\in$ XP	Thm. 7
MinRF-w	–	**2**	–	$\notin$ XP	Thm. 13
	p	–	–	$\in$ XP	Thm. 10
	–	–	**p**	$\in$ XP	Thm. 9

Table 2: Problems MaxRF-s and MinRF-w.

(see Theorem 2), but fixed-parameter tractable with respect to the parameter $|F|$ (see Theorem 3).

With respect to computing equality-free factorisations with small width, due to the results by Condon et al. [3], it was already known that MinEFF-w is not in XP with respect to β or Σ, but Theorem 5 shows the fixed-parameter tractability if both β or Σ are parameters. It is interesting to note that if instead of a small width we require a large size, the problem, which generally is also NP-complete (see [4]), becomes easier, i. e., fixed-parameter tractable if parameterised only by α (see Theorem 6). However, it is still open whether MaxEFF-s is fixed-parameter tractability, or at least in XP, with respect to the remaining parameter $|\Sigma|$.

With respect to repetitive factorisations, we know more about computing factorisations with a small width, too, and also this seems to be the harder variant of the problem. More precisely, MinRF-w is in XP with respect to parameter k and $|\Sigma|$ (see Theorems 9 and 10, respectively), but not in XP with respect to β (Theorem 13), while, on the other hand, MaxRF-s is in XP with respect to k, α or $|\Sigma|$ (see Theorems 7 and 8). However, we wish to emphasise that the XP membership results of MaxRF-s become obsolete if MaxRF-s is not NP-complete, which is still open.

While above we compared, for computing equality-free factorisations and computing repetitive factorisations separately, the width variant with the size variant, we can as well compare the problem of computing equality-free factorisations with the problem of computing repetitive factorisations (for the width

variant and the size variant separately). In this regards, the old and new results point out that computing factorisations with a small width is harder if we are looking for equality-free factorisations, since repetitive factorisations with a small width can be computed in polynomial-time for constant alphabets, while this seems impossible for equality-free factorisations. For the size variant, due to the cases left open, we cannot conclude such a statement.

It is still open whether or not the XP-membership results (with respect to MaxRF-s and MinRF-w) can be improved to fixed-parameter tractability results. Since for these cases XP-membership is known, in order to show fixed-parameter intractability, we need more sophisticated concepts of parameterised complexity, e. g., the hardness for one of the classes of the W-hierarchy (see Flum and Grohe [5] for more details).

In the field of algorithmics and combinatorics on words, important properties of strings are often related to special types of factorisations (e. g., Lempel-Ziv factorisations, Lyndon factorisations, etc.). Therefore, purely combinatorial questions about string factorisations are often of interest. In this regard, a trivial and expected combinatorial property of equality-free factorisations is mentioned in Observation 1, i. e., if a string w has an equality-free factorisation of maximal size α, then it necessarily has equality-free factorisations of sizes $1, 2, \ldots, \alpha$. However, the analogous question with respect to the width seems more difficult to answer: if a string w has an equality-free factorisation of minimal width β, does it always have equality-free factorisations of widths $\beta, \beta+1, \ldots, |w|$? It seems very unlikely that, for some word w and $\beta_1 < \beta_2 < \beta_3$, w has equality-free factorisations with width β_1 and β_3, but no equality-free factorisation with width β_2; however, proving this is not straightforward, since a procedure that increases the width by exactly one, while preserving the equality-freeness is not obvious.

Acknowledgements

The author wishes to thank the anonymous referees of this paper for pointing out simpler fpt-algorithms for Theorems 3 and 5 that also achieve a better running time compared to the ones presented in the conference version [12].

References

[1] L. Bulteau, F. Hüffner, C. Komusiewicz, and R. Niedermeier. Multivariate algorithmics for NP-hard string problems. *EATCS Bulletin*, 114:31–73, 2014.

[2] A. Condon, J. Maňuch, and C. Thachuk. Complexity of a collision-aware string partition problem and its relation to oligo design for gene synthesis. In *Proc. 14th Annual International Computing and Combinatorics Conference, COCOON 2008*, volume 5092 of *LNCS*, pages 265–275, 2008.

[3] A. Condon, J. Maňuch, and C. Thachuk. The complexity of string partitioning. *Journal of Discrete Algorithms*, 32:24–43, 2015.

[4] H. Fernau, F. Manea, R. Mercaş, and M.L. Schmid. Pattern matching with variables: Fast algorithms and new hardness results. In *Proc. 32nd Symposium on Theoretical Aspects of Computer Science, STACS 2015*, volume 30 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 302–315, 2015.

[5] J. Flum and M. Grohe. *Parameterized Complexity Theory*. Springer-Verlag New York, Inc., 2006.

[6] T. Gagie, S. Inenaga, J. Karkkainen, D. Kempa, M. Piatkowski, S. J. Puglisi, and S. Sugimoto. Diverse palindromic factorization is NP-complete. Technical Report 1503.04045, 2015. `http://arxiv.org/abs/1503.04045`.

[7] M. R. Garey and D. S. Johnson. *Computers And Intractability*. W. H. Freeman and Company, 1979.

[8] H. Jiang, B. Su, M. Xiao, Y. Xu, F. Zhong, and B. Zhu. On the exact block cover problem. In *Proc. 10th International Conference on Algorithmic Aspects in Information and Management, AAIM 2014*, volume 8546 of *LNCS*, pages 13–22, 2014.

[9] D E. Knuth, J. H. Morris, and V. R. Pratt. Fast pattern matching in strings. *Communications of the ACM*, 6(2):323–350, 1977.

[10] M. Lothaire. *Combinatorics on Words*, chapter 9. Cambridge University Press, Cambridge, New York, 1997.

[11] H.C. Papadimitriou. *Computational Complexity*. Addison-Wesley, Reading, MA, 1995.

[12] M. L. Schmid. Computing equality-free string factorisations. In *Proc. 11th Conference on Computability in Europe, CiE 2015*, volume 9136 of *LNCS*, pages 313–323, 2015.