

The K-Centre Problem for Necklaces

Duncan Adamson, Argyrios Deligkas, Vladimir V. Gusev, Igor Potapov

May 21, 2020

Abstract

In graph theory, the objective of the k -centre problem is to find a set of k vertices for which the largest distance of any vertex to its closest vertex in the k -set is minimised. In this paper, we introduce the k -centre problem for sets of necklaces, i.e. the equivalence classes of words under the cyclic shift. This can be seen as the k -centre problem on the complete weighted graph where every necklace is represented by a vertex, and each edge has a weight given by the overlap distance between any pair of necklaces. Similar to the graph case, the goal is to choose k necklaces such that the distance from any word in the language and its nearest centre is minimised. However, in a case of k -centre problem for languages the size of associated graph maybe exponential in relation to the description of the language, i.e., the length of the words ℓ and the size of the alphabet q . We derive several approximation algorithms for the k -centre problem on necklaces, with logarithmic approximation factor in the context of ℓ and k , and within a constant factor for a more restricted case.

1 Introduction

In graph theory, the objective in k -centre problem is to find a set of k vertices for which the largest distance of any vertex of the graph and its closest vertex in this k -set is minimised. The numerous applications of the problem in various areas of computer science, lead to different definitions of connectivity and distance between the vertices depending on the application at hand. The k -centre problem on graphs is known to be NP-hard. The best performance ratio for a polynomial-time approximation solution is 2 unless $P = NP$, but it is unlikely to be fixed-parameter tractable (FPT) in a context of the most natural parameter k , which is the number of centres [6].

A different form of the k -centre problem appears in stringology and it was linked with important applications in computational biology for example to find the approximate gene clusters for a set of words over the DNA alphabet [12]. This problem is also NP-hard problem; there are fixed-parameter algorithms and heuristic algorithms for it without any performance guarantee. The closest string problem aims to find a new string within a distance d to each input of n strings and such that d is minimized. The natural generalization of k -Closest String problem is of finding k -centre strings of a given length minimizing the distance from every string to closest centre [7, 9]. This problem has been mainly studied for the most popular distance measure which is the Hamming distance. The major application of this distance is in the coding theory, but it also has been intensively used in many biological applications which aim to discover a region of similarity or to design probes or primers [11].

In this paper we define and study a new variant of the k -centre problem on the new objects, the class of *necklaces*, and using a different distance function, *the overlap coefficient* to define the closeness of strings or necklaces. *Necklaces* are classical structures in combinatorics, which can be defined as a set of ℓ -character strings over an alphabet of size q , that are equivalent under the

cyclic shift operation. The research on necklaces in combinatorics has been mainly focused on characterisation of these objects, their efficient generation and comparison, ordering as well as on the design of an efficient ranking and unranking procedures [10, 16]. In this paper, we link several problems and interconnect research ideas on these fascinating objects as well as design approximation algorithms for the k -centres problem on necklaces, which have applications for combinatorial crystal structure prediction [1, 2].

In particular we are motivated to study k -centre problem for a class of necklaces by the *Extended Module Materials Assembly (EMMA)* method used for in silico predictions of novel materials [4, 5]. The idea of this method is to consider materials assembled from well-chosen layers, where the arrangements with low enough energy constitute potentially stable materials. Since the space of all potential stackings is typically too large for exhaustive search, one is looking for a diverse and representative sample of this space to be used with further optimisation strategies. To approach this problem, we model layers as letters and materials (periodic crystals) as necklaces due to their invariance under the cyclic shift operation. The set of necklaces during the optimisation is often further constrained: fixed chemical composition corresponds to necklaces with the fixed Parikh vector and constraints on the relative position of layers lead to necklaces with forbidden subwords. These considerations lead us to the general problem of sampling from languages: \mathcal{L}_ℓ^Σ – the set of all words of length ℓ ; $\mathcal{L}_\ell^{\leq \Sigma}$ – the set of all words of length at most ℓ ; $\mathcal{L}_\mathbf{P}^\Sigma$ – the set of all words with the Parikh vector \mathbf{P} and $\mathcal{L}_\ell^{\Sigma \setminus F}$ – the set of all words of length at most ℓ that do not contain words from a finite set \mathcal{F} as factors. Apart from the Hamming distance, there are several well known methods for comparing words with their own advantages and disadvantages [3, 13, 14]. In order to define the closeness between different necklaces we use one of such methods based on computing the overlap coefficient between each pair of necklaces. In particular, in the context of the material science, two patterns of layers may have closer properties if they have more common fragments.

In general the k -centre problem on necklaces can be seen as the k -centre problem on the complete weighted graph where every word is represented by a node, and each edge has a weight given by the overlap distance between the two words. As in the graph case, the goal is to choose k words such that the distance from any word in the language and its nearest centre is minimised. However, in a case of k -centre problem for languages the associated graph can be of exponential size in the context of the input.

The main results of this paper is in the design of approximation algorithms for several finite languages of necklaces \mathcal{L}_ℓ^Σ , $\mathcal{L}_\mathbf{P}^\Sigma$ and $\mathcal{L}_\ell^{\Sigma \setminus F}$ with logarithmic approximation factor in the context of ℓ and k and log-linear for $\mathcal{L}_\ell^{\leq \Sigma}$. The first algorithm is based on building a prefix tree of necklaces utilising previously designed ranking and unranking procedures. Then we extend an approximation algorithm for a language of necklaces with forbidden words by designing new procedures for ranking and unranking of necklaces under forbidden words and Parikh map constraints, where both limitations are motivated by natural material science constraints. Finally we propose a different technique based on building k centres for \mathcal{L}_ℓ^Σ via the de Bruijn sequences, which can find a solution in linear time with a constant approximation factor.

2 Preliminaries

A formal *language* \mathcal{L} consists of words whose letters are taken from an alphabet and are defined according to a specific set of rules. We focus on *cyclic languages* languages consisting of cyclic words *only*. A cyclic word is the equivalence class of words under the *cyclic shift* operation, also known as *necklace*. A cyclic shift of size i moves the suffix of length i to the front of the word, while maintaining the relative order within the suffix. More formally, the cyclic shift of length i

on the word $w = w_1w_2 \dots w_n$ will transform it to $w_{n-i+1} \dots w_nw_1 \dots w_{n-i}$. Any word that is a member of this equivalence class is a *representation* of the cyclic word. In general, cyclic words are represented by the lexicographically smallest word in this equivalence class, known as the *canonical representation*. *Lyndon words* form the set of aperiodic necklaces; necklaces such that given the canonical form $w_1w_2 \dots w_\ell$, there exists no cyclic shift such that $w_iw_{i+1} \dots w_\ellw_1 \dots w_{i-1} = w_1 \dots w_\ell$ for any $i \neq 1$.

The class of *fixed length cyclic languages* consists of all cyclic words from an alphabet Σ , made of any combination of characters of Σ with length ℓ . This is equivalent to the set of necklaces of length ℓ over the alphabet Σ . This language will be denoted \mathcal{L}_ℓ^Σ . We focus on two restricted cases of this language.

The first is the *fixed length cyclic language with forbidden subwords*. This is a fixed length cyclic language where any word does not contain subwords from a set of forbidden words. Given the set of forbidden words F , the language of all words of length ℓ without these will be denoted $\mathcal{L}_\ell^{\Sigma \setminus F}$. Formally a word $w \in \mathcal{L}_\ell^\Sigma$ will be in $\mathcal{L}_\ell^{\Sigma \setminus F}$ if there is no representation of it of the form $w_1 \dots w_i f_1 \dots f_j w_{i+j+1} \dots w_\ell$ for any word $f_1 \dots f_j \in F$.

The second language is the *fixed content cyclic language*. Here, in any word the number of occurrences of each letter of Σ is fixed. The number of occurrences of each character will be given as a vector \mathbf{P} , where \mathbf{P}_i denotes the number of occurrences of the i^{th} character. This language will be denoted $\mathcal{L}_{\mathbf{P}}^\Sigma$.

A generalisation of \mathcal{L}_ℓ^Σ and $\mathcal{L}_\ell^{\Sigma \setminus F}$ is to *maximum length languages*. These contain every word of length less than or equal to ℓ in the corresponding fixed length language. For a given ℓ , the maximum length generalisation of \mathcal{L}_ℓ^Σ will be denoted $\mathcal{L}_{\leq \ell}^\Sigma$ and $\mathcal{L}_\ell^{\Sigma \setminus F}$ will be denoted $\mathcal{L}_{\leq \ell}^{\Sigma \setminus F}$. Formally, this can be written as $\mathcal{L}_{\leq \ell}^\Sigma = \bigcup_{i=1}^{\ell} \mathcal{L}_i^\Sigma$ and $\mathcal{L}_{\leq \ell}^{\Sigma \setminus F} = \bigcup_{i=1}^{\ell} \mathcal{L}_i^{\Sigma \setminus F}$. As a cyclic word of length ℓ can be seen as a word of infinite length with a period of a most ℓ , when comparing two cyclic words it makes sense to look at two representatives of these words with the same length, which will be called the *same length representatives* of the words.

The Overlap Coefficient. The *Overlap coefficient* of the sets A and B is defined as the size of the intersection of the two sets, normalised by the size of the smaller set, i.e. $\mathfrak{C}(A, B) = \frac{|A \cap B|}{\min(|A|, |B|)}$. For the overlap coefficient measures the closeness of two sets A and B , where a value of 1 means that the two sets are identical, and a value of 0 means there are no shared elements.

The Overlap coefficient $\mathfrak{C}(\alpha, \beta)$ for two cyclic words α and β is defined as the overlap coefficient between the multisets of all subwords of the same length representatives of α and β . Given the same length representative a some word α of length $p \cdot \ell$, the multiset of subwords of length l is the multiset of each subword starting at every position from 1 to $p \cdot \ell$, labelled by the number of occurrences of this subword up to this point. Note that as this word is cyclic, subwords of length l may occur beginning in the last $l - 1$ positions of the word, giving a total of $p \cdot \ell$ subwords of length l for any l . For example, given the word $aaab$, the multiset of subwords of length 2 are $\{aa_1, aa_2, ab_1, ba_1\}$. The multiset of all subwords is simply the union of the multisets of the subwords for every length from 1 to $p \cdot \ell$, having a total size of $(\ell \cdot p)^2$. An example of this is given explicitly between ab and abb in Figure 1.

To use this as a distance, the measure will be inverted so that a value of 1 will imply the strings share no similarity and a value of 0 implies they represent the same word. From this, the Overlap distance between two cyclic words α and β will be given by

$$\mathfrak{D}(\alpha, \beta) = \begin{cases} \infty, & \text{if } \mathfrak{C}(\alpha, \beta) = 0 \\ 0 & \text{if } \mathfrak{C}(\alpha, \beta) = 1 \\ \frac{1}{\mathfrak{C}(\alpha, \beta)} & \text{Otherwise.} \end{cases}$$

	word ab with representative $(ab)^3$	word abb with representative $(abb)^2$
1	$\mathbf{a}_1, \mathbf{b}_1, \mathbf{a}_2, \mathbf{b}_2, \mathbf{a}_3, \mathbf{b}_3$	$\mathbf{a}_1, \mathbf{b}_1, \mathbf{b}_2, \mathbf{a}_2, \mathbf{b}_3, \mathbf{b}_4$
2	$\mathbf{ab}_1, \mathbf{ba}_1, \mathbf{ab}_2, \mathbf{ba}_2, \mathbf{ab}_3, \mathbf{ba}_3$	$\mathbf{ab}_1, \mathbf{bb}_1, \mathbf{ba}_1, \mathbf{ab}_2, \mathbf{bb}_2, \mathbf{ba}_2$
3	$\mathbf{aba}_1, \mathbf{bab}_1, \mathbf{aba}_2, \mathbf{bab}_2, \mathbf{aba}_3, \mathbf{bab}_3$	$\mathbf{abb}_1, \mathbf{bba}_1, \mathbf{bab}_1, \mathbf{abb}_2, \mathbf{bba}_2, \mathbf{bab}_2$
4	$\mathbf{abab}_1, \mathbf{baba}_1, \mathbf{abab}_2, \mathbf{baba}_2, \mathbf{abab}_3, \mathbf{baba}_3$	$\mathbf{abba}_1, \mathbf{bbab}_1, \mathbf{babb}_1, \mathbf{abba}_2, \mathbf{bbab}_2, \mathbf{babb}_2$
5	$\mathbf{ababa}_1, \mathbf{babab}_1, \mathbf{ababa}_2, \mathbf{babab}_2, \mathbf{ababa}_3, \mathbf{babab}_3$	$\mathbf{abbab}_1, \mathbf{bbabb}_1, \mathbf{babba}_1, \mathbf{abbab}_2, \mathbf{bbabb}_2, \mathbf{babba}_2$
6	$\mathbf{ababab}_1, \mathbf{bababa}_1, \mathbf{ababab}_2, \mathbf{bababa}_2, \mathbf{ababab}_3, \mathbf{bababa}_3$	$\mathbf{abbabb}_1, \mathbf{bbabba}_1, \mathbf{babbab}_1, \mathbf{abbabb}_2, \mathbf{bbabba}_2, \mathbf{babbab}_2$

Figure 1: Example of the overlap coefficient calculation for a pair of words ab and abb . There are 11 common subwords out of the total number of 36 subwords of length from 1 till 6 in the same length cyclic words representatives $(ab)^3$ and $(abb)^2$, so $\mathfrak{C}(ab, abb) = \frac{11}{36}$ and $\mathfrak{D}(ab, abb) = \frac{36}{11}$.

$\alpha \backslash \beta$	A	B	C	D	E	F
A	0	$\frac{16}{6}$	$\frac{16}{3}$	8	16	∞
B	$\frac{16}{6}$	0	$\frac{16}{7}$	$\frac{16}{6}$	4	16
C	$\frac{16}{3}$	$\frac{16}{7}$	0	$\frac{16}{6}$	2	$\frac{16}{3}$
D	8	$\frac{16}{6}$	$\frac{16}{6}$	0	$\frac{16}{10}$	8
E	16	4	2	$\frac{16}{6}$	0	$\frac{16}{6}$
F	∞	16	$\frac{16}{3}$	8	2	0

Figure 2: Example of the overlap distance $\mathfrak{D}(\alpha, \beta)$ for binary cyclic words of length 4.

The k -Centre Problem on Necklaces. With this distance, the k -centre problem for languages can be defined. This can be thought of as the k -centre problem on the complete weighted graph where every word is represented by a node, and each edge has a weight given by the overlap distance between the two words. As in the graph case, the goal here is to choose k words such that the distance from any word in the language and its nearest centre is minimised. However in a case of k -centre problem for languages the size of associated graph maybe exponential in relation to the description of the language, i.e., the input size.

Problem 1. k -Centre problem for languages: Given a finite language \mathcal{L} and an integer k , select k words from \mathcal{L} forming a sample S of k -centres, minimising the maximum overlap distance between every word in \mathcal{L} and the nearest member of S :

$$\mathfrak{D}_{\mathcal{L},k} = \min_{|S|=k} \max_{v \in \mathcal{L}} \min_{s \in S} \mathfrak{D}(s, v). \quad (1)$$

Our goal is to maximise the length λ of the longest subword such that every word in \mathcal{L} shares a subword of length λ with at least one member of the sample. The idea behind this approach is that if a word shares a subword of length λ with the sample, it will also share 2 words of length $\lambda - 1$, 3 of length $\lambda - 2$, and so on, for a total of $\frac{\lambda(\lambda+1)}{2}$ common subwords. Therefore by increase the length of these subwords by 1, there is a quadratic increase in the size of the intersection in the overlap coefficient. This provides a bound on the maximum distance between every word in \mathcal{L} and the centres is created of $\mathfrak{D}_{\mathcal{L},k} \leq \frac{\ell^2}{\lambda(\lambda+1)}$. So our algorithms will be focused on maximising the length of λ for the languages \mathcal{L}_ℓ^Σ , $\mathcal{L}_{\leq \ell}^\Sigma$, $\mathcal{L}_\ell^{\Sigma \setminus F}$, $\mathcal{L}_{\leq \ell}^{\Sigma \setminus F}$, and \mathcal{L}_ℓ^Σ .

Lemma 1. For the language \mathcal{L}_ℓ , given λ as longest length such that every subword in \mathcal{L} shares a subword of length λ with at least one centre, $\mathfrak{D}_{\mathcal{L}_\ell, k} \leq \frac{2\ell^2}{\lambda(\lambda+1)}$.

Proof. Let us assume that every word $w \in \mathcal{L}$ shares at least one subword of length λ with at least one centre w_c . Thus, w_c will contain all subwords of this shared word, giving a total of $\frac{\lambda(\lambda+1)}{2}$ shared subwords. This gives an intersection of w with the closest member of the sample set of size at least $\frac{\lambda(\lambda+1)}{2}$. As the size of the multiset of subwords (in cyclic words of length l) will be ℓ^2 , the total distance will be $\frac{2\ell^2}{\lambda(\lambda+1)}$. \square

Lemma 2. *For the language \mathcal{L}_ℓ^Σ and a sample S of k -centres the optimal overlap distance $\mathcal{D}_{\mathcal{L},k}$ between every word in \mathcal{L} and the nearest member of S is no less than $\frac{\ell^2}{\log_q(\ell^2 k)(\log_q(\ell^2 k)+1)}$.*

Proof. Let λ be the longest subword such that every word in \mathcal{L} shares a subword of length λ with at least one of the k centres. To get an upper bound on the size of the λ , observe that every word in \mathcal{L} is a necklace, therefore every word must have at least one subword of length λ that is a prefix of a necklace. Therefore the longest value for λ is the largest value such that there are fewer than $k \times \ell$ necklace prefixes. A simple lower bound on the number of necklace prefixes of length λ for an alphabet of size q is $\frac{q^\lambda}{\lambda}$. This can be rewritten as an inequality in terms of k and ℓ as $\frac{q^\lambda}{\lambda} \leq k\ell$. Observing that $\lambda \leq \ell$ gives $\frac{q^\lambda}{\ell} \leq k\ell$, giving as a bound on λ , $\lambda \leq \log_q(\ell^2 k)$.

Assume that the furthest word share two subwords of length λ with the nearest centre. For this to be the case, every centre must contain every subword of length λ , with at least one occurring twice, requiring the string to be of length $q^\lambda + 1$, which is clearly greater than ℓ under the assumption that $\lambda = \log_q(\ell^2 k)$. Using this as an upper bound, the intersection may be of size no more than $\lambda(\lambda + 1)$, giving a distance of $\frac{\ell^2}{\lambda(\lambda+1)}$. Using the upper bound on λ gives a lower bound on the distance of $\frac{\ell^2}{\log_q(\ell^2 k)(\log_q(\ell^2 k)+1)}$. \square

Lemma 3. *Given an algorithm that can approximate the solution to Problem 1 for \mathcal{L}_ℓ^Σ within a factor f , the same sample will be an approximation of $(\ell - 1)f$ of the optimal solution to Problem 1 for $\mathcal{L}_{\leq \ell}^\Sigma$.*

Proof. Let λ be the length of the longest subword such that every word in \mathcal{L}_ℓ^Σ shares a subword of length λ with at least one centre in the sample. Observe that for every length $l \leq \ell$, every word in \mathcal{L}_l^Σ will occur as a subword of at least q words in \mathcal{L}_ℓ^Σ . As the samples must include every subword of length λ as a subword, any word of length $\lambda < l < \ell$ will also share a common subword of length λ with each centre, which when converted to the same length representatives of length $l \cdot \ell$ will lead to an overlap coefficient of $\frac{l\lambda(\lambda+1)}{2(l\ell)^2}$. For words of length $l < \lambda$, observe that the same length representative with a word of length ℓ will also be of length at least ℓ , ensuring that it shares a common subword of length at least λ . As in the case $l \geq \lambda$, the overlap coefficient between some word with length $l < \lambda$ will be $\frac{l\lambda(\lambda+1)}{2(l\ell)^2}$.

Dividing this by the bound given in Lemma 1 gives $\frac{2l\lambda(\lambda+1)\ell^2}{2\ell^2 l^2 \lambda(\lambda+1)} = \frac{1}{l}$. The worst case for this will be $l = \ell - 1$. Therefore, the solution for an algorithm guaranteeing an approximation factor of f for Problem 1 on the language \mathcal{L}_ℓ^Σ will give a solution that is an approximation of the optimal solution by a factor of $(\ell - 1)f$. \square

3 Sampling via Prefix Trees

In this section we will look at a generic framework for sampling necklaces under various constraints. This will give a logarithmic approximation factor relative to the number of samples in the general case. In Section 3.1 we will present the algorithm and show how it preforms on the languages \mathcal{L}_ℓ^Σ and $\mathcal{L}_\mathbf{P}^\Sigma$. In Section 3.2 we will extend the ranking function for necklaces to the language $\mathcal{L}_\ell^{\Sigma \setminus F}$.

3.1 The general algorithm

The underlying idea behind the first algorithm is of building samples based on covering different possible prefixes of necklaces. The reasoning behind this is twofold: first the set of prefixes for necklaces is much more limited than it is for unconstrained words, and secondly by covering all prefixes up to a given length λ , all words are guaranteed to share a subword of length λ with some sample.

In order to do this effectively, it will be important to compute how many necklaces have a given prefix. This may be done by *ranking* the necklace. The rank of a necklace is the number of necklaces for which the canonical representation is lexicographically smaller than it. The first algorithm to rank necklaces was given by Kopparty et. al. [10] without a tight bound, followed by an algorithm by Sawada and Williams [16] who provided an $O(\ell^3)$ time algorithm. Sawada and Williams show how this may be used to find the number of necklaces with a given prefix, by computing the difference between the ranks of smallest and largest necklace with the given prefix, both of which may be done in quadratic time. This ranking function has been further extended to the fixed content case by Hartman and Sawada [8].

The k-centres selection based on a tree of necklace prefixes: The algorithm recursively builds the tree of possible necklace prefixes, starting with the empty string, in a breadth first manner, continuing until there are k such prefixes. Once these prefixes have been generated, the centres can be built as necklaces containing these prefixes.

This is achieved as follows. At each step there is the set of prefixes P of a length l such that the number of prefixes is less than k . Observe that every prefix in the set of prefixes of length $l+1$, P' , will consist of a prefix from P followed by a character. Note also that every $p \in P$ must be the prefix for at least one member of P' . Therefore to generate P' , each prefix in p must be considered. Given $p \in P$ and $\sigma \in \Sigma$, $p\sigma$ will be in P' if and only if it is the prefix of a necklace. To determine this property, the *rank* of the smallest and largest non-cyclic words starting with $p\sigma$ amongst the set of necklaces can be used. The rank of a word w amongst the set of necklaces will be denoted $rank(w)$. Let $p\sigma 1^{\ell-l-1}$ denote the smallest word starting with $p\sigma$, i.e. the word consisting of $p\sigma$ followed by $\ell-l-1$ copies of the smallest character, and $p\sigma q^{\ell-l-1}$ denote the largest word starting with $p\sigma$. The number of necklaces sharing the prefix $p\sigma$ will be given by $rank(p\sigma q^{\ell-l-1}) - rank(p\sigma 1^{\ell-l-1})$. If there are no such necklaces, then $p\sigma$ will be discarded, otherwise it will be added to P' . The set P' will be generated by repeating this process for every $p \in P$, $\sigma \in \Sigma$. Once the size of P' is greater than k , the algorithm will terminate using the prefixes in P as a basis. For each $p \in P$, a centre will be generated by appending an arbitrary subword following the prefix.

Lemma 4. *There exists a polynomial-time algorithm to construct k centres of \mathcal{L}_ℓ^Σ such that every word in \mathcal{L}_ℓ^Σ shares a common substring of length at least $\log_q k - 1$ with the nearest centre, and therefore will be at a distance of no more than $\frac{2\ell^2}{\log_q^2 k}$ from the nearest centre.*

Proof. Let λ be the length of the prefixes at the termination of the algorithm. To bound the length of λ , observe that each sample corresponds to a prefix of length λ . Therefore, this becomes the problem of determining the largest value of λ such that the size of the set is less than k . An upper bound on the size of the set of necklace prefixes of length λ can be taken as the sum of the upper bound on the number of Lyndon words of length 1 to λ . This gives the $\sum_{i=1}^{\lambda} \frac{q^i}{i}$. Ignoring the divisor by i allows the upper bound to be rewritten into the inequality $\frac{q(q^\lambda-1)}{(q-1)} \leq k$. Note that $\log_q(k(q-1)) - 1 = \log_q k + \log_q(q-1) - 1$, but $\log_q(q-1) < 1$ for any $q \geq 2$, so if we are considering only integer values then $\lfloor \log_q(k(q-1)) - 1 \rfloor = \log_q(k) - 1$. Lemma 1 gives a lower bound on the

distance between every word in \mathcal{L}_ℓ^Σ and the nearest centre in the sample of $\frac{2\ell^2}{(\log_q k-1)(\log_q k-2)}$, which is bounded by $\frac{2\ell^2}{\log_q^2 k}$.

To show that the method will terminate in polynomial time, we note that the time to compute the number of necklaces with a given prefix will be $O(\ell^2)$. For every $i \leq \lambda + 1$, at most k samples will be checked. To determine the longest λ , let there be p_i prefixes of length i . Observe that there will be at least $p_i + q - 1$ words of length $i = 1$. Therefore the number of prefixes of length i will be at least $(i + 1)q - i$. Therefore the longest length will be $\frac{k-q}{q-1}$. Thus the maximum number prefixes that need to be checked will be $k \cdot \frac{k-q}{q-1}$ and the total complexity will be $O\left(k \cdot \frac{k-q}{q-1} \ell^2\right)$. \square

Theorem 1. *Problem 1 can be solved by the polynomial time approximation algorithm for a language \mathcal{L}_ℓ^Σ with an approximation factor $O(\log_k^2 \ell)$ and a language $\mathcal{L}_{\leq \ell}^\Sigma$ with $O(\ell \cdot \log_k^2(\ell))$.*

Proof. Recall, from Lemma 2 the lower bound on the value $\mathfrak{D}_{\mathcal{L}_\ell^\Sigma, k}$ is $\frac{\ell^2}{\log_q(\ell^2 k)(\log_q(\ell^2 k)+1)}$. Dividing the bound from Lemma 4 by the lower bound on the distance from Lemma 2 gives a performance ratio $\frac{2\log_q(\ell^2 k)(\log_q(\ell^2 k)+1)}{\log_q^2 k}$. Then in the big O notation it can be simplified to $O\left(\frac{\log_q^2(\ell^2 k)}{\log_q^2(k)}\right) = O\left(\frac{\log_q^2 \ell}{\log_q^2 k}\right)$ and then to $O(\log_k^2 \ell)$ by changing the base to k . Therefore Problem 1 can be solved in polynomial time within a factor of $O(\log_k^2 \ell)$ for \mathcal{L}_ℓ^Σ and by Lemma 3 $O(\ell(\log_k^2(\ell)))$ for $\mathcal{L}_{\leq \ell}^\Sigma$. \square

Lemma 5. *There exists a polynomial time algorithm for the k -centre problem on $\mathcal{L}_\mathbf{P}^\Sigma$ that can ensure that no word in $\mathcal{L}_\mathbf{P}^\Sigma$ shares a common substring of length at least $\log_q(k(q-1)) - 1$ with the nearest centre. Furthermore, every word in $\mathcal{L}_\mathbf{P}^\Sigma$ will be no more than $\frac{2\ell^2}{(\log_q^2(k(q-1)))}$ from the nearest centre.*

Proof. This follows from the same methods given for \mathcal{L}_ℓ^Σ . Using the ranking function for fixed content necklaces given by a straight forward adaptation of the ranking function given by Hartman and Sawada [8], the set of prefixes may be generated in the same way as before. The primary difference in these settings is that the number of necklaces and prefixes thereof are considerably smaller. Once a set of prefixes is generated, the centres can be generated in the same way as before, with the added constraint that the word satisfies the Parikh vector.

Let λ be the length of the longest substring shared by the word in \mathcal{L}_ℓ^Σ that is furthest from one of the centres. By the same arguments as in Theorem 4, a lower bound of λ can be given as $\lambda \geq (\log_q(k(q-1))) - 1$. However, unlike in the general case, all necklaces start with the same first character, increasing the lower bound on λ to $\lambda \geq (\log_q(k(q-1)))$. Furthermore, as every word in $\mathcal{L}_\mathbf{P}^\Sigma$ shares the same number of occurrences of every character, therefore the size of the intersection in the overlap coefficient will be $\ell + \frac{(\log_q(k(q-1)))(\log_q(k(q-1))+1)}{2} - (\log_q(k(q-1)))$. Therefore the lower bound on the distance will be no more than $\frac{2\ell^2}{2\ell + (\log_q^2(k(q-1))) - 2(\log_q(k(q-1)))}$ and then bounded by $\frac{2\ell^2}{(\log_q^2(k(q-1)))}$. \square

Theorem 2. *Problem 1 can be solved for a language $\mathcal{L}_\mathbf{P}^\Sigma$ can be solved by an approximation factor of $O\left(\frac{q^{1/2} k \cdot \ell}{\log_q(k \cdot q)^2}\right)$.*

Proof. First we must establish a lower bound on the distance for this setting. Note than, unlike for \mathcal{L}_ℓ^Σ , every word contains the same set of characters. Therefore size of the intersection in the overlap co-efficient will be at least ℓ for every word. Moreover, if any character occurs more than $\frac{\ell}{2}$ times, there will be a subword of length 2 shared by every word.

Let λ be the length of the longest subword such that every word in $\mathcal{L}_{\mathbf{P}}^{\Sigma}$ share a subword of length at least λ with at least one of the centres. An upper bound on the length of λ comes from the number of prefixes to fixed content necklaces. For any $q \leq l \leq \ell$, there will be at least $\frac{l!}{\max(1, l-q+1)!}$ possible subwords of length l . As there are $k \cdot \ell$ possible subwords of length λ , this requires $k \cdot \ell \geq \frac{\lambda!}{\max(1, \lambda-q+1)!}$. assuming $\lambda \geq q$, $\frac{\lambda!}{\max(1, \lambda-q+1)!}$ may be approximated as $(\lambda - q + 1)^q$, giving $k \cdot \ell \geq (\lambda - q + 1)^q$ bounding λ as $\lambda \leq \sqrt[q]{k \cdot \ell} + q - 1$.

For any combination of subwords of length less than or equal to λ guaranteeing that every word in $\mathcal{L}_{\mathbf{P}}^{\Sigma}$ has an intersection of size $\lambda(\lambda + 1)$ requires $\lambda^2 \leq \sqrt[q]{k \cdot \ell} + q - 1$, however for any $\lambda \geq 2$, this contradicts the assumption that λ is the largest value such that $\lambda \leq \sqrt[q]{k \cdot \ell} + q - 1$. Therefore the size of the intersection must be less than or equal to $\ell - \lambda + (\lambda)(\lambda + 1)$, which by substituting the upper bound of $\sqrt[q]{k \cdot \ell} + q - 1$ gives $\ell - \sqrt[q]{k \cdot \ell} + q - 1 + (\sqrt[q]{k \cdot \ell} + q - 1)(\sqrt[q]{k \cdot \ell} + q)$. This gives an lower bound on the distance as $\frac{\ell^2}{\ell - \sqrt[q]{k \cdot \ell} + q - 1 + (\sqrt[q]{k \cdot \ell} + q - 1)(\sqrt[q]{k \cdot \ell} + q)}$.

To get the approximation ratio, the lower bound in the distance given in Lemma 5 by this upper bound gives $\frac{2\ell - 2\sqrt[q]{k \cdot \ell} + 2q - 2 + 2(\sqrt[q]{k \cdot \ell} + q - 1)(\sqrt[q]{k \cdot \ell} + q)}{\log_q^2(k(q-1))}$. Assuming that $\ell \leq (\sqrt[q]{k \cdot \ell} + q - 1)^2$, this can be simplified for big O notation to $O\left(\frac{q/2\sqrt[q]{k \cdot \ell}}{\log_q(k \cdot q)^2}\right)$. \square

3.2 Sampling with forbidden subwords

In order to generalise the algorithm described in Theorem 4, the ranking and unranking functions must be generalised to account for forbidden words. This is a much more challenging problem compared to the general case primarily due to the cyclic nature of the words. Unlike with a non-cyclic word, when counting the number of cyclic words without a given forbidden word, it must be ensured that it does not occur for *any* shift, as opposed to just one. This is further complicated when considering multiple forbidden words, where it must be checked that no forbidden word occurs for any rotation. Ruskey and Sawada [15] computed the size of $\mathcal{L}_{\ell}^{\Sigma}$ as $N_q^{\ell}(F) = \sum_{d|\ell} \phi(d) C_q^{\frac{\ell}{d}}(F)$, where

$C_q^{\ell}(F)$ is a function for counting the number of cyclic words of length ℓ containing no subword in F . This can be computed in polynomial time for a constant size of $|F|$. The number of Lyndon words of length ℓ with not containing any subword in F , denoted $L_q^{\ell}(F)$, is given relative to the number of Necklace, using the function:

$$L_q^{\ell}(F) = \sum_{d|\ell} \cdot \mu(d) N_q^{\ell/d}(F). \quad (2)$$

For the remainder of this section, let $\mathbf{N}_q^{\ell}(F)$ and $\mathbf{L}_q^{\ell}(F)$ denote the sets of necklaces and Lyndon words respectively.

Before introducing the function for ranking and unranking of forbidden words, some theoretical results must be established. Let $\mathbf{T}(w, F)$ be the set of words such that the canonical representation for each every word $v \in \mathbf{T}(w, F)$ is smaller than w , and no forbidden word in F occurs as a subword. Let $\langle x \rangle$ denote the canonical rotation of some word x , and let $f \not\subseteq w$ denote that f is not a subword of w . Using this notation we get $\mathbf{T}(w, F) = \{x \in \Sigma^{\ell} : \langle x \rangle < w, \forall f \in F, f \not\subseteq x\}$

Three further sets are needed for the purpose of ranking. The first of these is the set of aperiodic words such that the smallest rotation is less than some given word w , denoted $\mathbf{T}'(w, F) = \{x \in \Sigma^{\ell} : \langle x \rangle < w, \forall f \in F, f \not\subseteq x, x \text{ is aperiodic}\}$. Next is the set of words on length $l \leq n$ where the smallest rotation is less than w , denoted $\mathbf{T}_l(w, F) = \{x \in \Sigma^l : \langle x \rangle < w, \forall f \in F, f \not\subseteq x\}$. For two words x and w of lengths l and n respectively, $\langle x \rangle < w$ if and only if $\langle x^n \rangle < w^l$. The final set is

that of aperiodic words of a given length l where the canonical representation is less than, denoted $\mathbf{T}'_l = \{x \in \Sigma^l : \langle x \rangle < w, \forall f \in F, f \not\subseteq x, x \text{ is a Lyndon word}\}$.

Lemma 6. *For every d that is a factor of l , the size of $\mathbf{T}_l(w, F)$ is equal to $\sum_{d|l} |\mathbf{T}'_d(w, F)|$.*

Proof. Observe that every word in the set $\mathbf{T}_l(w, F)$ will either be aperiodic, in which case it will belong also to the set $\mathbf{T}'_l(w, F)$, or it will be periodic. If it is periodic, the period must be some value that is a factor of l . Given some word with a period d , if it is smaller than w , then it will occur in the set $\mathbf{T}'_d(w, F)$. By definition, any word greater than w will not occur in any set $\mathbf{T}'_b(w, F)$ for any b such that $\ell \bmod b \equiv 0$. As each set $\mathbf{T}'_d(w, F)$ consists only of aperiodic words, there can be no word that occurs in both $\mathbf{T}'_d(w, F)$ and $\mathbf{T}'_e(w, F)$ for $d \neq e$. Therefore the size of $\mathbf{T}_l(w, F)$ can be computed as $|\mathbf{T}_l(w, F)| = \sum_{d|l} |\mathbf{T}'_d(w, F)|$. \square

By application of the Möbius inversion formula to $|\mathbf{T}_l(w, F)| = \sum_{d|l} |\mathbf{T}'_d(w, F)|$, an equation for the size of $\mathbf{T}'_l(w, F)$ can be derived as:

$$|\mathbf{T}'_l(w, F)| = \sum_{d|l} \mu\left(\frac{\ell}{d}\right) |\mathbf{T}_d(w, F)|. \quad (3)$$

This can be used to rank some word w amongst the set of Lyndon words without any forbidden subword.

Lemma 7. *The number of Lyndon word smaller than some word w without any forbidden subword in F will be given by $\text{rank}_L(w, F) = \frac{1}{\ell} \cdot \mathbf{T}'(w, F)$.*

Proof. As every Lyndon word is aperiodic, it has n unique rotations. Therefore, for any given word w , each Lyndon word will occur n times within the set of aperiodic words with some rotation smaller than w , if and only if the canonical representation of the Lyndon word is smaller than w . \square

In the next lemma we compute the number of necklaces smaller than w using $\text{rank}_L(w, F)$.

Lemma 8. *The number of necklaces smaller than w without any forbidden subword in F is equal to $\text{rank}_N(w, F) = \sum_{d|l} \frac{1}{d} \cdot \mathbf{T}'_d(w, F)$.*

Proof. It follows from Lemma 6 that all necklaces smaller than some word will either be aperiodic, or periodic with a period that is a factor of the length of the necklace. From Lemma 7, the necklaces that are smaller than w and are aperiodic are $\frac{1}{\ell} \cdot \mathbf{T}'_d(w, F)$. Similarly, the necklaces with a period of some factor d of n are $\frac{1}{\ell} \cdot \mathbf{T}'_d(w, F)$. \square

The problem now becomes computing $|\mathbf{T}_l(w, F)|$. To do this, the set will be partitioned to the set $\mathbf{A}_w(t, j, F)$ such that for every word $v \in \mathbf{A}_w(t, j, F)$ the following hold.

- t is the smallest cyclic shift such that shifting v by t , denoted $v \cdot t$, makes the resulting word smaller than w , i.e. $v \cdot t < w$.
- Under the shift by t , j is the length of the longest prefix of $v \cdot t$ that is also a prefix of j .

Lemma 9. *The size of $\mathbf{A}_w(t, j, F)$ can be computed in $O(q^{\ell|F|+2})$ time.*

Proof. This can be done by considering two possible cases for the set. First is the case $t + j \leq \ell$. In this case, every word will be of the form $\beta w_1 \dots w_j x \rho$ where:

- β is some word of length t with no forbidden subword such that every suffix is greater than w ;
- $w_1 \dots w_j$ is the prefix of w of length j ;
- x is a character smaller than w_{j+1} ;
- ρ is a word with no restrictions other than having no forbidden substrings.

To compute the number of possible words satisfying ρ and β , we define the function $B'_\alpha(l, j, t, P, S)$. A full definition of this function is given in Appendix A. At a high level, this function works by recursively checking how many possible ways of extending the string based on the sets P and S . S represents the set of suffixes of α that are prefixes of $w_1 \dots w_j$. P initially represents the set of prefixes of α that are suffixes of $\alpha_1 \dots \alpha_j$. As this function can be computed recursively, the computational complexity will be equal to the number of potential calls to B' . There are ℓ possible value for l and $t + j$, and $\ell^{|F|}$ for both S and P . This gives a total time complexity of This will take $O(q\ell^{|F|+3})$ time to compute in the worst case. Alongside this, two auxiliary functions $\theta(w_1 \dots w_j \sigma)$ and $\Omega(w_1 \dots w_j \sigma)$ are needed. These, respectively, compute the sets of prefixes and suffixes of forbidden words of $w_1 \dots w_j \sigma$, for some character σ . Using the above, we get:

$$|\mathbf{A}_w(t, j, F)| = \sum_{\sigma=1}^{w_{j+1}-1} B'_w(\ell - j - 1, 0, \ell - (j + t + 1), \theta(w_1 \dots w_j \sigma), \Omega(w_1 \dots w_j \sigma))$$

In the second case, every word will be of the form $w_s w_{s+1} \dots w_j x \beta w_1 \dots w_{s-1}$. Let δ be the length of the longest prefix of w that is a suffix $w_s \dots w_j$. If $x < w_{\delta+1}$, then the shift by $s - \delta$ would be smaller than α . Therefore x must be greater than or equal to $w_{\delta+1}$. From this, the size of $\mathbf{A}_w(t, j, F)$ can be computed as: $|\mathbf{A}_w(t, j, F)| = B'_w(\ell - j - 1, \delta + 1, 0, \theta(w_1 \dots w_{\delta+1}), \Omega(w_1 \dots w_{\delta+1})) + \sum_{\sigma=w_{\delta+1}+1}^{w_{j+1}-1} B'_w(\ell - j - 1, 0, 0, \theta(w_1 \dots w_{\delta+1}), \Omega(w_1 \dots w_{\delta+1}))$

As $B'_\alpha(l, j, t, P, S)$ can be computed in $O(q\ell^{|F|+2})$ and stored for all value of values of l, j, t, P and S , the time to compute the size of $\mathbf{A}_w(t, j, F)$ will be $O(q)$. As this is dominated by the time to compute B' , the total time will be $O(q\ell^{|F|+3})$. \square

Theorem 3. *The rank of a word amongst all necklaces without any forbidden subword may be computed in $O(q\ell^{|F|+2} \log_2^2(\ell))$.*

Proof. It follows from Lemma 9 that by separately computing the values of B' , the time to compute the size of $\mathbf{A}_w(j, t, F)$ will be $O(q\ell^{|F|+2})$. Following Lemma 8, the number of Necklaces may be computed by summing the size of $\mathbf{T}'_d(w, F)$ for every factor d of ℓ . Note that there are at most $\log_2(\ell)$ factors of ℓ . The size of $\mathbf{T}'_d(w, F)$ can be computed using Lemma Equation 3. In the worst case there will be $O(\log_2 \ell)$ sets of $\mathbf{T}'_d(w, F)$, each of which taking at most $O(q\ell^{|F|+2})$ time to compute. Putting this together, the total time complexity will be $O(q\ell^{|F|+2} \log_2^2(\ell))$ \square

Theorem 4. *For a constant number of forbidden words, Problem 1 for $\mathcal{L}_\ell^{\Sigma \setminus F}$ can be solved in polynomial time with an approximation factor of $O(\log_k^2(\ell))$.*

Centre	Word
	000000100001100010100011100100101100110100111101010111011011111
1	000000100001100010100
2	101000111001001011001
3	110011010011110101011
4	010110110111111000000

Figure 3: Example of how to split the de Bruijn sequence of order 6 between 4 samples. Highlighted parts are the shared subwords between two samples.

Proof. The same approach given in Section 3.1 may be used to build centres. In this case the ranking function described in this section may be used in place of the ones used there. Clearly the same bounds on length of the prefix will hold in the worst case, giving an upper bound on the distance of $\frac{2\ell^2}{\log_q^2(k)}$. A lower bound follows from the same observation that the length of the longest common subword between the furthest word in the language and the nearest centre will be bound from above by the number of Lyndon words without any forbidden subwords. A bound on this is $\frac{q^\lambda - (\ell|F|q^{\lambda-2})}{\ell}$, which will be of order $O(\frac{q^\lambda}{\ell})$ for a constant size $|F|$. Thus the approximation factor between the bound given by this algorithm will be the same as in the unconstrained case, giving a factor of $O(\log_k^2(\ell))$. \square

4 Sampling via de Bruijn sequences

The primary issue with the prenecklace based algorithm is that it does not take advantage of any additional space left in the samples. As such a different approach will have to be considered to build the samples. Following the same motivation of maximising the length of the subword shared between every word in the language and the nearest centre, observe that this requires every subword must occur at some point in the sample.

For a given length λ , there are q^λ subwords. A de Bruijn sequence of order λ is a word of length q^λ where every word of length λ occurs as a subword. This makes it a natural candidate for use as the basis for the centres.

Lemma 10. *There exists an algorithm with a worst case running time of $O(\ell k)$ for the k -centre problem on \mathcal{L}_ℓ^Σ ensuring that every word in \mathcal{L}_ℓ^Σ shares a common substring of length at least $\log_q(k)$. Further this will ensure that no word in \mathcal{L}_ℓ^Σ is a distance of more than $\frac{2\ell^2}{\log_q^2(k\ell)}$ from the nearest centre.*

Proof. The main idea of this algorithm is to take the de Bruijn sequence of order λ and divide it between samples while ensuring that all subwords of length λ occur at some point as a subword of a sample. Note that the length of the de Bruijn sequence of order λ will be q^λ . The de Bruijn sequence may be efficiently generated in time linear to the length of the sequence [15], which must be less than $\ell \times k$.

Naively splitting the sequence between the k centres may lead to subwords being lost. In order to account for this, the sequence may be split into samples of size $\ell - \lambda + 1$. The first centre can be generated by taking the first ℓ characters of the de Bruijn sequence. To ensure that every subword of length λ occurs, the first $\lambda - 1$ characters of the second centre will be the same as the last $\lambda - 1$ character of the first centre. Repeating this, the i^{th} centre will be the subword of length ℓ starting at position $i(\ell - \lambda)$ in the de Bruijn sequence. An example of this is given in Figure 3.

To determine the length of λ relative to k and ℓ , note that the size of the corresponding de Bruijn sequence must be small enough such that every subword may occur. Formally, for λ to be feasible, $q^\lambda \leq k(\ell - \lambda + 1)$. This may be rearranged in terms of λ , giving as an upper bound $\leq \log_q(k\ell)$.

Using these, the centres may be made be formed by taking each of these samples, and appending the first $\lambda - 1$ of the next sample. For a given k centres of length at most ℓ , λ will be the largest value such that $q^\lambda \leq k(\ell - \lambda + 1)$, which may be rewritten as $\lambda \leq \log_q(k(\ell - \lambda + 1))$; $\lambda \leq \log_q(k\ell)$. An upper bound on the distance may be gained using Lemma 1, giving $\frac{2\ell^2}{(\log_q(k\ell))(\log_q(k\ell)+1)}$, which may be bounded by $\frac{2\ell^2}{\lceil \log_q^2(k\ell) \rceil}$.

In terms of complexity, there are known algorithms to output the de Bruijn sequence in $O(q^\lambda)$ time, which by the definition of λ will be $O(k\ell)$ time in the worst case. From the sequence, the process of dividing into k samples will take no more than time linear to the size of the sequence, giving a total complexity of $O(k\ell)$. It is worth noting that any algorithm that outputs the centres will have a complexity of at least $O(k\ell)$. \square

Theorem 5. *The algorithm described in Lemma 10 will approximate the optimal solution within a factor of 8.*

Proof. Recall from Lemma 1 that the minimum distance the word that is furthest from the sample can be is $\frac{\ell^2}{\log_q^2(k\ell^2)}$. The upper bound on distance given by Lemma 10 is $\frac{2\ell^2}{\log_q^2(k\ell)}$. Dividing this upper bound by the lower bound gives $\frac{2\log_q^2(k\ell^2)}{\log_q^2(k\ell)}$. Rewriting this in terms of base k gives $\frac{2\log_k^2(k\ell^2)}{\log_k^2(k\ell)} = 2\left(\frac{\log_k(k\ell^2)}{\log_k(k\ell)}\right)^2 = 2\left(\frac{1+\log_k(\ell^2)}{1+\log_k(\ell)}\right)^2 = 2\left(\frac{1+2\log_k(\ell)}{1+\log_k(\ell)}\right)^2 = 2\left(\frac{1+2\log_k(\ell)+1-1}{1+\log_k(\ell)}\right)^2 = 2\left(2 - \frac{1}{1+\log_k(\ell)}\right)^2$. If $\left(2 - \frac{1}{1+\log_k(\ell)}\right)^2 \leq 4$ then this algorithm will approximate the optimal solution within a factor of 8. As $\frac{1}{1+\log_k(\ell)}$ will be no more than 1, this will hold. \square

References

- [1] Duncan Adamson, Argyrios Deligkas, Vladimir V. Gusev, and Igor Potapov. On the hardness of energy minimisation for crystal structure prediction. In Alexander Chatzigeorgiou, Riccardo Dondi, Herodotos Herodotou, Christos A. Kapoutsis, Yannis Manolopoulos, George A. Papadopoulos, and Florian Sikora, editors, *SOFSEM 2020: Theory and Practice of Computer Science - 46th International Conference on Current Trends in Theory and Practice of Informatics, SOFSEM 2020, Limassol, Cyprus, January 20-24, 2020, Proceedings*, volume 12011 of *Lecture Notes in Computer Science*, pages 587–596. Springer, 2020.
- [2] Dmytro Antypov, Argyrios Deligkas, Vladimir Gusev, Matthew J. Rosseinsky, Paul G. Spirakis, and Michail Theofilatos. Crystal structure prediction via oblivious local search. *CoRR*, abs/2003.12442, 2020.
- [3] William W Cohen, Pradeep Ravikumar, Stephen E Fienberg, et al. A comparison of string distance metrics for name-matching tasks. In *IIWeb*, volume 2003, pages 73–78, 2003.
- [4] C. Collins, M. S. Dyer, M. J. Pitcher, G. F. S. Whitehead, M. Zanella, P. Mandal, J. B. Claridge, G. R. Darling, and M. J. Rosseinsky. Accelerated discovery of two crystal structure types in a complex inorganic phase field. *Nature*, 546(7657):280–284, jun 2017.

- [5] Matthew S. Dyer, Christopher Collins, Darren Hodgeman, Philip A. Chater, Antoine Demont, Simon Romani, Ruth Sayers, Michael F. Thomas, John B. Claridge, George R. Darling, and Matthew J. Rosseinsky. Computationally assisted identification of functional inorganic materials. *Science*, 340(6134):847–852, 2013.
- [6] Andreas Emil Feldmann and Dániel Marx. The parameterized hardness of the k-center problem in transportation networks. *Algorithmica*, 2020.
- [7] Leszek Gasieniec, Jesper Jansson, and Andrzej Lingas. Efficient approximation algorithms for the hamming center problem. In Robert Endre Tarjan and Tandy J. Warnow, editors, *Proceedings of the Tenth Annual ACM-SIAM Symposium on Discrete Algorithms, 17-19 January 1999, Baltimore, Maryland, USA*, pages 905–906. ACM/SIAM, 1999.
- [8] Patrick Hartman and Joe Sawada. Ranking and unranking fixed-density necklaces and Lyndon words. *Theoretical Computer Science*, 791:36–47, oct 2019.
- [9] Yishan Jiao, Jingyi Xu, and Ming Li. On the k-closest substring and k-consensus pattern problems. In Suleyman Cenk Sahinalp, S. Muthukrishnan, and Ugur Dogrusoz, editors, *Combinatorial Pattern Matching*, pages 130–144, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.
- [10] Swastik Kopparty, Mrinal Kumar, and Michael Saks. Efficient Indexing of Necklaces and Irreducible Polynomials over Finite Fields. *Theory of Computing*, 12(1):1–27, 2016.
- [11] J. Kevin Lanctot, Ming Li, Bin Ma, Shaojiu Wang, and Louxin Zhang. Distinguishing string selection problems. *Inf. Comput.*, 185(1):41–55, August 2003.
- [12] Ming Li, Bin Ma, and Lusheng Wang. On the closest string and substring problems. *J. ACM*, 49(2):157–171, March 2002.
- [13] Jakub Piskorski, Marcin Sydow, and Karol Wielech. Comparison of string distance metrics for lemmatisation of named entities in polish. In *Language and Technology Conference*, pages 413–427. Springer, 2007.
- [14] Gabriel Recchia and Max M Louwerse. A comparison of string similarity measures for toponym matching. In *COMP@ SIGSPATIAL*, pages 54–61, 2013.
- [15] Frank Ruskey and Joe Sawada. Generating necklaces and strings with forbidden substrings. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 1858, pages 330–339. Springer Verlag, 2000.
- [16] Joe Sawada and Aaron Williams. Practical algorithms to rank necklaces, Lyndon words, and de Bruijn sequences. *Journal of Discrete Algorithms*, 43:95–110, mar 2017.

A Ranking functions

The function $B'_\alpha(l, j, t, P, S)$ can be thought of as counting the size of the set of words where, for every v in the set:

- v has length t .
- For every $p \in P$, and $s \in S$, no forbidden subword occurs in the word pvs .
- The first j characters of $v = w_1w_2 \dots w_j$.
- Every suffix of the subword $v_{t-l}v_{t-l+1} \dots v_t$ is greater than w .

To compute $B'_\alpha(l, j, t, P, S)$, two auxiliary functions will be introduced to compute the P and S after the next character σ is introduced. $\theta(P, \sigma)$ will take as argument the set of prefixes P and some character σ , and returning the set of prefixes of F where either σ is the first character of a forbidden word or there exists some $p \in P$ where $p\sigma \subseteq F$ - i.e. the prefixes that may be continued by adding this character. $\Omega(S, \sigma)$ will return the members of S that are suffixes of some forbidden word, and the words $s\sigma$ for $s \in S$ where $s\sigma$ is a subword of a forbidden word. To compute $B'_\alpha(l, j, t, P, S)$, the set may be further partitioned by the next character σ . If $l > 0$, then there is no lower bound on the value of σ , otherwise σ must be greater than or equal to α_{j+1} . For each σ the number of words for which the $(j+1)^{th}$ character equals σ will be as follows:

- If there exists some $p \in P$ such that $p\sigma = f$ for some forbidden word f then there will be no subsequent words.
- Otherwise, if $t - j = 1$ then if there exists some $p \in P$ and some $s \in S$ where $p\sigma s = f$ or $\sigma s = f$ then there will also be no words in this set, otherwise there will be only 1 word.
- If $l > 0$ and $t - j > 1$ then the size of the set will be equal to $B'_\alpha(l-1, t-t, j, \theta(P, \sigma), \Omega(S, \sigma))$.
- If $l = 0$ and $\sigma > \alpha_{j+1}$ then the size of the set will be equal to the size of the set $B'_\alpha(0, t-1, 0, \theta(P, \sigma), \Omega(S, \sigma))$.
- Otherwise, $l = 0$ and $\sigma = \alpha_{j+1}$, therefore the size of this set will be equivalent to the set $B'_\alpha(0, t, j+1, \theta(P, \sigma), \Omega(S, \sigma))$.

Therefore $B'_\alpha(l, t, j, P, S)$ may be computed recursively as follows: $B'_\alpha(l, t, j, P, S) =$

$$= \begin{cases} 0 & t = j > 0 \\ 1 & t = j = 0 \\ \sum_{\sigma=1}^q \begin{cases} B'_\alpha(l-1, t-1, 0, \theta(P, \sigma), \Omega(S, \sigma)) & \nexists p \in P, f \in F \text{ s.t. } p\sigma = f \\ 0 & \text{Otherwise.} \end{cases} & l > 0, t > 1 \\ \left(\sum_{\sigma=\alpha_j+1}^q \begin{cases} B'_\alpha(l-1, t-1, 0, \theta(P, \sigma), \Omega(S, \sigma)) & \nexists p \in P, f \in F \text{ s.t. } p\sigma = f \\ 0 & \text{Otherwise.} \end{cases} \right) \\ + \left(\begin{cases} B'_\alpha(0, t, j+1, \theta(P, \alpha_{j+1}), \Omega(S, \alpha_{j+1})) & \nexists p \in P, f \in F \text{ s.t. } p\sigma = f \\ 0 & \text{Otherwise.} \end{cases} \right) & l = 0, t - j > 1 \\ \sum_{\sigma=1}^q \begin{cases} 1 & \nexists p \in P \cup \{\emptyset\}, f \in F, s \in S \cup \{\emptyset\} \text{ s.t. } p\sigma s = f \\ 0 & \text{Otherwise.} \end{cases} & l > 0, t = 1 \\ \left(\sum_{\sigma=\alpha_{j+1}+1}^q \begin{cases} 1 & \nexists p \in P \cup \{\emptyset\}, f \in F, s \in S \cup \{\emptyset\} \text{ s.t. } p\sigma s = f \\ 0 & \text{Otherwise.} \end{cases} \right) \\ + \left(\begin{cases} 1 & \nexists p \in P \cup \{\emptyset\}, f \in F, s \in S \cup \{\emptyset\} \text{ s.t. } p\alpha_{j+1}s = f \\ 0 & \text{Otherwise.} \end{cases} \right) & \text{Otherwise.} \end{cases}$$

Note that there are at most ℓ possible values for l, t and j , and $\ell^{|F|}$ possible values for P and S . In the worst case each of these must be computed. Assuming for some given arguments that the values of B'_α for each character has been computed, then it will take $O(q)$ time to compute the value of B'_α for these arguments.

B Prefix algorithm

Algorithm 1 Prefix algorithm

```
1: procedure PREFIX( $\Sigma, n, k$ )
2:   Set of current prefixes,  $P$ 
3:   Set of new prefixes,  $P'$ 
4:   Prefix in  $P$ ,  $p$ 
5:   Character in  $\Sigma$ ,  $\beta$ 
6:    $P \leftarrow \emptyset$ 
7:    $P' \leftarrow \emptyset$ 
8:   while  $|P| < N_k$  do
9:      $P' \leftarrow \emptyset$ 
10:    for  $p \in P$  do
11:      for  $\beta \in \Sigma$  do
12:        if  $\text{rank}(\min(p\beta)) \neq \text{rank}(\max(p\beta))$  then
13:           $P' \leftarrow P \cup \{p\beta\}$ 
14:        end if
15:        if  $|P \cup P'| = k$  and  $\beta \neq |\Sigma|$  then
16:          return  $P \cup P'$ 
17:        end if
18:      end for
19:       $P \leftarrow P \setminus p$ 
20:      if  $|P \cup P'| = k$  then
21:        return  $P \cup P'$ 
22:      end if
23:    end for
24:     $P \leftarrow P'$ 
25:  end while
26:   $\text{Prefixes} \leftarrow \emptyset$ 
27:  for  $p \in P'$  do
28:     $\text{Prefixes} \leftarrow \text{Prefixes} \cup \{\text{unrank}(\min(p) + \lfloor \frac{\min(p) + \max(p)}{2} \rfloor)\}$ 
29:  end for
30:  return  $P$ 
31: end procedure
```
