String Factorization via Prefix Free Families

Matan Kraus
Bar-Ilan University, Ramat-Gan, Israel

Moshe Lewenstein
Bar-Ilan University, Ramat-Gan, Israel

Alexandru Popa
Faculty of Mathematics and Computer Science, University of Bucharest, Romania

Ely Porat
Bar-Ilan University, Ramat-Gan, Israel

Yonathan Sadia
Bar-Ilan University, Ramat-Gan, Israel

Abstract

A factorization of a string S is a partition of w into substrings $u_1, \ldots, u_k$ such that $S = u_1 u_2 \cdots u_k$. Such a partition is called equality-free if no two factors are equal: $u_i \neq u_j, \forall i, j$ with $i \neq j$. The *maximum equality-free factorization problem* is to find for a given string S, the largest integer k for which S admits an equality-free factorization with k factors.

Equality-free factorizations have lately received attention because of their applications in DNA self-assembly. The best approximation algorithm known for the problem is the natural greedy algorithm, that chooses iteratively from left to right the shortest factor that does not appear before. This algorithm has a $\sqrt{n}$ approximation ratio (SOFSEM 2020) and it is an open problem whether there is a better solution.

Our main result is to show that the natural greedy algorithm is a $\Theta(n^{1/4})$ approximation algorithm for the maximum equality-free factorization problem. Thus, we disprove one of the conjectures of Mincu and Popa (SOFSEM 2020) according to which the greedy algorithm is a $\Theta(\sqrt{n})$ approximation.

The most challenging part of the proof is to show that the greedy algorithm is an $O(n^{1/4})$ approximation. We obtain this algorithm via *prefix free* factor families, i.e. a set of non-overlapping factors of the string which are pairwise non-prefixes of each other. In the paper we show the relation between prefix free factor families and the maximum equality-free factorization. Moreover, as a byproduct we present another approximation algorithm that achieves an approximation ratio of $O(n^{1/4})$ that we believe is of independent interest and may lead to improved algorithms. We then show that the natural greedy algorithm has an approximation ratio that is $\Omega(n^{1/4})$ via a clever analysis which shows that the greedy algorithm is $\Theta(n^{1/4})$ for the maximum equality-free factorization problem.

2012 ACM Subject Classification Theory of computation → Approximation algorithms analysis

Keywords and phrases string factorization, NP-hard problem, approximation algorithm

Digital Object Identifier 10.4230/LIPIcs.CPM.2023.19

Funding This work was supported by a grant of the Ministry of Research, Innovation and Digitization, CNCS - UEFISCDI, project number PN-III-P1-1.1-TE-2021-0253, within PNCDI III. Matan kraus, Ely Porat and Yonathan Sadia were supported by ISF grants no. 1278/16 and 1926/19, by a BSF grant 2018364, and by an ERC grant MPM under the EU's Horizon 2020 Research and Innovation Programme (grant no. 683064).

1 Introduction

A factorization of a string S is a partition of S into substrings $u_1, u_2, \ldots, u_k$ such that $S = u_1 u_2 \cdots u_k$. Factorizations are central objects of study in stringology, a famous example being the Lempel-Ziv algorithm [14]. String factorizations have many other applications as we show next. For instance, finding an occurrence of a string v in a text T can be formulated as T admitting a factorization $T = uvw$. Then, a string v is a prefix of another string T if $T = vw$ and it is a suffix of T if $T = uv$. Moreover, many string problems can be seen as string factorization problems [9] such as: Shortest Common Superstring, Longest Common Subsequence and Shortest Common Supersequence, to name a few. Another example of a string factorization problem is the Minimum Common String Partition [6, 7], a problem concerned with identifying factorizations for two strings such that the sequence of factors for one string is the permutation of the other's.

In this paper we focus on the equality-free factorization, a special case of string factorization in which all factors are distinct. The equality-free factorization is a restricted variant of a more famous problem, termed *generalized function matching* which has a long history starting from 1979 (see, e.g., [12] and the references therein for more details). In the generalized function matching the input consists of a text t over an alphabet Σ_1 and a pattern $p = p_1 p_2 \ldots p_m$ over an alphabet Σ_2. The goal is to find an injective function from $f : \Sigma_2 \to \Sigma_1^+$ such that $t = f(p_1)f(p_2)\ldots f(p_m)$. Thus, the maximum equality free factorization problem is a particular case of the generalized function matching in which all the characters of the pattern p are distinct. In turn, generalized function matching is a particular case of string equations, which is a notoriously difficult problem (see, e.g., the JACM paper [13]). In fact, even the version which restricts character-to-character function matching is extremely difficult, see [1], as opposed to the more restricted parameterized matching [2, 3, 10] which is simpler. Thus, maximum equality free factorization is part of family of fundamental problems in stringology.

The maximum equality free factorization problem is also motivated by applications in DNA synthesis [4]. More specifically, it is possible to produce short DNA fragments that will self-assemble into the wanted DNA structure. However, to obtain the desired structure, it is required that no two fragments are identical. Since the fragments must be short, one approach is to split the target DNA sequence into as many distinct pieces as possible.

Previous work

The equality-free factorization problem was first introduced by Condon, Maňuch and Thachuk [4] where it was presented as the *string partitioning problem*. The string partitioning problem asks for a factorization into distinct factors such that each factor is at most of a certain length. The problem was studied in a more general setting where the measure of collision between two factors is either equality or one is a prefix/suffix of the other. Condon et al. showed that these variants are $\mathcal{NP}$-complete. More recently, Fernau, Manea, Mercaş and Schmid [5] presented a similar problem that imposes a lower bound on the number of factors instead of an upper bound on factor length. Fernau et al. showed that this variant is also $\mathcal{NP}$-complete. Afterwards, Schmid [9] studied the Fixed-Parameter Tractability of the two problems.

The decision version of the problem, that is, given a string S and an integer k, decide if there exists an equality free factorization of S with at least k factors, is termed MAXEFF-s (this is the notation of Schmid [9] and we decide to use it for the sake of consistency). The optimization version, in which we are given S and the goal is to find an equality free factorization with as many factors as possible, is termed OPTEFF-s. The acronyms for the two problems were introduced in the previous papers [5, 9, 11] and we will use them in our paper for consistency (OPT stands for *optimization*).

Mincu and Popa [11] study OPTEFF-s and another variant named Maximum Gapped Equality Free Factorization (OPTGEFF-s). In the latter it is *not* required that all the characters of the input strings are part of the factorization. That is, the goal is to find an equality free factorization using a maximum number of factors of a substring of the input string. More formally, a *gapped factorization* of string S over alphabet Σ is a tuple $(u_1, u_2, \ldots, u_k)$ such that $S = \alpha_0 u_1 \alpha_1 u_2 \alpha_2 \cdots \alpha_{k-1} u_k \alpha_k$, where $u_i \in \Sigma^+$ are the factors and $\alpha_i \in \Sigma^*$ are the gaps. OPTGEFF-s asks, for a given string S, to find the largest integer k such that S admits a gapped equality-free factorization of size k. In [11] a 2-approximation for the OPTGEFF-s and a $\sqrt{n}$-approximation (where n is the size of the input string) for the OPTEFF-s were shown. Moreover, it was conjectured [11] that the greedy approximation ratio is $\Omega(\sqrt{n})$. Grüttemeier et al. [8] show a randomized algorithm for solving the MAXEFF-s with running time $2^k \cdot k^{O(1)} + O(n)$.

Our results

As mentioned, the best-known approximation algorithm, the greedy algorithm, for the OPTEFF-s has an approximation ratio of $\sqrt{n}$ and it was conjectured that the greedy algorithm has an approximation ratio of $\Theta(\sqrt{n})$. In this paper, we show a better approximation algorithm for OPTEFF-s with ratio $O(n^{1/4})$. We then use this algorithm to show that the greedy algorithm has the same approximation ratio of $O(n^{1/4})$. Hence, this disproves the conjecture from [11] saying that the approximation ratio of the greedy algorithm is $\Omega(\sqrt{n})$.

The challenge is to show that the greedy algorithm has an approximation ratio of $O(n^{1/4})$. To get our approximation ratio we start with an (approximate) prefix free solution for the version with gaps. Then, we use the prefix free property to map the factors of a solution returned by the greedy algorithm to the aforementioned prefix free solution. Moreover, besides the greedy algorithm, we introduce another approximation algorithm for OPTEFF-s with an approximation ratio of $O(n^{1/4})$, that uses some interesting techniques and is of independent interest. We claim that our techniques give some key insights and perhaps open the path for better approximation algorithms for the problem.

Finally, we use a clever analysis to also show that the greedy algorithm cannot have an approximation ratio better than $O(n^{1/4})$ and, hence, the approximation ratio of the greedy algorithm is $\Theta(n^{1/4})$ for the maximum equality-free factorization problem.

2 The prefix-free property

For the OPTGEFF-s problem (the version of factorization with gaps), a 2-approximation algorithm via a reduction from a scheduling problem was shown in [11]. A natural direction for proving an approximation algorithm for OPTEFF-s is to transform a factorization with gaps, obtained from the approximation algorithm of OPTGEFF-s, into a solution without gaps. However, it is difficult to transform a given factorization with gaps into a factorization without gaps with roughly the same number of factors.

In this section we show that it is possible to transform a special case of a factorization with gaps into a factorization without gaps. We introduce the notion of a *prefix-free gapped factorization*, which has an important role in our algorithm and might be of independent interest.

▶ **Definition 1.** *Let $n, k \in \mathbb{N}$, let S be a string of length n and let $F_k = \{S_1, S_2, \ldots, S_k\}$ be a set of non-overlapping factors of S (possibly with gaps). F is a* prefix-free gapped factorization *of S if for all $i \neq j$, S_i is not a prefix of S_j.*

Given a prefix-free gapped factorization F such that $|F| = k$, we prove that there is a transformation of F into a factorization without gaps with the same number of factors k, since each factor S_i can be extended until the next factor S_{i+1} without colliding with another factor S_j.

▶ **Lemma 2.** *Let $n, k \in \mathbb{N}$ and let S be a string of length n with a prefix-free gapped factorization F with $|F| = k$. Then, there is an equality free factorization for S without gaps with at least k factors.*

Proof. Denote $S = T_0 S_1 T_1 S_2 T_2 S_2 \ldots T_{k-1} S_k T_k$. Denote with $R_i = S_i T_i$. Note that for each $i \neq j$, R_i and R_j are not prefixes of each other because their prefixes are S_i and S_j, respectively, which are not prefixes of each other.

Now, consider $S = T_0 R_1 R_2 \ldots R_k$. If, for all i, $T_0 \neq R_i$, then we have an equality free $(k+1)$-factorization. Otherwise, there exists an i such that $T_0 = R_i$. We distinguish two cases.

In the first case, if $i < k$, then we set $Q_i = R_i R_{i+1}$. Thus, Q_i and all other factors R_j are still not prefixes of each other. T_0, which equals R_i, is not a prefix of any other R_j (because it equaled R_i) and is shorter than Q_i. Hence S has a k-factorization $S = R_i R_1 R_2 \ldots R_{i-1} Q_i R_{i+2} \ldots R_k$.

In the second case, if $i = k$, then we set $Q_k = R_{k-1} R_k$ and using a similar argument as above we obtain a k-factorization $S = R_k R_1 R_2 \ldots R_{k-2} Q_k$. ◀

3 An $O(n^{1/4})$-approximation algorithm

In this section we show an algorithm that has an $O(n^{1/4})$-approximation to OPTEFF-s.

Our algorithm (see Algorithm 1) is composed of two algorithms: a greedy algorithm, called Greedy1 (see Algorithm 3), which always yields an $\sqrt{n}$-approximation, and a new algorithm (Algorithm 2) which is described next. Algorithm 1 simply selects the better of the two algorithms and returns it.

Algorithm 1 An $O(n^{1/4})$-approximation algorithm for OPTEFF-s.

```
Input: String $S$.
1 $F \leftarrow Algorithm\ 2(S)$;
2 $G \leftarrow Algorithm\ 3(S)$;
3 if $|G| > |F|$ then
4     return $G$
5 return $F$
```

The basic idea behind Algorithm 2 is to find, for every fixed integer $1 \leq i \leq 2\sqrt{n}$, a greedy equality free gapped factorization of the input string in which every factor has length exactly i. The algorithm chooses from these gapped fixed length factorizations, the factorization with the most factors. Then, due to Lemma 2, we append to each of these factors the following adjacent (possibly empty) gap and we obtain an equality free factorization. See Algorithm 2 for more details.

Algorithm 2 Fixed length greedy factorization.

```
Input: String $S$.
1  $F \leftarrow \emptyset$;
2  for $i \leftarrow 1$ to $2\sqrt{n}$ do
3      $j \leftarrow 1$,$G \leftarrow \emptyset$;
4      while $j \leq n - i$ do
5          if $S[j..j+i-1] \notin G$ then
6              $G \leftarrow G \cup \{S[j..j+i-1]\}$;
7              $j \leftarrow j + i - 1$;
8          $j \leftarrow j + 1$;
9      if $|G| > |F|$ then
10         $F \leftarrow G$;
11 Extend each factor $w_i \in F$ until factor $w_{i+1}$ (for the last factor, extend it until the
     end of $S$);
12 return $F$
```

▶ **Lemma 3.** *Algorithm 2 yields an equality-free factorization without gaps.*

Proof. First, in the for loop, at each step, Algorithm 2 adds to G only distinct substrings of S. Then, notice that for every two factors $w_1, w_2 \in G$, it holds that w_1 is not a prefix of w_2, since both w_1 and w_2 have the same length. Therefore, G is a prefix-free gapped factorization, and due to Lemma 2, the factors are extended as in line 11 to have an equality-free factorization without gaps. ◀

Analysis

Here we prove that when the optimal solution, denoted OPT, has "many" factors, Algorithm 2 returns a good approximation of OPT.

Formally, let F be the factorization returned by Algorithm 2. Let α be $n/|OPT|$. Notice that $|OPT| = n/\alpha$. We claim that $|F| = \Omega(n/\alpha^2)$.

We first give an overview of the proof. First, we prove in Lemma 5 that there are at least $n/2\alpha$ short factors (of length at most 2α) in OPT. Then we prove in Lemma 6 that there are at least $\Omega(n/\alpha^2)$ factors of the exact same length in OPT. Next, we prove in Lemma 9 that the factorization F returned in Algorithm 2 is a 2-approximation of optimal fixed length factorization (see Definition 7). Finally, we prove in Lemma 10 that $|F| = \Omega(n/\alpha^2)$.

▶ **Definition 4.** *An x-short factor of S is a factor of length $\leq x$. An x-long factor of S is a factor of length $> x$. When x is clear we will simply call them short factors and long factors.*

▶ **Lemma 5.** *There exist at least $n/2\alpha$ factors in OPT that are 2α-short.*

Proof. Let LF denote the set of 2α-long factors in OPT and SF denote set of the 2α-short factors in OPT. We will use an argument on n, the length of S. Each long factor must be, by definition, of length $\geq 2\alpha+1$. Hence, by length arguments, $|LF|\cdot(2\alpha+1)+|SF|\cdot 1 \leq n$ and, hence, $|LF| \leq n/(2\alpha+1) - |SF|/(2\alpha+1)$. On the other hand, since $|OPT| = n/\alpha$, we have that $|SF| = n/\alpha - |LF|$. Putting these two equations together yields that $|SF| = n/\alpha - |LF| \geq n/\alpha - n/(2\alpha+1) + |SF|/(2\alpha+1)$ and hence, $|SF| - |SF|/(2\alpha+1) \geq n/\alpha - n/(2\alpha+1)$ which in turn yields $2\alpha|SF|/(2\alpha+1) \geq (n\alpha+n)/\alpha(2\alpha+1)$. Hence, $2\alpha^2|SF| \geq n\alpha + n$ and $|SF| \geq n/2\alpha + n/2\alpha^2 \geq n/2\alpha$. ◀

Next, we show that among the short factors, $\Omega(1/\alpha)$ fraction of them actually have exactly the same length.

▶ **Lemma 6.** *There exists an integer $\ell \leq 2\alpha$ such that there are at least $n/4\alpha^2$ short factors in OPT of length exactly ℓ.*

Proof. By Lemma 5, there are at least $n/2\alpha$ short factors in OPT. The average number of factors of each short length, is at least $\frac{n/2\alpha}{2\alpha} = n/4\alpha^2$. By the pigeonhole principle, there exists an integer $\ell \leq 2\alpha$ such that there are at least $n/4\alpha^2$ short factors in OPT of length exactly ℓ. ◀

Next we prove that Algorithm 2 is a constant approximation algorithm to the problem of gapped factorization with fixed lengths.

▶ **Definition 7.** *The Fixed-Length Maximum Gapped Equality-Free Factorization Size (FLOptGEFF-s) problem is defined as follows. For a given string S and an integer r, find the largest integer m, such that S admits a gapped equality-free factorization of size m where all factors are of length r.*

In [11], the problem of OPTGEFF-s is reduced to the Job Interval Selection Problem with k intervals (JISPk, see Theorem 8), which has a 2-approximation algorithm.

▶ **Theorem 8** (restated from [11]). *Given n jobs containing k time intervals each, find the maximum number of intervals that can be selected such that (i) no two intervals intersect and (ii) at most one time interval is selected per job.*

Analogously to [11], FLOptGEFF-s is reducible as well to JISPk, and here we briefly show the reduction.

▶ **Lemma 9.** *Algorithm 2 is a 2-approximation for FLOptGEFF-s.*

Sketch Proof. We construct an instance of JISPk with $O(n)$ jobs from a string S with n characters. For each distinct substring of S with length r, we create a job. For each substring s we add $[a, b)$ as a time interval of s for all occurrences $s = S[a, b]$ in S.

Since JISPk has a 2-approximation algorithm, we have that FLOptGEFF-s has a 2-approximation algorithm as well. Moreover, the algorithm that approximates FLOptGEFF-s(S, r) for some string S and integer r is in fact the inner loop of Algorithm 2, on the iteration where $i = r$. ◀

We are ready to prove a lower bound on the number of factors returned by Algorithm 2.

▶ **Lemma 10.** *Let F be the factorization returned by Algorithm 2. Then, $|F| = \Omega(n/\alpha^2)$.*

Proof. By Lemma 6, there exists ℓ such that there are at least $n/4\alpha^2$ short factors in OPT of length ℓ.

Let S^{ℓ}_{ALG} be the number of factors of length ℓ produced by Algorithm 2, let S^{ℓ}_{GEFF} be FLOptGEFF-s(S, ℓ), and let S^{ℓ}_{OPT} be the number of factors of length ℓ in OPT.

By Lemma 9, there is a polynomial algorithm that is a 2-approximation of the number of occurrences of a factor of length ℓ in OPT. Moreover, the algorithm behind Lemma 9 is in fact the inner loop of Algorithm 2. Notice that $\ell \leq 2\alpha \leq 2n/|OPT| \leq 2\sqrt{n}$, since $|OPT| \geq \sqrt{n}$ and therefore there is an iteration where $i = \ell$. Then,

$$S^{\ell}_{OPT}/2 \leq S^{\ell}_{GEFF}/2 \leq S^{\ell}_{ALG}$$

where the first inequality is due to the definition of FLOptGEFF-s and the second inequality is due to Lemma 9.

Hence, combining Lemma 6 and Lemma 9, for the iteration where $i = \ell$ on line 9, $|G| = S^{\ell}_{ALG} \geq (n/4\alpha^2)/2$. Since the number of factors returned by the algorithm is at least $|G|$ (i.e. $|F| \geq S^{\ell}_{ALG}$), we have that $|F| = \Omega(n/\alpha^2)$. ◀

As stated before, Algorithm 1 is composed by two algorithms, Algorithm 2 and Algorithm 3. Algorithm 3 was introduced in [11] as Greedy1 algorithm. In a nutshell, consider starting "at the left" of the string and adding the next shortest substring (distinct from the already selected factors) to the incumbent factorization at each step of the algorithm. See [11] for details.

Algorithm 3 Greedy1.

```
Input: String $S$.
1 $j \leftarrow 1$,$F \leftarrow \emptyset$;
2 for $i \leftarrow 1$ to $n$ do
3     if $S[j..i] \notin F$ then
4         $F \leftarrow F \cup S[j..i]$;
5         $j \leftarrow i + 1$;
6 Extend the last factor of $F$ until the end of $S$;
7 return $F$
```

It was proven in [11] that Greedy1 yields an equality-free factorization. Moreover, they prove that Greedy1 produces at least $\Omega(\sqrt{n})$ factors.

▶ **Theorem 11.** *Algorithm 1 is an $O(n^{1/4})$-approximation polynomial time algorithm for the OPTEFF-s problem.*

Proof. Combining Greedy1 with Algorithm 2, we have an algorithm that produces at least $\Omega(\max((n/\alpha^2), \sqrt{n}))$ factors. This gives an approximation ratio of $O(\min(\frac{n/\alpha}{n/\alpha^2}, (n/\alpha)/\sqrt{n})) = O(\min(\alpha, \sqrt{n}/\alpha))$, which is maximized at $\alpha = \sqrt{n}/\alpha$, i.e. at $\alpha = n^{1/4}$.

Finally, notice that the both Greedy1 and Algorithm 2 run in polynomial time of at most $O(n^{1.5} \log n)$. ◀

4 The natural greedy algorithm is a $\Theta(n^{1/4})$-approximation

In this section we prove that Greedy1 itself achieves an approximation ratio of $O(n^{1/4})$.

▶ **Lemma 12.** *Greedy1 is a 2-approximation of Algorithm 2.*

Proof. Let S be a string, and let ℓ be a positive integer. Let F_ℓ be a fixed ℓ length gapped factorization on S such that $|F_\ell| =$FLOptGEFF-s(S, ℓ). Let F_G be the factorization that is the output of the Greedy1 algorithm. We show that $|F_G| \geq |F_\ell|/2$.

We map each factor of F_ℓ to a factor of F_G as follows. Let $f \in F_\ell$ be a factor in F_ℓ and let $i \leq j$ be two indices such that $f = S[i, j]$. In F_G, denote the factors that cover $S[i]$ and $S[j]$ as g_i and g_j, respectively. If $g_i \neq g_j$, map f to g_j. If f is a suffix of g_j, then also map f to g_j. Otherwise, f is fully contained in a factor of F_G and f is not a suffix of g_j. Then, it must be the case that there is a factor g_s in F_G such that the suffix of g_s is exactly f, as otherwise Greedy1 would have cut the factor g_j right after index j, but f is not a suffix of g_j. Therefore, map f to g_s (if there are more than one factors with f as a suffix in F_G, map to one of them arbitrarily).

Now, let g be a factor in F_G, and let $\hat{\imath} \leq \hat{\jmath}$ be two indices such that $g = S[\hat{\imath}, \hat{\jmath}]$. We claim that there are at most two factors in F_ℓ that are mapped to g. There is at most one factor in F_ℓ that overlaps $S[\hat{\imath}]$, and therefore mapped to g because of overlapping two factors in F_G. Moreover, since all the factors in F_ℓ have the same size, there is at most one factor in F_ℓ such that the suffix of g is equal to the factor. Therefore, there are at most 2 factors in F_ℓ that are mapped to g. To conclude, there are at most $2 \cdot |F_G|$ factors in F_ℓ, for every ℓ.

Let F_A be the factorization returned by Algorithm 2. There is an ℓ such that $|F_\ell| \geq |F_A|$. Since we proved that $|F_G| \geq |F_\ell|/2$ for every ℓ, we also have that $|F_G| \geq |F_A|/2$. ◀

Combining the above lemma with the Theorem 11, we conclude the following theorem.

▶ **Theorem 13.** *The approximation ratio of Greedy1 is $O(n^{1/4})$.*

Proof. By Lemma 12, Greedy1 is a constant approximation of Algorithm 2 and therefore Greedy1 is also a constant approximation of Algorithm 1 (that simply uses Algorithm 2 and Greedy1 and returns the maximum between them). Since by Theorem 11 Algorithm 1 is an $O(n^{1/4})$-approximation for OPTEFF-s we have that Greedy1 is also an $O(n^{1/4})$-approximation. ◀

5 Tightness of Algorithm 1

In this section we prove that our analysis of Algorithm 1 is actually tight. We show that there is a case where both Greedy1 and Algorithm 2 have an approximation ratio that is at least $\Omega(n^{1/4})$.

Similar to [11], we define a string S as follows. Let n be a square of an even number, i.e. there is an integer k such that $n = (2k)^2$. Let $\Sigma = \{x_1, x_2, \ldots, x_{\sqrt{n}}\}$ be an alphabet. We define *variables* $X_1, X_2, \ldots, X_{\sqrt{n}}$ such that for each variable X_i, define $X_i = x_1 x_2 .. x_i$. The string S is defined as $S = X_1 X_2 .. X_{\sqrt{n}}$. Note that $|S| = \Theta(n)$.

▶ **Lemma 14.** *There exists a factorization of S with $\Omega(n^{3/4})$ factors.*

Proof. We first factorize S into $\Omega(n^{3/4})$ factors with gaps, and afterwards we get rid of the gaps. We factorize the variables $X_1 .. X_{\sqrt{n}/2-1}$ using only one factor. Then, the variable $X_{\sqrt{n}/2}$ is factorized into $x_1; x_2; ..; x_{\sqrt{n}/2}$, for a total of $\sqrt{n}/2$ factors. At least $\sqrt{n}/2 - 1$ factors are produced by the variables $X_{\sqrt{n}/2+1} X_{\sqrt{n}/2+2}$ as follows. The variable $X_{\sqrt{n}/2+1}$ is factorized into $x_1 x_2; x_3 x_4; \ldots$, for a total of $\lfloor |X_{\sqrt{n}/2+1}|/2 \rfloor$ factors. The variable $X_{\sqrt{n}/2+2}$ is factorized into $x_2 x_3; x_4 x_5; \ldots$, also for a total of at least $\lfloor (|X_{\sqrt{n}/2+1}|)/2 \rfloor$ factors.

In general, at each iteration i, the algorithm produces factors of length i using i variables and i offsets. Each variable is of length at least $\sqrt{n}/2$, therefore at least $i \cdot \lfloor (\sqrt{n}/2)/i \rfloor \geq \sqrt{n}/2 - i$ factors are produced by i variables. For each iteration i, the rth variable X_j of iteration i produces factors of length i starting at index r with respect to the beginning of X_j. This procedure produces an equality free factorization with gaps.

There is a constant $c > 0$ such that there are $c\sqrt{\sqrt{n}/2} = cn^{1/4}$ iterations to the process. Therefore, at least

$$\sum_{i=1}^{cn^{1/4}} \sqrt{n}/2 - i \geq cn^{3/4}/2 - c^2\sqrt{n} = \Omega(n^{3/4})$$

factors are produced in this process.

We are left with handling the gaps. Notice that there are two reasons for a gap to occur. First, (1) on iteration i and variable X_j, we produce $\lfloor |X_j|/i \rfloor$ factors, and $|X_j| - i \lfloor |X_j|/i \rfloor > 0$, so we have a gap at the end of X_j. Second, (2) on iteration i and the rth variable of the iteration X_j, when X_j is not the first variable of iteration i ($r \neq 1$), the factorization of X_j does not start from x_1 but from x_r.

For gaps of type 1, let X_j be a variable that has a gap at the end of X_j. Then, if $j \neq \sqrt{n}$, we extend the first factor of X_{j+1} backwards to close the gap. This extended factor is unique since there is only one instance of $x_j x_1$ in S. If $j = \sqrt{n}$ and we are in the last variable, we extend the last factor of X_j. This extended factor is unique since there is only one instance of this length in S.

For gaps of type 2, let X_j be a variable that has a gap at the beginning of X_j. Then, we extend the last factor of X_{j-1} forward to close the gap. This extended factor is unique since there is only one instance of $x_{j-1}x_1$ in S.

For gaps with both types 1 and 2, we just handle them as gaps with type 1. ◀

On string S, Greedy1 produces $\Theta(\sqrt{n})$ factors. Hence, and by Lemma 14, we have the following corollary.

▶ **Corollary 15.** *The approximation ratio of Greedy1 is $\Omega(n^{1/4})$.*

On the string S described above, Algorithm 2 produces $O(\sqrt{n})$ factors. To see this, let l be some length that is being observed in line 2 of Algorithm 2. There are (at most) $\sqrt{n}$ x_1's in string S. Therefore there are at most $\sqrt{n}$ factors containing x_1. On the other hand, every factor that does not contain x_1, must start in a unique character (since before the extension, every factor is of length exactly l). There are (at most) $\sqrt{n}$ unique characters in S. Therefore, there are at most $\sqrt{n}$ factors *not* containing x_1.

Hence, and by Lemma 14, we have the following corollary.

▶ **Corollary 16.** *The approximation ratio of Algorithm 2 is $\Omega(n^{1/4})$.*

Finally, since both lower bounds were based on the same case of string S, we have the following corollary.

▶ **Corollary 17.** *The approximation ratio of Algorithm 1 is $\Omega(n^{1/4})$.*

6 Conclusions and future work

In this paper we disproved one of the conjectures of Mincu and Popa [11] and show that the natural greedy algorithm for the OPTEFF-s problem has a $\Theta(n^{1/4})$-approximation factor. It is, of course, a natural open question to improve the approximation ratio for OPTEFF-s using a different algorithm than the greedy. We believe that the key in succeeding to obtain such an algorithm is finding a better approximation algorithm for the case when the number of factors in an optimal solution is relatively small. Thus, the ideas introduced in Section 3, where we present an alternative $O(n^{1/4})$ approximation algorithm, represent a promising direction.

References

1 Amihood Amir, Yonatan Aumann, Moshe Lewenstein, and Ely Porat. Function matching. *SIAM Journal on Computing*, 35(5):1007–1022, 2006.

2 Brenda S. Baker. Parameterized pattern matching: Algorithms and applications. *Journal of Computer and System Sciences*, 52(1):28–42, 1996.

3 Richard Cole, Carmit Hazay, Moshe Lewenstein, and Dekel Tsur. Two-dimensional parameterized matching. *ACM Transactions on Algorithms*, 11(2):12:1–12:30, 2014.

4 Anne Condon, Ján Maňuch, and Chris Thachuk. The complexity of string partitioning. *Journal of Discrete Algorithms*, 32:24–43, 2015.

5 Henning Fernau, Florin Manea, Robert Mercas, and Markus L. Schmid. Pattern matching with variables: Fast algorithms and new hardness results. In *32nd International Symposium on Theoretical Aspects of Computer Science, 2015, March 4-7, 2015, Garching, Germany*, pages 302–315, 2015.

6 Avraham Goldstein, Petr Kolman, and Jie Zheng. Minimum common string partition problem: Hardness and approximations. *The Electronic Journal of Combinatorics*, 12, 2005.

7 Isaac Goldstein and Moshe Lewenstein. Quick greedy computation for minimum common string partition. *Theoretical Computer Science*, 542:98–107, 2014.

8 Niels Grüttemeier, Christian Komusiewicz, Nils Morawietz, and Frank Sommer. String factorizations under various collision constraints. In *31st Annual Symposium on Combinatorial Pattern Matching (CPM 2020)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2020.

9 Markus L. Schmid. Computing equality-free and repetitive string factorisations. *Theoretical Computer Science*, 618:42–51, 2016.

10 Moshe Lewenstein. Parameterized pattern matching. In *Encyclopedia of Algorithms*, pages 1525–1530. Springer, 2016.

11 Radu Stefan Mincu and Alexandru Popa. The maximum equality-free string factorization problem: Gaps vs. no gaps. In *International Conference on Current Trends in Theory and Practice of Informatics*, pages 531–543. Springer, 2020.

12 Sebastian Ordyniak and Alexandru Popa. A parameterized study of maximum generalized pattern matching problems. *Algorithmica*, 75(1):1–26, 2016.

13 Wojciech Plandowski. Satisfiability of word equations with constants is in PSPACE. *Journal of the ACM (JACM)*, 51(3):483–496, 2004.

14 Jacob Ziv and Abraham Lempel. Compression of individual sequences via variable-rate coding. *IEEE Transactions on Information Theory*, 24:530–536, 1978.