Finding an Optimal Alphabet Ordering for Lyndon Factorization is Hard

Daniel Gibney
Department of Computer Science, University of Central Florida, USA
https://www.cs.ucf.edu/~dgibney/
daniel.gibney@ucf.edu

Sharma V. Thankachan
Department of Computer Science, University of Central Florida, USA
http://www.cs.ucf.edu/~sharma/
sharma.thankachan@ucf.edu

Abstract

This work establishes several strong hardness results on the problem of finding an ordering on a string's alphabet that either minimizes or maximizes the number of factors in that string's Lyndon factorization. In doing so, we demonstrate that these ordering problems are sufficiently complex to model a wide variety of ordering constraint satisfaction problems (OCSPs). Based on this, we prove that (i) the decision versions of both the minimization and maximization problems are NP-complete, (ii) for both the minimization and maximization problems there does not exist a constant approximation algorithm running in polynomial time under the Unique Game Conjecture and (iii) there does not exist an algorithm to solve the minimization problem in time $poly(|T|) \cdot 2^{o(\sigma \log \sigma)}$ for a string T over an alphabet of size σ under the Exponential Time Hypothesis (essentially the brute force approach of trying every alphabet order is hard to improve significantly).

2012 ACM Subject Classification

Keywords and phrases Lyndon Factorization, String Algorithms, Burrows-Wheeler Transform

Digital Object Identifier 10.4230/LIPIcs.CVIT.2016.23

42nd Conference on Very Important Topics (CVIT 2016).
Editors: John Q. Open and Joan R. Access; Article No. 23; pp. 23:1–23:15
Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

1 Introduction

A Lyndon word is a string that is lexicographically strictly smallest among all of its cyclic shifts. Letting $\circ$ denote concatenation, the Lyndon factorization of a string T is the partitioning of T into Lyndon words T_1, T_2, ..., T_f that are lexicographically non-increasing and $T = T_1 \circ T_2 \circ \ldots \circ T_f$. For example, the Lyndon factorization of $0, 1, 0, 0, 2, 1, 1, 0, 0, 1, 0, 1, 1, 2$ is $(0, 1)(0, 0, 2, 1, 1)(0, 0, 1, 0, 1, 1, 2)$, assuming the usual ordering, $0 < 1 < 2$.

Lyndon words and Lyndon factorization are well-studied, and play an important role in string algorithms [1, 2, 10, 24, 28, 30], algebra and combinatorics [7, 17, 25], and data compression [12, 18, 20, 34, 35]. As an example, it was shown in [29] that local suffixes inside each Lyndon factor can be sorted independently and then merged to construct a string's suffix array. As another example, Lyndon factorization is used in both the construction of a string's bijective Burrows-Wheeler transform (BBWT) [13] and in performing pattern matching on indexes built from the string's BBWT [3], where the number of steps used to locate occurrences of a pattern P depends on the number of Lyndon factors within a particular suffix of P. Because of such applications, it would be beneficial to be able to control the number of factors in the Lyndon factorization of a string. Unfortunately, the Lyndon factorization of a string is fixed and unique under a fixed ordering of its alphabet [26]. However, it can vary under different alphabet orderings. For instance, if we change the alphabet ordering to $2 < 0 < 1$ in our example above, we obtain the Lyndon factorization $(0, 1)$, (0), (0), $(2, 1, 1, 0, 0, 1, 0, 1, 1)$, (2). This leads to the following problems:

▶ **Problem 1** (Lyndon Factor Minimization - Decision Version). *Given an integer A and text T over alphabet Σ, does there exist an ordering on Σ such that the number of Lyndon factors of T is at most A?*

▶ **Problem 2** (Lyndon Factor Maximization - Decision Version). *Given an integer A and text T over alphabet Σ, does there exist an ordering on Σ such that the number of Lyndon factors of T is at least A?*

We will also consider the *optimization variants* of these problems. The objective cost of a solution is the number of factors in its Lyndon factorization. In particular, for the minimization problem, a λ-approximation for $\lambda > 1$, is a polynomial-time algorithm that outputs an alphabet ordering where the number of factors is at most λ times the minimum possible number of factors over all possible alphabet orderings. Similarly, for the maximization problem, a λ-approximation for $\lambda < 1$, is a polynomial-time algorithm that outputs an alphabet ordering where the number of factors is at least λ times the maximum number of possible factors over all possible alphabet orderings.

These problems were first considered by Clare and Daykin, who proposed a polynomial-time greedy algorithm that can be adjusted to provide either a small number of factors or a large number of factors [8]. Through experiments, the authors showed that the number of factors can be significantly affected by their algorithm. Another approach that uses evolutionary algorithms to find alphabet orderings to optimize the number of Lyndon factors was considered in [9] and in [27]. Again, it was shown that there is often a significant effect on the number of factors, which can be controlled by the use of different fitness functions within the evolutionary algorithms. These techniques, although appearing to have a significant impact on the number of factors, do not provide any approximation guarantee. Motivated by this, and by other alphabet ordering hardness results presented in [4], we present the first set of hardness results on these problems.

▶ **Theorem 1.** *The decision version of Lyndon Factor Minimization is NP-complete.*

▶ **Theorem 2.** *Under the Exponential Time Hypothesis, the optimization version of Lyndon Factor Minimization cannot be solved in time* $poly(|T|) \cdot 2^{o(\sigma \log \sigma)}$.

▶ **Theorem 3.** *Under the Unique Games Conjecture, the optimization version of Lyndon Factor Minimization does not admit a* λ*-approximation for any constant* $\lambda > 1$.

▶ **Theorem 4.** *The decision version of Lyndon Factor Maximization is NP-complete.*

▶ **Theorem 5.** *Under the Unique Games Conjecture, the optimization version of Lyndon Factor Maximization does not admit a* λ*-approximation for any constant* $\lambda < 1$.

We will prove these theorems in Section 3.1, Section 3.2, Section 3.3, Section 4.1, and Section 4.2, respectively. We leave open whether it is possible to have a result similar to Theorem 2 for Lyndon Factor Maximization.

Our main line of attack is to model ordering constraint satisfaction problems (OCSPs), a subject of extensive research in its own right [5, 6, 15, 16, 31, 33]. In these problems, the task is to find a linear ordering on a set of variables subject to some additional constraints. Our work shows that a solver for these Lyndon factorization problems would be powerful enough to solve difficult OCSP instances. Our results make use of strings that allow us to model different constraint satisfaction problems and thus prove our hardness results.

2 Preliminaries

We denote the concatenation of the strings u and v using the '$\circ$' symbol, writing their concatenation as $u \circ v$. However, we omit '$\circ$' where the concatenation is clear from context and it would be cumbersome to use. Throughout this paper, we will use '$<$' and '$>$' to refer to alphabet order between symbols, the lexicographic order between strings, and the usual ordering between real numbers. Again, context will make it clear which type of order is meant. A suffix of a string T is a string v such that $T = u \circ v$ for some string u. The suffix array $SA[\cdot]$ of a string $T[1, n]$ is a length n array where $SA[i]$ is equal to the starting index of the i^{th} lexicographically smallest suffix of T. The inverse suffix array $ISA[\cdot]$ is defined as the length n array such that $i = ISA[SA[i]]$, i.e., the position in the lexicographic order of the suffix starting at index i.

The Lyndon factorization (defined in Section 1) of a string can be computed in linear time. This can be done using the well known Duval's algorithm [11], or by using the inverse suffix array, ISA, which can be constructed in linear time [22]. Lemma 6 makes it clear why the latter technique works.

▶ **Lemma 6** (Theorem 2.2 [29]). *The starting index,* i*, of a suffix in* T *that is lexicographically smaller than any suffix starting at index* $j < i$ *is an index where a Lyndon factor begins.*

We will use this observation to construct strings that model constraints that occur in an ordering constraint satisfaction problem (OCSP). The definition of an OCSP used here is less general than the one given in [14], but still sufficient for our purposes.

▶ **Definition 7.** *An OCSP of arity* k *is specified by a set* $\Lambda \subseteq S_k$ *where* S_k *is the set of permutations of* $\{1, 2, ..., k\}$*. An instance of such an OCSP consists of a set of variables,* $V = \{x_1, \dots, x_n\}$*, and* m *constraints,* $C_1, \dots, C_m$*, each of which is an ordered* k*-tuple of* V*. The objective is to find a global ordering* σ *of* V *that maximizes* $\sum_{i=1}^{m} \chi_\Lambda(\sigma_{|C_i})$*, where* $\sigma_{|C_i} \in S_k$ *is the ordering of the* k *elements of* C_i *induced by the global ordering* σ*, and* $\chi_\Lambda(\sigma_{|C_i}) = 1$ *if* $\sigma_{|C_i} \in \Lambda$ *and* 0 *otherwise. If* $\chi_\Lambda(\sigma_{|C_i}) = 1$*, we say that* C_i *is satisfied.*

Note that $m \leq n!/(n-k)! \leq n^k$. Additionally, we will only consider OCSP instances where each variable appears in at least two constraints. Under this last assumption, we can relate the number of variables, n, to the number of clauses, m.

▶ **Lemma 8.** *For OCSPs with arity k constraints, n variables, and m constraints, where every variable appears in at least two clauses, $n \leq \frac{k}{2}m$.*

Proof. Since every variable appears in at least two constraints,

$$2n \leq \sum_{i=1}^{n}(\text{the number of times variable } x_i \text{ appears in total}) = km.$$ ◀

One of the simplest OCSPs is the *Maximum Acyclic Subgraph Problem* (MAS), where $k = 2$, making constraints of the form (x_i, x_j), and where Λ contains only the identity permutation that orders $x_i < x_j$. The dual minimization problem of MAS is known as *Feedback Arc Set* (FAS). In this problem, the cost of a solution is the number of constraints being violated, instead of the number of constraints being satisfied. The problem is otherwise identical. The following hardness result for FAS is used when proving Theorem 3.

▶ **Lemma 9** ([14]). *Conditioned on the Unique Games Conjecture, for every constant $C > 1$, it is NP-hard to find a C-approximation for FAS.*

The Unique Games Conjecture is described in [21]. We will use the term Unique-Games-hard to refer to problems that, conditioned on the Unique Games conjecture, are NP-hard.

We can always assume that at least half of the constraints in an instance of MAS can be satisfied. To see this, take an arbitrary ordering of the variables. Either this ordering or its reversal must satisfy at least $m/2$ constraints. This is just a specific instance of a more general result. We can always assume our optimal solution satisfies at least $|\Lambda|m/k!$ constraints. Since the expected number of constraints satisfied by a random ordering on the variables is $|\Lambda|m/k!$, we know the maximum number of constraints satisfied by any ordering is bounded below by this quantity. It turns out, however, that finding a solution that does better than this expected value is computationally difficult. We give a simplified statement of the main result in [14], maintaining only the pertinent details for our problem.

▶ **Theorem 10** ([14]). *For an OCSP with arity k, for every constant $\varepsilon > 0$, it is Unique-Games-hard to find an ordering for the variables that achieves a ratio of satisfied constraints over total constraints that is at least $|\Lambda|/k! + \varepsilon$.*

Our results also make use of the OCSP known as the *Betweenness Problem*. In this problem $k = 3$ and Λ is of size two. For a constraint (x_i, x_j, x_k) to be satisfied either $x_i < x_j < x_k$ or $x_k < x_j < x_i$. For example, the ordering $x_4 < x_5 < x_3 < x_2 < x_1$ satisfies the constraint (x_1, x_2, x_5), but not the constraint (x_4, x_2, x_5). By applying Theorem 10 to the Betweenness problem, we obtain that it is Unique-Games-hard to achieve a ratio of satisfied constraints to total constraints better than $2/3! = 1/3$.

For hardness under the Exponential Time Hypothesis (ETH) [19], we will use a result by Kim and Gonçalves appearing in [23]. An Arity k Permutation CSP as defined in [23] is a OCSP where Λ consists of only the identity permutation, i.e., a constraint $(x_{i_1}, x_{i_2} \ldots, x_{i_j})$, is satisfied iff it is ordered $x_{i_1} < x_{i_2} < \ldots < x_{i_j}$, and constraints *up to* arity k are allowed. This is different from our definition of OCSPs, where all constraints are of exactly arity k. The differences between these two definitions are accommodated for whenever Lemma 11 is used. In [23] the authors prove the following.

▶ **Lemma 11** ([23]). *Assuming ETH, there is no $2^{o(n \log n)}$-algorithm for Arity 4 Permutation CSP (and thus for Arity k Permutation CSP, $k \geq 4$).*

3 Hardness of Lyndon Factor Minimization

The first reduction is from the Betweenness problem to the Lyndon Factor Minimization Problem. It is used to demonstrate NP-completeness. An alternative proof can be done with a reduction from MAS. Our reasoning for choosing one over the other is we believe that the Betweenness problem provides a good initial illustration of the power of a hypothetical solver to these Lyndon factorization problems. It also provides a warm-up for the techniques used in Section 3.2. Moreover, we will use a reduction from MAS as a short proof to illustrate NP-completeness for the maximization problem, before introducing a more involved reduction to prove an inapproximability result.

3.1 NP-Completeness of Lyndon Factor Minimization

We are given as input an instance ϕ of the Betweenness problem consisting of n variables x_1, x_2, …, x_n and m constraints C_1, C_2, …, C_m. Let $F(T)$ denote the number of Lyndon factors of a string T under the alphabet ordering currently under consideration. We will use $F_T(T_1)$ to denote the number of Lyndon factors of T starting within the *first occurrence* of the substring T_1 of T. The subscript T is to remind us that the factors starting in T_1 are sensitive to the other symbols in T. By a *run* of a symbol, we mean a maximal unary substring containing that symbol.

▶ **Lemma 12.** *Let T be any string of the form $T = T_1 \circ (x_0)^\alpha \circ (x_1^\gamma\ x_2^\gamma\ \dots\ x_n^\gamma)^\beta$ where T_1 is over the alphabet $\{x_0, \dots, x_n\}$, α is greater than the length of any run of x_0 in T_1, γ is greater than the length of any run of any symbol other than x_0 in T_1, and $\beta > 1$. If x_0 is the smallest symbol in the ordering, then $F(T) \leq F_T(T_1) + 1$.*

Proof. If T_1 does not end with an x_0, then the first x_0 in the $(x_0)^\alpha$ marks the start of a new Lyndon factor in T since $(x_0)^\alpha$ is lexicographically smaller than any preceding suffix. Then this factor includes the remaining suffix of T. In this case $F(T) = F_T(T_1) + 1$. If T_1 contains a suffix consisting of only x_0's, then a new Lyndon factor must start at the first of these x_0's, and again this factor contains the remaining suffix of T. In this case, $F(T) = F_T(T_1)$. ◀

▶ **Lemma 13.** *Let T be defined as in Lemma 12. If x_0 is not the smallest symbol in the ordering, $F(T) \geq \beta - 1$.*

Proof. In this case, the smallest symbol must be one of $x_1, \dots, x_n$. Suppose the smallest is x_i. Then the first symbol in the first x_i^γ marks the beginning of a Lyndon factor. This factor is of the form $x_i^\gamma\ x_{i+1}^\gamma\ \dots\ x_n^\gamma\ x_1^\gamma\ \dots x_{i-1}^\gamma$ and is repeated at least $\beta - 1$ times. In particular, the suffix $x_{i+1}^\gamma\ \dots\ x_n^\gamma$ is preceded by $\beta - 1$ factors of the form $x_i^\gamma\ x_{i+1}^\gamma\ \dots\ x_n^\gamma\ x_1^\gamma\ \dots\ x_{i-1}^\gamma$. ◀

Lemmas 12 and 13 will be useful in proving that x_0 must be smallest in an optimal ordering. We now introduce our constraint gadgets.

▶ **Lemma 14.** *Let x_0 be the smallest symbol in T. For $i, j, k > 0$, consider the first instance of a substring S of T where*

$$S = x_0^\eta\ x_j\ x_0^\eta\ x_i\ x_0^\eta\ x_j\ x_0^\eta\ x_i\ x_0^\eta\ x_k\ x_0^\eta\ x_i\ x_0^\eta\ x_i\ x_0^\eta\ x_j\ x_0^\eta\ x_j\ x_0^\eta\ x_j$$

and η is larger than the length of any run of x_0 preceding S in T, and S is immediately followed by the run $x_0^{\eta+1}$. The symbols in this first instance of S, make up three complete Lyndon factors if x_j is ordered between x_i and x_k, and four complete Lyndon factors otherwise.

Proof. Since the number of times x_0 is repeated is more than the length of any previous run, it must be the case that a new factor begins at the start of S. The six possible cases and their corresponding factorizations are:

$$\begin{aligned}
x_0 < x_i < x_j < x_k &: (x_0^\eta\ x_j), (x_0^\eta\ x_i\ x_0^\eta\ x_j\ x_0^\eta\ x_i\ x_0^\eta\ x_k), (x_0^\eta\ x_i\ x_0^\eta\ x_i\ x_0^\eta\ x_j\ x_0^\eta\ x_j\ x_0^\eta\ x_j) \\
x_0 < x_i < x_k < x_j &: (x_0^\eta\ x_j), (x_0^\eta\ x_i\ x_0^\eta\ x_j), (x_0^\eta\ x_i\ x_0^\eta\ x_k), (x_0^\eta\ x_i\ x_0^\eta\ x_i\ x_0^\eta\ x_j\ x_0^\eta\ x_j\ x_0^\eta\ x_j) \\
x_0 < x_j < x_i < x_k &: (x_0^\eta\ x_j\ x_0^\eta\ x_i\ x_0^\eta\ x_j\ x_0^\eta\ x_i\ x_0^\eta\ x_k\ x_0^\eta\ x_i\ x_0^\eta\ x_i), (x_0^\eta\ x_j), (x_0^\eta\ x_j), (x_0^\eta\ x_j) \\
x_0 < x_k < x_i < x_j &: (x_0^\eta\ x_j), (x_0^\eta\ x_i\ x_0^\eta\ x_j), (x_0^\eta\ x_i), (x_0^\eta\ x_k\ x_0^\eta\ x_i\ x_0^\eta\ x_i\ x_0^\eta\ x_j\ x_0^\eta\ x_j\ x_0^\eta\ x_j) \\
x_0 < x_j < x_k < x_i &: (x_0^\eta\ x_j\ x_0^\eta\ x_i\ x_0^\eta\ x_j\ x_0^\eta\ x_i\ x_0^\eta\ x_k\ x_0^\eta\ x_i\ x_0^\eta\ x_i), (x_0^\eta\ x_j), (x_0^\eta\ x_j), (x_0^\eta\ x_j) \\
x_0 < x_k < x_j < x_i &: (x_0^\eta\ x_j\ x_0^\eta\ x_i), (x_0^\eta\ x_j\ x_0^\eta\ x_i), (x_0^\eta\ x_k\ x_0^\eta\ x_i\ x_0^\eta\ x_i\ x_0^\eta\ x_j\ x_0^\eta\ x_j\ x_0^\eta\ x_j)
\end{aligned}$$

Notice that only in the first and last orderings where the constraint is satisfied are there three factors. The other cases have four. ◀

For each constraint $C_t = (x_i, x_j, x_k)$ in the instance ϕ of the Betweenness problem, where $1 \le t \le m$, we construct the gadget from Lemma 14,

$$S(C_t) := x_0^t\ x_j\ x_0^t\ x_i\ x_0^t\ x_j\ x_0^t\ x_i\ x_0^t\ x_k\ x_0^t\ x_i\ x_0^t\ x_i\ x_0^t\ x_j\ x_0^t\ x_j\ x_0^t\ x_j.$$

We next define $S(\phi) := S(C_1) \circ S(C_2) \circ \ldots \circ S(C_m) \circ (x_0)^{m+1} \circ (x_1^2\ x_2^2\ \ldots\ x_n^2)^\beta$ where $\beta = 3m + 3$.

▶ **Lemma 15.** *The string $S(\phi)$ has an alphabet ordering yielding at most $3m + 1$ Lyndon factors iff there exists a variable ordering satisfying all constraints in ϕ.*

Proof. Assuming there exists a constraint satisfying variable ordering for ϕ, make x_0 the smallest symbol and order the remaining symbols $x_1, \ldots, x_n$ according to the variable ordering. By Lemma 14, each of the substrings $S(C_t)$ for $1 \le t \le m$ contributes three factors, and by the analysis in Lemma 12 the remaining suffix contributes one additional factor. This creates $3m + 1$ factors in total.

Conversely, assume that no variable ordering exists that satisfies the constraints. If x_0 is the smallest symbol, then at least one $S(C_t)$ gadget contributes four factors while the others contribute at least three. The remaining suffix contributes one factor making the number of factors at least $4 + 3(m - 1) + 1 = 3m + 2$. If x_0 is not the smallest symbol, then by Lemma 13, the number of factors is at least $\beta - 1 = (3m + 3) - 1 = 3m + 2$. ◀

Since determining if there exists a variable ordering satisfying all constraints in an instance of the Betweenness problem is NP-hard [32], determining whether there exists an alphabet order where there are at most $3m + 1$ Lyndon factors is NP-hard as well. With a symbol ordering as a polynomial sized certificate, the problem is clearly in NP, proving Theorem 1.

3.2 ETH Hardness of Lyndon Factor Minimization

Here we reduce Arity 4 Permutation CSP to Lyndon Factor Minimization. Assume for the moment that x_0 is the smallest symbol, and that each substring $S(C_t)$ (yet to be defined) is followed by a run of x_0 longer than any run of x_0 that precedes it.

For an arity 2 constraint $C_t = (x_i, x_j)$, we construct a string using the symbols x_0, x_i, and x_j that has either 3 or 4 factors depending on the ordering on the variables. We will demonstrate which orderings create which factorizations. The string we construct is

$S(C_t) = x_0^t\ x_i\ x_0^t\ x_i\ x_0^t\ x_i\ x_0^t\ x_j\ x_0^t\ x_i\ x_0^t\ x_i$, which has the factorizations for different orderings,

Ordering	Factorization	# factors
$x_i < x_j :$	$(x_0^t\ x_i\ x_0^t\ x_i\ x_0^t\ x_i\ x_0^t\ x_j)(x_0^t\ x_i)(x_0^t\ x_i)$	3
$x_j < x_i :$	$(x_0^t\ x_i)(x_0^t\ x_i)(x_0^t\ x_i)(x_0^t\ x_j\ x_0^t\ x_i\ x_0^t\ x_i)$	4

Slightly more involved are the strings to model arity 3 constraints $C_t = (x_i, x_j, x_k)$, $S(C_t) = x_0^t\ x_i\ x_0^t\ x_i\ x_0^t\ x_j\ x_0^t\ x_i\ x_0^t\ x_i\ x_0^t\ x_k\ x_0^t\ x_i\ x_0^t\ x_j\ x_0^t\ x_i\ x_0^t\ x_i$., where

Ordering	Factorization	# factors
$x_i < x_j < x_k :$	$(x_0^t\ x_i\ x_0^t\ x_i\ x_0^t\ x_j\ x_0^t\ x_i\ x_0^t\ x_i\ x_0^t\ x_k\ x_0^t\ x_i\ x_0^t\ x_j)(x_0^t\ x_i)(x_0^t\ x_i)$	3
$x_i < x_k < x_j :$	$(x_0^t\ x_i\ x_0^t\ x_i\ x_0^t\ x_j)(x_0^t\ x_i\ x_0^t\ x_i\ x_0^t\ x_k\ x_0^t\ x_i\ x_0^t\ x_j)(x_0^t\ x_i)(x_0^t\ x_i)$	4
$x_j < x_i < x_k :$	$(x_0^t\ x_i)(x_0^t\ x_i)(x_0^t\ x_j\ x_0^t\ x_i\ x_0^t\ x_i\ x_0^t\ x_k\ x_0^t\ x_i)(x_0^t\ x_j\ x_0^t\ x_i\ x_0^t\ x_i)$	4
$x_k < x_i < x_j :$	$(x_0^t\ x_i\ x_0^t\ x_i\ x_0^t\ x_j)(x_0^t\ x_i)(x_0^t\ x_i)(x_0^t\ x_k\ x_0^t\ x_i\ x_0^t\ x_j\ x_0^t\ x_i\ x_0^t\ x_i)$	4
$x_j < x_k < x_i :$	$(x_0^t\ x_i)(x_0^t\ x_i)(x_0^t\ x_j\ x_0^t\ x_i\ x_0^t\ x_i\ x_0^t\ x_k\ x_0^t\ x_i)(x_0^t\ x_j\ x_0^t\ x_i\ x_0^t\ x_i)$	4
$x_k < x_j < x_i :$	$(x_0^t\ x_i)(x_0^t\ x_i)(x_0^t\ x_j\ x_0^t\ x_i\ x_0^t\ x_i)(x_0^t\ x_k\ x_0^t\ x_i\ x_0^t\ x_j\ x_0^t\ x_i\ x_0^t\ x_i)$	4

The most involved is the gadget for an arity 4 constraint $C_t = (x_i, x_j, x_k, x_h)$, $S(C_t) = x_0^t\ x_i\ x_0^t\ x_j\ x_0^t\ x_i\ x_0^t\ x_k\ x_0^t\ x_i\ x_0^t\ x_j\ x_0^t\ x_i\ x_0^t\ x_h\ x_0^t\ x_j\ x_0^t\ x_i\ x_0^t\ x_k\ x_0^t\ x_i\ x_0^t\ x_j\ x_0^t\ x_i$ which has the following factorizations depending on the ordering given to its symbols,

Ordering ('<' omitted)	Factorization	#
$x_i, x_j, x_k, x_h :$	$(x_0^t\ x_i\ x_0^t\ x_j\ x_0^t\ x_i\ x_0^t\ x_k\ x_0^t\ x_i\ x_0^t\ x_j\ x_0^t\ x_i\ x_0^t\ x_h\ x_0^t\ x_j\ x_0^t\ x_i\ x_0^t\ x_k)(x_0^t\ x_i\ x_0^t\ x_j)(x_0^t\ x_i)$	3
$x_i, x_j, x_h, x_k :$	$(x_0^t\ x_i\ x_0^t\ x_j\ x_0^t\ x_i\ x_0^t\ x_k)(x_0^t\ x_i\ x_0^t\ x_j\ x_0^t\ x_i\ x_0^t\ x_h\ x_0^t\ x_j\ x_0^t\ x_i\ x_0^t\ x_k)(x_0^t\ x_i\ x_0^t\ x_j)(x_0^t\ x_i)$	4
$x_i, x_k, x_j, x_h :$	$(x_0^t\ x_i\ x_0^t\ x_j)(x_0^t\ x_i\ x_0^t\ x_k\ x_0^t\ x_i\ x_0^t\ x_j\ x_0^t\ x_i\ x_0^t\ x_h\ x_0^t\ x_j)(x_0^t\ x_i\ x_0^t\ x_k\ x_0^t\ x_i\ x_0^t\ x_j)(x_0^t\ x_i)$	4
$x_i, x_h, x_j, x_k :$	$(x_0^t\ x_i\ x_0^t\ x_j\ x_0^t\ x_i\ x_0^t\ x_k)(x_0^t\ x_i\ x_0^t\ x_j)(x_0^t\ x_i\ x_0^t\ x_h\ x_0^t\ x_j\ x_0^t\ x_i\ x_0^t\ x_k\ x_0^t\ x_i\ x_0^t\ x_j)(x_0^t\ x_i)$	4
$x_i, x_k, x_h, x_j :$	$(x_0^t\ x_i\ x_0^t\ x_j)(x_0^t\ x_i\ x_0^t\ x_k\ x_0^t\ x_i\ x_0^t\ x_j\ x_0^t\ x_i\ x_0^t\ x_h\ x_0^t\ x_j)(x_0^t\ x_i\ x_0^t\ x_k\ x_0^t\ x_i\ x_0^t\ x_j)(x_0^t\ x_i)$	4
$x_i, x_h, x_k, x_j :$	$(x_0^t\ x_i\ x_0^t\ x_j)(x_0^t\ x_i\ x_0^t\ x_k\ x_0^t\ x_i\ x_0^t\ x_j)(x_0^t\ x_i\ x_0^t\ x_h\ x_0^t\ x_j\ x_0^t\ x_i\ x_0^t\ x_k\ x_0^t\ x_i\ x_0^t\ x_j)(x_0^t\ x_i)$	4
$x_j, x_i, x_k, x_h :$	$(x_0^t\ x_i)(x_0^t\ x_j\ x_0^t\ x_i\ x_0^t\ x_k\ x_0^t\ x_i\ x_0^t\ x_j\ x_0^t\ x_i\ x_0^t\ x_h)(x_0^t\ x_j\ x_0^t\ x_i\ x_0^t\ x_k\ x_0^t\ x_i)(x_0^t\ x_j\ x_0^t\ x_i)$	4
$x_j, x_i, x_h, x_k :$	$(x_0^t\ x_i)(x_0^t\ x_j\ x_0^t\ x_i\ x_0^t\ x_k\ x_0^t\ x_i)(x_0^t\ x_j\ x_0^t\ x_i\ x_0^t\ x_h\ x_0^t\ x_j\ x_0^t\ x_i\ x_0^t\ x_k\ x_0^t\ x_i)(x_0^t\ x_j\ x_0^t\ x_i)$	4
$x_k, x_i, x_j, x_h :$	$(x_0^t\ x_i\ x_0^t\ x_j)(x_0^t\ x_i)(x_0^t\ x_k\ x_0^t\ x_i\ x_0^t\ x_j\ x_0^t\ x_i\ x_0^t\ x_h\ x_0^t\ x_j\ x_0^t\ x_i)(x_0^t\ x_k\ x_0^t\ x_i\ x_0^t\ x_j\ x_0^t\ x_i)$	4
$x_h, x_i, x_j, x_k :$	$(x_0^t\ x_i\ x_0^t\ x_j\ x_0^t\ x_i\ x_0^t\ x_k)(x_0^t\ x_i\ x_0^t\ x_j)(x_0^t\ x_i)(x_0^t\ x_h\ x_0^t\ x_j\ x_0^t\ x_i\ x_0^t\ x_k\ x_0^t\ x_i\ x_0^t\ x_j\ x_0^t\ x_i)$	4
$x_k, x_i, x_h, x_j :$	$(x_0^t\ x_i\ x_0^t\ x_j)(x_0^t\ x_i)(x_0^t\ x_k\ x_0^t\ x_i\ x_0^t\ x_j\ x_0^t\ x_i\ x_0^t\ x_h\ x_0^t\ x_j\ x_0^t\ x_i)(x_0^t\ x_k\ x_0^t\ x_i\ x_0^t\ x_j\ x_0^t\ x_i)$	4
$x_h, x_i, x_k, x_j :$	$(x_0^t\ x_i\ x_0^t\ x_j)(x_0^t\ x_i\ x_0^t\ x_k\ x_0^t\ x_i\ x_0^t\ x_j)(x_0^t\ x_i)(x_0^t\ x_h\ x_0^t\ x_j\ x_0^t\ x_i\ x_0^t\ x_k\ x_0^t\ x_i\ x_0^t\ x_j\ x_0^t\ x_i)$	4
$x_j, x_k, x_i, x_h :$	$(x_0^t\ x_i)(x_0^t\ x_j\ x_0^t\ x_i\ x_0^t\ x_k\ x_0^t\ x_i\ x_0^t\ x_j\ x_0^t\ x_i\ x_0^t\ x_h)(x_0^t\ x_j\ x_0^t\ x_i\ x_0^t\ x_k\ x_0^t\ x_i)(x_0^t\ x_j\ x_0^t\ x_i)$	4
$x_j, x_h, x_i, x_k :$	$(x_0^t\ x_i)(x_0^t\ x_j\ x_0^t\ x_i\ x_0^t\ x_k\ x_0^t\ x_i)(x_0^t\ x_j\ x_0^t\ x_i\ x_0^t\ x_h\ x_0^t\ x_j\ x_0^t\ x_i\ x_0^t\ x_k\ x_0^t\ x_i)(x_0^t\ x_j\ x_0^t\ x_i)$	4
$x_k, x_j, x_i, x_h :$	$(x_0^t\ x_i)(x_0^t\ x_j\ x_0^t\ x_i)(x_0^t\ x_k\ x_0^t\ x_i\ x_0^t\ x_j\ x_0^t\ x_i\ x_0^t\ x_h\ x_0^t\ x_j\ x_0^t\ x_i)(x_0^t\ x_k\ x_0^t\ x_i\ x_0^t\ x_j\ x_0^t\ x_i)$	4
$x_h, x_j, x_i, x_k :$	$(x_0^t\ x_i)(x_0^t\ x_j\ x_0^t\ x_i\ x_0^t\ x_k\ x_0^t\ x_i)(x_0^t\ x_j\ x_0^t\ x_i)(x_0^t\ x_h\ x_0^t\ x_j\ x_0^t\ x_i\ x_0^t\ x_k\ x_0^t\ x_i\ x_0^t\ x_j\ x_0^t\ x_i)$	4
$x_k, x_h, x_i, x_j :$	$(x_0^t\ x_i\ x_0^t\ x_j)(x_0^t\ x_i)(x_0^t\ x_k\ x_0^t\ x_i\ x_0^t\ x_j\ x_0^t\ x_i\ x_0^t\ x_h\ x_0^t\ x_j\ x_0^t\ x_i)(x_0^t\ x_k\ x_0^t\ x_i\ x_0^t\ x_j\ x_0^t\ x_i)$	4
$x_h, x_k, x_i, x_j :$	$(x_0^t\ x_i\ x_0^t\ x_j)(x_0^t\ x_i)(x_0^t\ x_k\ x_0^t\ x_i\ x_0^t\ x_j\ x_0^t\ x_i)(x_0^t\ x_h\ x_0^t\ x_j\ x_0^t\ x_i\ x_0^t\ x_k\ x_0^t\ x_i\ x_0^t\ x_j\ x_0^t\ x_i)$	4
$x_j, x_k, x_h, x_i :$	$(x_0^t\ x_i)(x_0^t\ x_j\ x_0^t\ x_i\ x_0^t\ x_k\ x_0^t\ x_i\ x_0^t\ x_j\ x_0^t\ x_i\ x_0^t\ x_h)(x_0^t\ x_j\ x_0^t\ x_i\ x_0^t\ x_k\ x_0^t\ x_i)(x_0^t\ x_j\ x_0^t\ x_i)$	4
$x_j, x_h, x_k, x_i :$	$(x_0^t\ x_i)(x_0^t\ x_j\ x_0^t\ x_i\ x_0^t\ x_k\ x_0^t\ x_i)(x_0^t\ x_j\ x_0^t\ x_i\ x_0^t\ x_h\ x_0^t\ x_j\ x_0^t\ x_i\ x_0^t\ x_k\ x_0^t\ x_i)(x_0^t\ x_j\ x_0^t\ x_i)$	4
$x_k, x_j, x_h, x_i :$	$(x_0^t\ x_i)(x_0^t\ x_j\ x_0^t\ x_i)(x_0^t\ x_k\ x_0^t\ x_i\ x_0^t\ x_j\ x_0^t\ x_i\ x_0^t\ x_h\ x_0^t\ x_j\ x_0^t\ x_i)(x_0^t\ x_k\ x_0^t\ x_i\ x_0^t\ x_j\ x_0^t\ x_i)$	4
$x_h, x_j, x_k, x_i :$	$(x_0^t\ x_i)(x_0^t\ x_j\ x_0^t\ x_i\ x_0^t\ x_k\ x_0^t\ x_i)(x_0^t\ x_j\ x_0^t\ x_i)(x_0^t\ x_h\ x_0^t\ x_j\ x_0^t\ x_i\ x_0^t\ x_k\ x_0^t\ x_i\ x_0^t\ x_j\ x_0^t\ x_i)$	4
$x_k, x_h, x_j, x_i :$	$(x_0^t\ x_i)(x_0^t\ x_j\ x_0^t\ x_i)(x_0^t\ x_k\ x_0^t\ x_i\ x_0^t\ x_j\ x_0^t\ x_i\ x_0^t\ x_h\ x_0^t\ x_j\ x_0^t\ x_i)(x_0^t\ x_k\ x_0^t\ x_i\ x_0^t\ x_j\ x_0^t\ x_i)$	4
$x_h, x_k, x_j, x_i :$	$(x_0^t\ x_i)(x_0^t\ x_j\ x_0^t\ x_i)(x_0^t\ x_k\ x_0^t\ x_i\ x_0^t\ x_j\ x_0^t\ x_i)(x_0^t\ x_h\ x_0^t\ x_j\ x_0^t\ x_i\ x_0^t\ x_k\ x_0^t\ x_i\ x_0^t\ x_j\ x_0^t\ x_i)$	4

The string construction for the overall reduction is almost identical to the one for ϕ in Section 3.1. We only need to select β to be slightly different. We let $\beta = 4m + 3$. This is enough to ensure that in an optimal solution x_0 must be the smallest symbol. If x_0 is smallest, in the worst-case, when all constraints are not satisfied, there are at most $4m + 1$ Lyndon factors. If x_0 is not smallest, as shown in Lemma 13, the number of factors is at least $\beta - 1 = 4m + 2$. Then, with x_0 as the minimum, each ordering on $x_1, \ldots, x_n$ gives us $3s + 4(m - s) + 1 = 4m + 1 - s$ factors, where s is the number of satisfied constraints when using the corresponding variable ordering in ϕ. Therefore, an optimal ordering for the n variables of ϕ is obtained by an order on the $(n + 1)$ symbols which minimizes the number of Lyndon factors in the string. This combined with Lemma 11 proves Theorem 2.

3.3 Inapproximability of Lyndon Factor Minimization

We will perform an approximation preserving reduction from FAS to Lyndon Factor Minimization. Recall that for FAS the arity k of the constraints is 2, so that constraints are of the form (x_i, x_j) and Λ consists of the identity permutation. In other words, the constraint is only satisfied if $x_i < x_j$. The cost of the solution will be the number of violated constraints, which we wish to minimize. Our gadget for constraint $C_t = (x_i, x_j)$ will be

$$S(C_t) = (x_0^t\ x_i) \circ (x_0^t\ x_j)^{\alpha-1}$$

where $\alpha > 1$ will be chosen later. The whole string for our reduction will be

$$T = S(\phi) = S(C_1) \circ S(C_2) \circ \ldots \circ S(C_m) \circ (x_0)^{m+1} \circ (x_1^2\ x_2^2\ \ldots\ x_n^2)^{\beta}$$

where $\beta = \alpha m + 3$. By Lemma 13, if x_0 is not smallest, then $F(T) \geq \beta - 1$. We consider next what happens in our constraint gadgets when x_0 is smallest.

▶ **Lemma 16.** *If x_0 is smallest and $x_i < x_j$ then $F_T(S(C_t)) = 1$.*

Proof. Since x_0^t is the longest run of x_0 seen so far, the start of $S(C_t)$ marks the smallest suffix seen so far when traversing T from left to right. Then, since $x_j > x_i$, the start of all substrings of the form $x_0^t\ x_j$ do not mark the start of the smallest suffix seen so far. ◀

▶ **Lemma 17.** *If x_0 is smallest and $x_j < x_i$ then $F_T(S(C_t)) = \alpha$.*

Proof. Again, since x_0^t is the longest run of x_0 seen so far, the start of $S(C_t)$ marks the smallest suffix seen so far when traversing T from left to right. However, now the start of each substring of the form $x_0^t\ x_j$ marks the start of the smallest suffix seen so far (recall after the last $x_0^t\ x_j$ there will be a longer run of x_0 than has been seen before). Hence, there are $\alpha - 1$ additional factors created. ◀

▶ **Lemma 18.** *Any alphabet ordering where x_0 is smallest has fewer factors than an alphabet ordering where x_0 is not the smallest.*

Proof. If x_0 is smallest, $F(T) = s + \alpha(m - s) + 1$ where s is the number of satisfied constraints and the $+1$ arises from the last factor, $(x_0)^{m+1} \circ (x_1^2\ x_2^2\ \ldots\ x_n^2)^{\beta}$. Because $\alpha > 1$, this is upper bounded by the case when $s = 0$ so that $F(T) \leq \alpha m + 1$. On the other hand, if x_0 is not smallest $F(T) \geq \beta - 1 = \alpha m + 2$. ◀

Henceforth, we only need to worry about the case when x_0 is the smallest. Our aim is to show that a constant approximation algorithm for Lyndon Factor Minimization allows us to construct a constant approximation algorithm for FAS. If our hypothetical approximation algorithm for Lyndon Factor Minimization ever returned a solution where x_0 is not smallest, we add the additional step of replacing that solution with any solution where x_0 is smallest, obtaining a solution that performs even better. Then our modified algorithm maintains being an approximation algorithm for Lyndon Factor Minimization (perhaps with an even smaller approximation factor).

Let s^*_F denote the number of constraints satisfied in an optimal solution of ϕ for FAS and let s^*_L denote the number of constraints in ϕ satisfied by the variable ordering obtained from our optimal, factor minimizing, alphabet order for the corresponding instance of Lyndon Factor Minimization. Also, let s denote the actual number of constraints satisfied by the variable ordering obtained from our approximate factor minimizing alphabet order for the corresponding instance of Lyndon Factor Minimization. A λ-approximation for Lyndon Factor Minimization with $\lambda > 1$ gives the following set of inequalities:

$$s^*_L + \alpha(m - s^*_L) + 1 \leq s + \alpha(m - s) + 1 \leq \lambda(s^*_L + \alpha(m - s^*_L) + 1).$$

Which can be equivalently written as

$$(m - s^*_L) + \frac{s^*_L + 1}{\alpha} \leq (m - s) + \frac{s + 1}{\alpha} \leq \lambda(m - s^*_L) + \lambda\frac{s^*_L + 1}{\alpha}. \quad (1)$$

We will show that by taking α large enough we can ensure $s^*_L = s^*_F$.

▶ **Lemma 19.** *With $\alpha = 2(m + 1) + 1$, we have that $s^*_L = s^*_F$.*

Proof. The cost of a solution of ϕ is of the form $m - s^*_F$. The solution for ϕ we get from mapping our solution for Lyndon factorization back to ϕ must have at least as many violated constraints as the optimal solution for ϕ, i.e., $m - s^*_L \geq m - s^*_F$, and so $s^*_F \geq s^*_L$. Let us suppose for the sake of contradiction that $s^*_F \geq s^*_L + 1$. This implies $m - s^*_L - (m - s^*_F) \geq 1$. Then, using in addition that $\frac{s^*_F+1}{\alpha} \leq \frac{m+1}{\alpha} \leq \frac{1}{2}$, we obtain

$$\frac{s^*_F + 1}{\alpha} - \frac{s^*_L + 1}{\alpha} \leq \frac{1}{2} < 1 \leq m - s^*_L - (m - s^*_F),$$

which implies that

$$m - s^*_F + \frac{s^*_F + 1}{\alpha} < m - s^*_L + \frac{s^*_L + 1}{\alpha}.$$

Or, written more naturally as the cost of a Lyndon Factor Minimization Problem's solution,

$$s^*_F + \alpha(m - s^*_F) + 1 < s^*_L + \alpha(m - s^*_L) + 1.$$

But then this implies that the ordering on $x_1, \ldots, x_n$ that is used to obtain the optimal solution for ϕ creates fewer Lyndon factors than our supposedly optimal solution for Lyndon Factor Minimization, a contradiction. ◀

Let us now upper bound $m - s$ (our approximate solution cost when the solution is mapped back to FAS) in terms of $\lambda(m - s^*_F)$. Combining the inequalities in (1) with Lemma 19, and the fact that $s^*_F = s^*_L \leq m$ when $\alpha = 2(m + 1) + 1$, we get that

$$m - s \leq m - s + \frac{s + 1}{\alpha} \leq \lambda(m - s^*_L) + \lambda\frac{s^*_L + 1}{\alpha} \leq \lambda\left(m - s^*_F + \frac{1}{2}\right).$$

The case where $m = s_F^*$ can easily be solved in polynomial time, so we can consider that check added to our hypothetical solution as well. Hence, we assume $m - s_F^* \geq 1 > 1/2$ and,

$$m - s_F^* \leq m - s \leq \lambda \left(m - s_F^* + \frac{1}{2} \right) < \lambda(m - s_F^* + m - s_F^*) = 2\lambda(m - s_F^*).$$

We have shown that a λ approximation for Lyndon Factor Minimization allows us to obtain, at worst, a 2λ approximation for FAS. Moreover, the α value we need to do this is polynomial in m so that the whole reduction is done in polynomial time. This polynomial time constant approximation algorithm is better then what is allowed by Lemma 9 under the Unique Games Conjecture. This completes the proof of Theorem 3.

4 Hardness of Lyndon Factor Maximization

Our approach will be similar to the one taken for minimization. First, we introduce some gadgetry for the NP-completeness proof that is later expanded upon to create an inapproximability result. As of now, the authors have not yet found gadgets to establish the same ETH hardness for the maximization problem.

4.1 NP-Completeness of Lyndon Factor Maximization

We perform a reduction from the dual of FAS, the Maximum Acyclic Subgraph Problem (MAS). Recall MAS is identical to FAS except for the cost of a solution now being the number of constraints satisfied, which we wish to maximize. For constraint $C_t = (x_i, x_j)$, we define our constraint gadget as $S(C_t) = x_0^{t+1}\ x_j\ x_0^{t+1}\ x_i$ (note the reversal of i and j). The entire string formed by our instance ϕ of FAS is

$$T = S(\phi) = (x_0\ x_1\ x_2\ \ldots\ x_n) \circ S(C_1) \circ S(C_2) \circ \ldots \circ S(C_m) \circ (x_0)^m.$$

▶ **Lemma 20.** *If x_0 is not the smallest symbol in the ordering, then $F(T) \leq n + m$.*

Proof. Suppose $x_i \neq x_0$ is the smallest symbol. Then the first Lyndon factor starting with x_i occurs in the prefix $(x_0\ x_1\ \ldots\ x_n)$. Subsequent Lyndon factors must begin with x_i. The prefix contributes at most n factors and there are at most m remaining occurrences of x_i. ◀

▶ **Lemma 21.** *In an ordering where x_0 is smallest, $F(T) = 2s + (m - s) + 1 + m$, where s is the number of constraints satisfied in MAS by the ordering given to $x_1, \ldots, x_n$.*

Proof. For a substring $S(C_t)$, if $C_t = (x_i, x_j)$ is not satisfied (i.e., $x_i > x_j$) then $F_T(S(C_t)) = 1$. If it is satisfied (i.e., $x_i < x_j$) then $F_T(S(C_t)) = 2$. The prefix $x_0\ x_1\ x_2\ \ldots\ x_n$ contributes exactly one additional factor. The suffix $(x_0)^m$ contributes m factors. ◀

▶ **Lemma 22.** *Any ordering where x_0 is the smallest has more factors than an ordering where x_0 is not the smallest.*

Proof. By Lemma 8, we can assume that $n \leq m$. Then by Lemma 20, we have that if x_0 is not smallest, $F(T) \leq n + m \leq 2m$. By Lemma 21, if x_0 is smallest then $F(T) = 2s + (m - s) + 1 + m = s + 2m + 1 > 2m$. ◀

The value $F(T)$ is maximized by an alphabet order which has the largest possible number of satisfied constraints, say s^*. This gives $(s^* + 2m + 1)$ Lyndon factors. Clearly, this solution also provides an ordering satisfying the maximum number of constraints in our MAS instance. Since MAS is NP-hard, we have shown Lyndon Factor Maximization is NP-hard as well. The decision problem is in NP using the ordering on $x_1 \ \dots \ x_n$ as a polynomial sized certificate, and this remains NP-hard as it could be used to solve the optimization problem. This completes the proof of Theorem 4.

4.2 Inapproximability of Lyndon Factor Maximization

First, let us describe the OCSP from which we are reducing. Let $k > 1$ be the arity of the constraints, which we will specify later. Each constraint will be satisfied iff the variables in that constraint have one of the $(k-1)!$ orderings where the last variable is ordered first, i.e., for constraint $(x_{i_1}, x_{i_2}, \dots, x_{i_{k-1}}, x_{i_k})$, the ordering over those variables will have $x_{i_k} < x_{i_j}$ for $j \in [1, k-1]$. According to Theorem 10, it is Unique-Games-Hard to find an approximation which beats $|\Lambda| m/k! = (k-1)! m/k! = m/k$ constraints being satisfied.

Our constraint gadget is of the form

$$S(C_t) = (x_0^{t+1} \ x_{i_1}) \circ (x_0^{t+1} \ x_{i_2}) \circ \dots \circ (x_0^{t+1} \ x_{i_{k-1}}) \circ (x_0^{t+1} \ x_{i_k})^{\alpha}$$

and our overall string constructed from our instance ϕ of OCSP is

$$T := S(\phi) = (x_0 \ x_1 \ x_2 \ \dots \ x_n) \circ S(C_1) \circ S(C_2) \circ \dots \circ S(C_m) \circ (x_0), \quad \text{where } \alpha = mn.$$

▶ **Lemma 23.** *If x_0 is not smallest then $F(T) \leq n + m$.*

Proof. Let $x_i \neq x_0$ be the smallest symbol instead. Then the prefix $(x_0 \ x_1 \ x_2 \ \dots \ x_n)$ contributes at most n factors, and each remaining factor must begin with x_i. We will show that there is at most 1 factor starting in each constraint gadget. For a given constraint containing x_i, if $x_i \neq x_{i_k}$ this is immediate. On the other hand, if $x_i = x_{i_k}$ then only its first occurrence can form a smaller suffix of T than those preceding it. In more detail, since $x_0 > x_i = x_{i_k}$, we have $x_{i_k} \ (x_0^t \ x_{i_k})^{\alpha-1} x_0 < x_{i_k} \ (x_0^t \ x_{i_k})^{\alpha-2} x_0 < x_{i_k} \ (x_0^t \ x_{i_k})^{\alpha-3} x_0 < \dots$. Note that this is the reason for the final x_0 appended to T. ◀

▶ **Lemma 24.** *If x_0 is smallest, and in constraint $C_t = (x_{i_1} \dots x_{i_k})$ the symbol x_{i_k} is smallest among $x_{i_1} \ \dots \ x_{i_k}$, then $F_T(S(C_t)) \geq \alpha$.*

Proof. Since $x_0^{t+1} x_{i_k} < x_0^{t+1} x_{i_j}$ for $j \in [1, k-1]$, and the string following $S(C_t)$ is either x_0^{t+2} (or x_0 then the empty string), the start of **each run** of x_0 in the substring $(x_0^{t+1} x_{i_k})^{\alpha}$ marks the start of a suffix smaller than any of those preceding it. ◀

▶ **Lemma 25.** *If x_0 is the smallest in the ordering, then $F(T) \geq \alpha s + 1$ where s is the number clauses in ϕ satisfied by the ordering given to $x_1 \ \dots, \ x_n$. This is larger than the number of factors from any ordering where x_0 is not the smallest.*

Proof. By Lemma 24, when x_0 is the smallest each of the satisfied constraint gadgets contributes at least α factors. In addition, the lone x_0 symbol at the end of T forms its own factor. For the second statement, we can always assume our approximate solution satisfies at least 1 constraint, hence $s \geq 1$ and $\alpha s + 1 \geq mn + 1 > m + n$, which by Lemma 23 is an upper bound on the number of factors when x_0 is not smallest. ◀

From here we only need to consider when x_0 is smallest, for the same reasoning as given in Section 3.3. Now, suppose we have a λ-approximation with $\lambda < 1$ for Lyndon Factor Maximization. Let s_L^* be the number of constraint gadgets satisfied from our optimal solution of Lyndon factor maximization, and s the number from the approximate solution. Then,

$$\lambda(\alpha s_L^* + 1 + y_L^*) \leq \alpha s + 1 + y \leq \alpha s_L^* + 1 + y_L^*$$

where y_L^* represents the number of additional factors contributed beyond $\alpha s_L^* + 1$ and y represents the number of factors beyond $\alpha s + 1$ for our approximate solution. We can equivalently write the above expression as

$$\lambda s_L^* \left(1 + \frac{1}{\alpha s_L^*} + \frac{y_L^*}{\alpha s_L^*}\right) \leq s \left(1 + \frac{1}{\alpha s} + \frac{y}{\alpha s}\right) \leq s_L^* \left(1 + \frac{1}{\alpha s_L^*} + \frac{y_L^*}{\alpha s_L^*}\right). \tag{2}$$

▶ **Lemma 26.** *For all $s \in [1, m]$, and for the corresponding y value as described above,*

$$1 \leq \left(1 + \frac{1}{\alpha s} + \frac{y}{\alpha s}\right) \leq 3.$$

Proof. We first bound y from above. Any factor in a constraint gadget begins at the start of a run x_0. In a satisfied constraint gadget, there are $k-1$ such runs outside of the $(x_0^{t+1} x_{i_k})^\alpha$ substring. Hence, each satisfied constraint gadget contributes at most $k-1$ additional factors beyond α. A constraint gadget that is not satisfied, i.e., has $x_{i_j} < x_{i_k}$ for some $j \neq k$, has the gadget's last factor beginning at the start of the substring $(x_0^{t+1}\ x_{i_j})$. This implies the substring $(x_0^{t+1}\ x_{i_k})^\alpha$ does not split into different factors. Therefore, an unsatisfied constraint gadget again contributes at most $k-1$ factors. Because of this, the m constraint gadgets contribute at most k additional factors in total and $y \leq m(k-1)$. Finally, $\alpha = mn$, hence

$$\frac{y}{\alpha s} \leq \frac{y}{\alpha} \leq \frac{m(k-1)}{\alpha} \leq \frac{mn}{\alpha} = 1 \quad \text{and} \quad \frac{1}{\alpha s} \leq \frac{1}{\alpha} = \frac{1}{nm} \leq 1.$$ ◀

Let s_C^* be the number of constraints satisfied in an optimal solution to ϕ. Like in Section 3.3, we know that $s \leq s_C^*$ and $s_L^* \leq s_C^*$, Using Lemma 26 we can easily make them differ by at most a constant factor.

▶ **Lemma 27.** *Using the definitions above, it holds that $s_C^* \leq 3 s_L^*$.*

Proof. For the sake of contradiction, assume instead that $s_C^* > 3 s_L^*$. Applying the ordering given by the optimal solution of ϕ to the symbols $x_1, \ldots, x_n$, and letting y_C^* be defined as above but for s_C^*, we have

$$s_C^* \left(1 + \frac{1}{\alpha s_C^*} + \frac{y_C^*}{\alpha s_C^*}\right) > s_C^* > 3 s_L^* \geq s_L^* \left(1 + \frac{1}{\alpha s_L^*} + \frac{y_L^*}{\alpha s_L^*}\right)$$

However, this implies $\alpha s_C^* + 1 + y_C^* > \alpha s_L^* + 1 + y_L^*$. Thus, s_L^* couldn't have been the number of constraints satisfied in an optimal solution to our Lyndon Factor Maximization instance, since using whichever ordering was used for the solution to ϕ would have given us more factors, a contradiction. ◀

By Lemma 27, we have $\frac{1}{3} s_C^* \leq s_L^*$. Multiplying both sides by $\lambda/3$, we obtain $\frac{\lambda}{9} s_C^* \leq \frac{\lambda}{3} s_L^*$. By Lemma 26 and our starting inequality in (2) we also have that

$$\lambda s_L^* \leq \lambda s_L^* \left(1 + \frac{1}{\alpha s_L^*} + \frac{y_L^*}{\alpha s_L^*}\right) \leq s \left(1 + \frac{1}{\alpha s} + \frac{y}{\alpha s}\right) \leq 3s.$$

From which we obtain $\frac{\lambda}{3} s_L^* \leq s$. Combining these inequalities with the fact that $s \leq s_C^*$, we get $\frac{\lambda}{9} s_C^* \leq s \leq s_C^*$. That is, a λ-approximation algorithm for Lyndon Factor Maximization provides at least a $\lambda/9$ -approximation algorithm for this set of OCSP problems.

To finish the proof of Theorem 5, suppose for the sake of contradiction there exists a λ-approximation algorithm for Lyndon factor maximization for some constant $\lambda < 1$. Consider the set of OCSPs problems described in beginning of Section 4.2 with arity k such that $1/k < \lambda/9$. With our reduction, we obtain a polynomial-time algorithm that can find a solution with approximation ratio better than $|\Lambda|/k! = 1/k$, proving the Unique Games Conjecture false by Theorem 10.

References

1 Hideo Bannai, Tomohiro I, Shunsuke Inenaga, Yuto Nakashima, Masayuki Takeda, and Kazuya Tsuruta. The "runs" theorem. *SIAM J. Comput.*, 46(5):1501–1514, 2017. doi:10.1137/15M1011032.

2 Hideo Bannai, Juha Kärkkäinen, Dominik Köppl, and Marcin Piatkowski. Constructing the bijective BWT. *CoRR*, abs/1911.06985, 2019. URL: http://arxiv.org/abs/1911.06985, arXiv:1911.06985.

3 Hideo Bannai, Juha Kärkkäinen, Dominik Köppl, and Marcin Piatkowski. Indexing the bijective BWT. In *30th Annual Symposium on Combinatorial Pattern Matching, CPM 2019, June 18-20, 2019, Pisa, Italy*, pages 17:1–17:14, 2019. doi:10.4230/LIPIcs.CPM.2019.17.

4 Jason W. Bentley, Daniel Gibney, and Sharma V. Thankachan. On the complexity of bwt-runs minimization via alphabet reordering. In *28th Annual European Symposium on Algorithms, ESA 2020, September 7-9, 2020, Pisa, Italy (Virtual Conference)*, pages 15:1–15:13, 2020. doi:10.4230/LIPIcs.ESA.2020.15.

5 Moses Charikar, Venkatesan Guruswami, and Rajsekar Manokaran. Every permutation CSP of arity 3 is approximation resistant. In *Proceedings of the 24th Annual IEEE Conference on Computational Complexity, CCC 2009, Paris, France, 15-18 July 2009*, pages 62–73, 2009. doi:10.1109/CCC.2009.29.

6 Moses Charikar, Konstantin Makarychev, and Yury Makarychev. On the advantage over random for maximum acyclic subgraph. In *48th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2007), October 20-23, 2007, Providence, RI, USA, Proceedings*, pages 625–633, 2007. doi:10.1109/FOCS.2007.47.

7 Kuo Tsai Chen, Ralph H Fox, and Roger C Lyndon. Free differential calculus, iv. the quotient groups of the lower central series. *Annals of Mathematics*, pages 81–95, 1958.

8 Amanda Clare and Jacqueline W. Daykin. Enhanced string factoring from alphabet orderings. *Inf. Process. Lett.*, 143:4–7, 2019. doi:10.1016/j.ipl.2018.10.011.

9 Amanda Clare, Jacqueline W. Daykin, Thomas Mills, and Christine Zarges. Evolutionary search techniques for the lyndon factorization of biosequences. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion, GECCO 2019, Prague, Czech Republic, July 13-17, 2019*, pages 1543–1550, 2019. doi:10.1145/3319619.3326872.

10 Maxime Crochemore and Dominique Perrin. Two-way string matching. *J. ACM*, 38(3):651–675, 1991. doi:10.1145/116825.116845.

11 Jean-Pierre Duval. Génération d'une section des classes de conjugaison et arbre des mots de lyndon de longueur bornée. *Theor. Comput. Sci.*, 60:255–283, 1988. doi:10.1016/0304-3975(88)90113-2.

12 Isamu Furuya, Yuto Nakashima, Tomohiro I, Shunsuke Inenaga, Hideo Bannai, and Masayuki Takeda. Lyndon factorization of grammar compressed texts revisited. In Gonzalo Navarro, David Sankoff, and Binhai Zhu, editors, *Annual Symposium on Combinatorial Pattern Matching, CPM 2018, July 2-4, 2018 - Qingdao, China*, volume 105 of *LIPIcs*, pages 24:1–24:10. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018. doi:10.4230/LIPIcs.CPM.2018.24.

13 Joseph Yossi Gil and David Allen Scott. A bijective string sorting transform. *CoRR*, abs/1201.3077, 2012. URL: http://arxiv.org/abs/1201.3077, arXiv:1201.3077.

14 Venkatesan Guruswami, Johan Håstad, Rajsekar Manokaran, Prasad Raghavendra, and Moses Charikar. Beating the random ordering is hard: Every ordering CSP is approximation resistant. *SIAM J. Comput.*, 40(3):878–914, 2011. doi:10.1137/090756144.

15 Venkatesan Guruswami and Yuan Zhou. Approximating bounded occurrence ordering csps. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques - 15th International Workshop, APPROX 2012, and 16th International Workshop, RANDOM 2012, Cambridge, MA, USA, August 15-17, 2012. Proceedings*, pages 158–169, 2012. doi:10.1007/978-3-642-32512-0_14.

16 Johan Håstad. Some optimal inapproximability results. In *Proceedings of the Twenty-Ninth Annual ACM Symposium on the Theory of Computing, El Paso, Texas, USA, May 4-6, 1997*, pages 1–10, 1997. doi:10.1145/258533.258536.

17 Christophe Hohlweg and Christophe Reutenauer. Lyndon words, permutations and trees. *Theor. Comput. Sci.*, 307(1):173–178, 2003. doi:10.1016/S0304-3975(03)00099-9.

18 Tomohiro I, Yuto Nakashima, Shunsuke Inenaga, Hideo Bannai, and Masayuki Takeda. Faster lyndon factorization algorithms for SLP and LZ78 compressed text. *Theor. Comput. Sci.*, 656:215–224, 2016. doi:10.1016/j.tcs.2016.03.005.

19 Russell Impagliazzo and Ramamohan Paturi. On the complexity of k-sat. *J. Comput. Syst. Sci.*, 62(2):367–375, 2001. doi:10.1006/jcss.2000.1727.

20 Juha Kärkkäinen, Dominik Kempa, Yuto Nakashima, Simon J. Puglisi, and Arseny M. Shur. On the size of lempel-ziv and lyndon factorizations. In Heribert Vollmer and Brigitte Vallée, editors, *34th Symposium on Theoretical Aspects of Computer Science, STACS 2017, March 8-11, 2017, Hannover, Germany*, volume 66 of *LIPIcs*, pages 45:1–45:13. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017. doi:10.4230/LIPIcs.STACS.2017.45.

21 Subhash Khot. On the unique games conjecture. In *46th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2005), 23-25 October 2005, Pittsburgh, PA, USA, Proceedings*, page 3, 2005. doi:10.1109/SFCS.2005.61.

22 Dong Kyue Kim, Jeong Seop Sim, Heejin Park, and Kunsoo Park. Linear-time construction of suffix arrays. In *Combinatorial Pattern Matching, 14th Annual Symposium, CPM 2003, Morelia, Michocán, Mexico, June 25-27, 2003, Proceedings*, pages 186–199, 2003. doi:10.1007/3-540-44888-8_14.

23 Eun Jung Kim and Daniel Gonçalves. On exact algorithms for the permutation CSP. *Theor. Comput. Sci.*, 511:109–116, 2013. doi:10.1016/j.tcs.2012.10.035.

24 Manfred Kufleitner. On bijective variants of the burrows-wheeler transform. In *Proceedings of the Prague Stringology Conference 2009, Prague, Czech Republic, August 31 - September 2, 2009*, pages 65–79, 2009. URL: http://www.stringology.org/event/2009/p07.html.

25 Pierre Lalonde and Arun Ram. Standard lyndon bases of lie algebras and enveloping algebras. *Transactions of the American Mathematical Society*, 347(5):1821–1830, 1995.

26 M. Lothaire. *Combinatorics on words*, volume 17. Cambridge university press, 1997.

27 Lily Major, Amanda Clare, Jacqueline W. Daykin, Benjamin Mora, Leonel Jose Peña Gamboa, and Christine Zarges. Evaluation of a permutation-based evolutionary framework for lyndon factorizations. In *Parallel Problem Solving from Nature - PPSN XVI - 16th International Conference, PPSN 2020, Leiden, The Netherlands, September 5-9, 2020, Proceedings, Part I*, pages 390–403, 2020. doi:10.1007/978-3-030-58112-1_27.

28 Sabrina Mantaci, Antonio Restivo, Giovanna Rosone, and Marinella Sciortino. Sorting suffixes of a text via its lyndon factorization. In Jan Holub and Jan Zdárek, editors, *Proceedings of the Prague Stringology Conference 2013, Prague, Czech Republic, September 2-4, 2013*, pages 119–127. Department of Theoretical Computer Science, Faculty of Information Technology, Czech Technical University in Prague, 2013. URL: http://www.stringology.org/event/2013/p11.html.

29 Sabrina Mantaci, Antonio Restivo, Giovanna Rosone, and Marinella Sciortino. Suffix array and lyndon factorization of a text. *J. Discrete Algorithms*, 28:2–8, 2014. doi:10.1016/j.jda.2014.06.001.

30 Marcin Mucha. Lyndon words and short superstrings. In *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2013, New Orleans, Louisiana, USA, January 6-8, 2013*, pages 958–972, 2013. doi:10.1137/1.9781611973105.69.

31 Alantha Newman. Cuts and orderings: On semidefinite relaxations for the linear ordering problem. In *Approximation, Randomization, and Combinatorial Optimization, Algorithms and Techniques, 7th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems, APPROX 2004, and 8th International Workshop on Randomization* *and Computation, RANDOM 2004, Cambridge, MA, USA, August 22-24, 2004, Proceedings*, pages 195–206, 2004. doi:10.1007/978-3-540-27821-4_18.

32 Jaroslav Opatrny. Total ordering problem. *SIAM J. Comput.*, 8(1):111–114, 1979. doi:10.1137/0208008.

33 Prasad Raghavendra. Optimal algorithms and inapproximability results for every csp? In *Proceedings of the 40th Annual ACM Symposium on Theory of Computing, Victoria, British Columbia, Canada, May 17-20, 2008*, pages 245–254, 2008. doi:10.1145/1374376.1374414.

34 Kazuya Tsuruta, Dominik Köppl, Yuto Nakashima, Shunsuke Inenaga, Hideo Bannai, and Masayuki Takeda. Grammar-compressed self-index with lyndon words. *CoRR*, abs/2004.05309, 2020. URL: https://arxiv.org/abs/2004.05309, arXiv:2004.05309.

35 Yuki Urabe, Yuto Nakashima, Shunsuke Inenaga, Hideo Bannai, and Masayuki Takeda. On the size of overlapping lempel-ziv and lyndon factorizations. In Nadia Pisanti and Solon P. Pissis, editors, *30th Annual Symposium on Combinatorial Pattern Matching, CPM 2019, June 18-20, 2019, Pisa, Italy*, volume 128 of *LIPIcs*, pages 29:1–29:11. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPIcs.CPM.2019.29.