Efficiency of Learned Indexes on Genome Spectra

Md. Hasin Abrar[†]
Department of Computer Science and Engineering, Penn State, University Park, PA, USA

Paul Medvedev[†]
Department of Computer Science and Engineering, Penn State, University Park, PA, USA
Department of Biochemistry and Molecular Biology, Penn State, University Park, PA, USA
Huck Institutes of the Life Sciences, Penn State, University Park, PA, USA

Giorgio Vinciguerra[†]
Department of Computer Science, University of Pisa, Italy

Abstract

Data structures on a multiset of genomic k-mers are at the heart of many bioinformatic tools. As genomic datasets grow in scale, the efficiency of these data structures increasingly depends on how well they leverage the inherent patterns in the data. One recent and effective approach is the use of learned indexes that approximate the rank function of a multiset using a piecewise linear function with very few segments. However, theoretical worst-case analysis struggles to predict the practical performance of these indexes.

We address this limitation by developing a novel measure of piecewise-linear approximability of the data, called CaPLa (Canonical Piecewise Linear approximability). CaPLa builds on the empirical observation that a power-law model often serves as a reasonable proxy for piecewise linear-approximability, while explicitly accounting for deviations from a true power-law fit. We prove basic properties of CaPLa and present an efficient algorithm to compute it. We then demonstrate that CaPLa can accurately predict space bounds for data structures on real data. Empirically, we analyze over 500 genomes through the lens of CaPLa, revealing that it varies widely across the tree of life and even within individual genomes. Finally, we study the robustness of CaPLa as a measure and the factors that make genomic k-mer multisets different from random ones.

2012 ACM Subject Classification Applied computing → Bioinformatics; Applied computing → Computational biology; Theory of computation → Data structures design and analysis

Keywords and phrases Genome spectra, piecewise linear approximation, learned index, k-mers

Digital Object Identifier 10.4230/LIPIcs.ESA.2025.18

Related Version A complete version of this work, including full proofs, is available but omitted here due to space limits. A preliminary version appeared in a preprint but is subsumed by this paper.

Full Version: `https://www.biorxiv.org/content/10.1101/2025.07.10.664199v1` [3]
Previous Version: `https://www.biorxiv.org/content/10.1101/2024.02.08.579510v1` [1]

Supplementary Material *Software (Source Code)*: `https://github.com/medvedevgroup/CaPLa` archived at `swh:1:dir:da45f156bdafa582fd16f04690ee49e184bf3590`

Funding This material is based upon work supported by the NSF under Grants No. DBI2138585 and OAC1931531. Research reported in this publication was supported by the National Institute Of General Medical Sciences of the NIH under Award Number R01GM146462. The content is solely the responsibility of the authors and does not necessarily represent the official views of the NIH. GV was supported by the NextGenerationEU – National Recovery and Resilience Plan (Piano Nazionale di Ripresa e Resilienza, PNRR) – Project: "SoBigData.it - Strengthening the Italian RI for Social Mining and Big Data Analytics" – Prot. IR0000013 – Avviso n. 3264 del 28/12/2021.

Acknowledgements We thank anonymous reviewers for helping improve various versions of this paper.

[†] The authors are listed in alphabetical order.

1 Introduction

Data structures on k-mers (fixed-length strings) originating from genomes form a crucial component of many bioinformatics tools [22]. Their space usage is now a major bottleneck at the forefront of biological discovery. For example, the Logan project needed two petabytes to index the 31-mers from the Sequence Read Archive [7]. Existing data structures often meet the space challenge by exploiting the non-uniform structure of genomic k-mer sets [6]. In doing so, they circumvent [2, 17, 23, 28] the theoretical worst- and average-case lower bounds, which offer a more pessimistic perspective [25].

A popular approach is based on learned indexes, a recent research direction that takes advantage of the internal patterns in the rank curve of a dataset [5, 10, 18, 32]. The rank of an element x in a multiset S is defined as the position of the first occurrence of x in the sorted list of S. A learned index is constructed from S and, given a query k-mer, returns an error-bounded prediction for its rank. Initial approaches used machine learning to construct the index [13, 14, 15], but it later turned out that in this setting, it is more effective to model the rank curve using a piecewise linear approximation [2, 4, 17], as depicted in Figure 1.

Learned indexes based on piecewise linear approximations (PLAs) of the rank curve have resulted in practically groundbreaking time/space tradeoffs, both on genomic [2, 8, 17] and other kinds of data [8, 11, 12, 16, 19, 20, 21, 31]. However, existing theoretical worst-case analyses only show improved performance under assumptions that are unrealistic for genomic datasets, such as the independence and identical distribution of gaps between k-mers [9]. This leaves a gap between the theoretical analysis of k-mer-based learned indexes and their much better performance on real data [2, 17]. By better understanding and characterizing the structural properties of genomic k-mer sets, we can not only better predict the performance of learned indexes but also drive the development of new PLA-based tools that maximally exploit the properties of real data.

Our approach in this paper is to develop a measure of piecewise linear (PL) approximability which can be used to parameterize the theoretical analysis of PLA-based methods. Parameterizing string complexity is an active area of research [26], but none of the existing measures capture the PL-approximability of a genome's rank curve. Previous papers have proposed modeling the PL-approximability of S using a power-law [9, 11], and have implicitly used the two parameters of a power-law fit as proxies for the PL-approximability of S. However, the interpretation of such parameters is limited when the rank curve strays from a power-law, as (we will show) occurs in real genomic data.

We develop a novel measure of PL-approximability, which we call CaPLa (Canonical PL-approximability) (Section 2). It builds on the idea of a power-law fit but adds the uncertainty of the fit as part of the measure. We prove basic properties of CaPLa, including its existence, and give an algorithm to compute it (Section 3). The algorithm is exact under certain conditions of the data, which we empirically show are nearly always met in practice. We demonstrate how CaPLa can be applied to derive space bounds that accurately reflect properties of real data (Section 4).

Finally, we apply CaPLa to analyze genome spectra, where a spectrum is the multiset of all k-mers appearing in a string.[1] We use a dataset of more than 500 genomes and find that CaPLa varies greatly across the tree of life and even within individual genomes (Section 5). We also show that CaPLa is a good predictor of the space usage of a learned index data structure (the PLA-index [2]). We use controlled experiments to elucidate what

[1] Our tool for computing CaPLa is available at `https://github.com/medvedevgroup/CaPLa`.

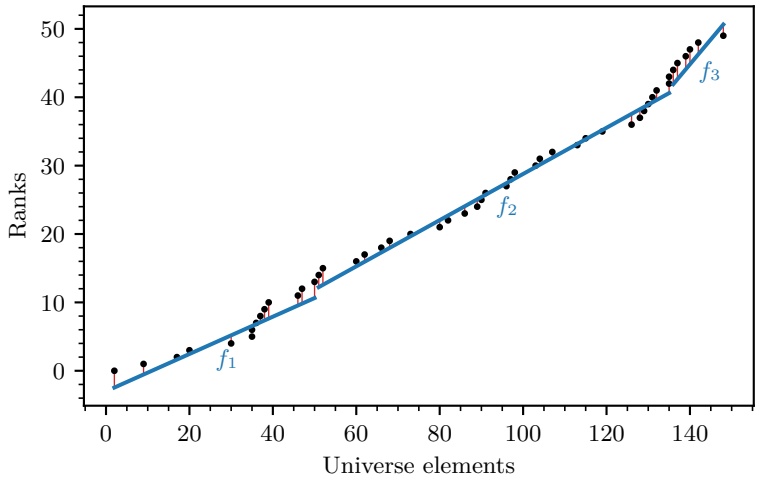

Figure 1 An example multiset S of size $N = 50$ with $n = 48$ distinct elements drawn from a universe of size $u = 150$, plotted as points $\{(x, \text{RANK}_S(x)\}_{x \in S}$ on the plane. The points are approximated using a PLA with error bound $\varepsilon = 2$, where the vertical orange lines show the error. The PLA uses 3 segments, which is the smallest possible for S at $\varepsilon = 2$, so $b(\varepsilon) = 3$.

makes genome spectra different from random k-mer multisets. Finally, we verify that CaPLa has some of the desired properties of a measure of genome complexity, namely robustness to variation in genome or k-mer size. Our work helps reduce the gap between theory and practice and aims to deepen our understanding of when and how to best apply PLA-based indexes to real genomic data.

2 PL-approximability definitions and motivation

In this section, we present the key definitions of PL-approximability and the motivation behind them. We start with a multiset S of elements from an integer universe $[u] = \{0, 1, \ldots, u-1\}$. In the case of k-mers, $u = 4^k$, though our definitions work for any u. We will use the notation that n is the number distinct and N is the total number of elements in S. The function we will be approximating is defined as $\text{RANK}_S(x) = |\{y \in S \mid y \leq x\}|$, for all $x \in [u]$. The key tool in designing PLA-based learned data structures [2, 8, 11, 12, 17] is the following:

▶ **Definition 1** (Piecewise linear ε-approximation). *For a given positive integer ε, a* piecewise linear ε-approximation (PLA) *of S is a partition of $[u]$ into subintervals such that, for each subinterval $[a_i, b_i]$, there exists a segment (i.e., a linear function) f_i such that $|f_i(x) - \text{RANK}_S(x)| \leq \varepsilon$ for all $x \in [a_i, b_i]$.*

Figure 1 shows an example of a PLA. The usefulness of a PLA comes from the fact that a data structure can avoid storing the full rank function and instead just store the PLA, as long as it has a way to handle the uncertainty due to the error of the approximation. To this end, the PLA with the smallest number of segments is the most space-efficient representation. Such a PLA can also be computed efficiently [27]. As a result, minimum-sized PLAs have formed the basis of several data structures [2, 8, 11, 19, 21, 31]. We formalize it with the following definition.

▶ **Definition 2** (PLA-size). *The* PLA-size *of S is the function $b\colon \mathbb{Z}^+ \to \mathbb{Z}^+$ mapping a positive integer ε to the smallest integer $b(\varepsilon)$ such that there exists a piecewise linear ε-approximation of S using exactly $b(\varepsilon)$ segments.*

In this paper, we introduce a closely related measure, but one that factors out the effect of the data size.

▶ **Definition 3** (PL-approximability). *The* PL-approximability *of S is the function mapping a positive integer ε to $\frac{n}{b(\varepsilon)}$, where $b(\varepsilon)$ is the PLA-size of S.*

The PL-approximability of S can be interpreted as the longest average number of elements spanned by a segment in a PLA of S, so it directly quantifies the efficiency of a PLA in capturing the structure and distribution of the underlying data.

The space and time performance of learned data structures based on PLAs are often bounded in terms of the PLA-size [2, 8, 11]. Typically, smaller values of ε lead to faster query times, but require a larger number of segments $b(\varepsilon)$ and thus increased space. The PLA-size and PL-approximability can then be tabulated for a given dataset and plugged into these space- and time-bounds to assess the resulting space-time trade-off. However, this approach provides little insight into broader trends and differences among data distributions. Ideally, the PL-approximability could be modeled as a parametrized family of functions, such that fitting this family to S yields parameters that fully capture the PL-approximability of S.

A key difficulty is that the shape of the PLA-size function varies significantly depending on the structure and distribution of the underlying data. In general, the PLA-size is bounded between $b(\varepsilon) = 1$ and $b(\varepsilon) = N/(2\varepsilon)$ for all ε [11]. For sets where the gaps between elements are random and independently drawn, the PLA-size has been shown to be $b(\varepsilon) = \mathcal{O}(N/\varepsilon^2)$ [9]. For real data, the PLA-size rarely admits a closed-form expression, but power-law relations of the form $b(\varepsilon) = n/(\beta\varepsilon^\alpha)$ have been empirically observed [9, 11]. Here, β and α are dataset-specific constants found through fitting to the tabulated PLA-size. This power-law modeling provides a degree of generality, but when the PLA-size does not exactly follow a power-law, it has several drawbacks. Figure 2 demonstrates that this type of modeling 1) has a high variability in the error, 2) is not robust to the choice of fitting algorithm, and 3) is unpredictable in whether it over- or under-estimates the true PLA-size. Such drawbacks make it challenging to incorporate this type of model into downstream analyses of data structures, especially since its predictions are neither worst- nor average-case bounds.

To address this challenge, we propose bounding the PLA-size using two power laws instead of relying on a single fitted curve. We first fix a finite set $\mathcal{E} \subset \mathbb{Z}^+$ of ε values for which the PL-approximability is evaluated. This reflects practical applications, where ε is chosen from a set of values according to the desired space-time trade-off.

▶ **Definition 4** (Bounded PL-approximability). *The PL-approximability of S is power-law bounded over $\mathcal{E}$ with parameters $(\alpha, \beta_{\text{low}}, \beta_{\text{high}})$ if*

$$\beta_{\text{low}}\varepsilon^\alpha \leq \frac{n}{b(\varepsilon)} \leq \beta_{\text{high}}\varepsilon^\alpha \quad \text{for all } \varepsilon \in \mathcal{E}. \tag{1}$$

When $\beta_{\text{low}} = \beta_{\text{high}}$, this is equivalent to a perfect power-law fit. The role of α is to capture the rate of segment growth as a function of ε. If $\beta_{\text{low}} = \beta_{\text{high}}$, then when ε doubles, the average segment length increases by a factor of 2^α. There is similarly an intuitive interpretation of β_{low} and β_{high}: when $\varepsilon = 1$, the average segment length is between β_{low} and β_{high}.

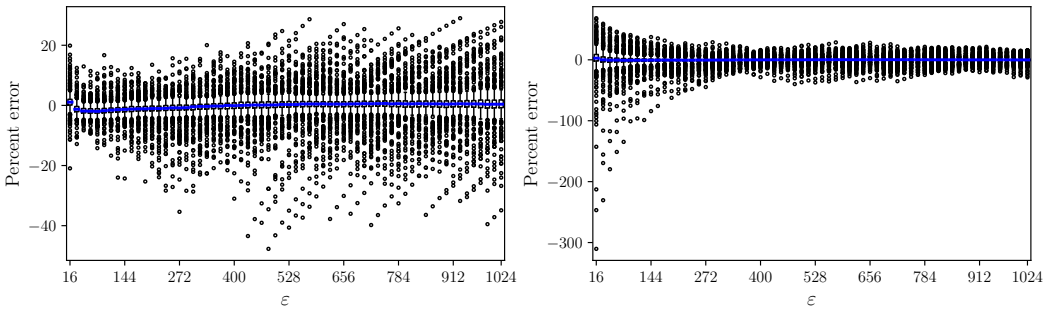

Figure 2 Errors due to fitting a single power-law curve to the PLA-size of a set of 513 genomes (dataset details are in Section 5.1). The left panel models the relationship as $n/b(\varepsilon) = \beta\varepsilon^{\alpha}$ and uses Python's *scipy.optimize.curve_fit* with the Levenberg–Marquardt algorithm to estimate α and β. The right panel uses the model $\log(n/b(\varepsilon)) = \alpha \log \varepsilon + \log \beta$ and applies Python's *scipy.stats.linregress* to estimate α and β. For each $\varepsilon \in \{16, 32, 48, 64, \ldots, 1024\}$, we calculate the percent error between $b(\varepsilon)$ and the prediction $n/(\beta\varepsilon^{\alpha})$ and show a box plot of the distribution aggregated over the 513 genomes (with the median line colored in blue).

Definition 4 captures a full spectrum of data distributions and their PLA-size, from the best case $b(\varepsilon) = 1$ (with $\alpha = 0$ and $\beta_{\text{low}} = \beta_{\text{high}} = n$), to the worst case $b(\varepsilon) = N/(2\varepsilon)$ from [11] (with $\alpha = 1$ and $\beta_{\text{low}} = \beta_{\text{high}} = 2n/N$), to the case of random gaps from [9] (with $\alpha = 2$ and $\beta_{\text{low}} = \beta_{\text{high}} = cn/N$ for a constant $c > 0$). More importantly, it enables bounding the PL-approximability of real datasets where an exact power-law fitting is not possible.

The PL-approximability of a single dataset can be power-law bounded for infinitely many values of $\alpha, \beta_{\text{low}}$, and β_{high}. To capture a single parameterization that can be used as a proxy for the PL-approximability of a dataset, we introduce the following notion of *canonical PL-approximability*.

▶ **Definition 5** (CaPLa). *A pinch point of S is any α^* that satisfies*

$$\alpha^* = \operatorname*{argmin}_{\alpha \geq 0} \big(H(\alpha) - L(\alpha)\big), \quad \textit{where} \quad L(\alpha) = \min_{\varepsilon \in \mathcal{E}} \frac{n}{\varepsilon^{\alpha} b(\varepsilon)}, \quad H(\alpha) = \max_{\varepsilon \in \mathcal{E}} \frac{n}{\varepsilon^{\alpha} b(\varepsilon)}. \tag{2}$$

If a unique pinch point α^* exists, then we let $\beta^*_{\text{low}} = L(\alpha^*)$ and $\beta^*_{\text{high}} = H(\alpha^*)$ and say that S has a canonical PL-approximability (CaPLa) of $(\alpha^*, \beta^*_{\text{low}}, \beta^*_{\text{high}})$. Otherwise, we say that the CaPLa of S is undefined.

We note that if S has CaPLa $(\alpha^*, \beta^*_{\text{low}}, \beta^*_{\text{high}})$, then Equation 1 immediately implies that S is power-law bounded with parameters $(\alpha^*, \beta^*_{\text{low}}, \beta^*_{\text{high}})$.

3 Existence and computation of the canonical PL-approximability

In this section, we show under which conditions the canonical PL-approximability exists and how it can be efficiently computed. We do so by introducing a geometric object, the *twisted ribbon*, which captures the entire family of valid power-law bounds for S. We then study the structural properties of this ribbon to identify a *pinch point*, which yields the canonical PL-approximability (Definition 5). Finally, we describe a two-phase algorithm that efficiently locates a pinch point based on the twisted ribbon's geometry.

We first observe that, for any given α, the tightest possible power-law bound on the PL-approximability of S is given by choosing β_{low} and β_{high} as $L(\alpha)$ and $H(\alpha)$, respectively. The proof follows easily from the definitions of L and H.

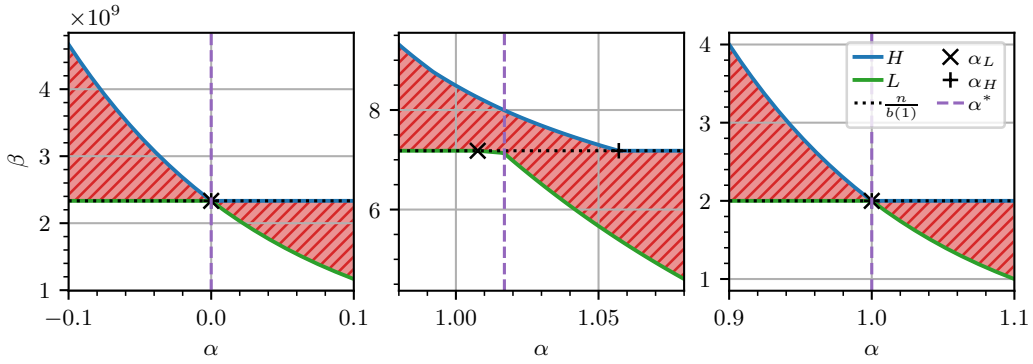

Figure 3 On the left, a twisted ribbon corresponding to a set S giving the best case $b(\varepsilon) = 1$ for any $\varepsilon \in \{1, \ldots, 1024\}$. On the center, a twisted ribbon derived from the human genome (T2TCHM13v2.0). On the right, a twisted ribbon corresponding to a set S giving the worst case $b(\varepsilon) = N/(2\varepsilon)$ for any ε. For symmetry with the other plots, the left plot extends to values of $\alpha < 0$ despite the ribbon being formally defined for $\alpha \geq 0$.

▶ **Lemma 6.** *For any $\alpha \geq 0$ and $\delta > 0$, we have that (i) S is power-law bounded over $\mathcal{E}$ with parameters $(\alpha, L(\alpha), H(\alpha))$, and (ii) S is not power-law bounded over $\mathcal{E}$ with parameters $(\alpha, L(\alpha) + \delta, H(\alpha) - \delta)$.*

We can plot the functions H and L on a Cartesian plane with α along the horizontal axis and β along the vertical axis. The region enclosed between these two curves forms the aforementioned twisted ribbon, depicted in Figure 3 and formally defined as follows.

▶ **Definition 7** (Twisted ribbon)**.** *The twisted ribbon of S is the region in the α-β space enclosed between the curves $L(\alpha)$ and $H(\alpha)$ for any $\alpha > 0$. Formally, the twisted ribbon of S is the set $\mathcal{R}_S = \{(\alpha, \beta) \in \mathbb{R}^2_{\geq 0} \mid L(\alpha) < \beta < H(\alpha), \alpha \geq 0\}$.*

Observe that choosing β_{low} or β_{high} inside of the twisted ribbon (that is, setting $\beta_{\text{low}} > L(\alpha)$ or $\beta_{\text{high}} < H(\alpha)$ for some α) would violate Equation 1, and therefore would not yield a valid bound on the PL-approximability of S. This reduces the space of relevant power-law bounds to that of $\{(\alpha, L(\alpha), H(\alpha))\}_{\alpha \geq 0}$.

We further aim to summarize this entire family of bounds into a single representative instance. Intuitively, this means identifying a value of α that yields the tightest bound. This corresponds to a value α^* that minimizes the twisted ribbon width $W(\alpha) = H(\alpha) - L(\alpha)$, i.e., what we call a pinch point (Definition 5).

To study the existence and computation of a pinch point, we first show some structural properties of the twisted ribbon. For technical convenience, we assume that the (nonincreasing integer) function $b(\varepsilon)$ has been extended to a continuous, nonincreasing, and differentiable function on the reals, e.g., via monotone cubic interpolation.

▶ **Lemma 8.** *If $1 \in \mathcal{E}$ and $|\mathcal{E}| > 1$, then:*

- **(i)** *H and L are continuous functions.*
- **(ii)** *For any $\alpha \geq 0$, it holds $0 \leq L(\alpha) \leq \frac{n}{b(1)} \leq H(\alpha) \leq \frac{n}{b(\varepsilon_{\max})}$, where $\varepsilon_{\max} = \max \mathcal{E}$.*
- **(iii)** *There exists a value α_L such that $L(\alpha) = \frac{n}{b(1)}$ for $\alpha \leq \alpha_L$, and $L(\alpha_L + \delta)$ decreases as $\delta > 0$ increases. We call α_L a flattening point of S.*
- **(iv)** *There exists a value α_H such that $H(\alpha) = \frac{n}{b(1)}$ for $\alpha \geq \alpha_H$, and $H(\alpha_H - \delta)$ increases as $\delta > 0$ increases. We also call α_H a flattening point of S.*
- **(v)** *The flattening points satisfy $\alpha_L \leq \alpha_H$.*

Note that the lemma holds for any dataset, as its conditions are related to a parameter ($\mathcal{E}$) chosen by the user. The following result shows that a pinch point exists and is confined between (or at) the flattening points mentioned in Lemma 8 and depicted in Figure 3.

▶ **Theorem 9.** *If $1 \in \mathcal{E}$ and $|\mathcal{E}| > 1$, then a pinch point α^* of S exists and is guaranteed to lie in the interval $[\alpha_L, \alpha_H]$, where α_L and α_H are the flattening points of S.*

We now focus on finding a pinch point, that is, on solving the problem $\operatorname{argmin}_{\alpha \geq 0} W(\alpha)$. Observe W is non-differentiable due to the presence of min and max operations. Moreover, we assume that W is *unimodal*, namely, it has a unique minimum α^*, is strictly decreasing for all $\alpha < \alpha^*$, and strictly increasing for $\alpha > \alpha^*$. While we do not have any general theoretical guarantee on the unimodality of W (except in simple cases such as those shown in the left and right plots of Figure 3), our empirical observation suggests that unimodality often holds in practice (see Section 5.5). If this is not the case, our algorithm finds a local minimum instead. We also assume $b(\varepsilon)$ has been precomputed for all $\varepsilon \in \mathcal{E}$, allowing $W(\alpha)$ to be evaluated in $\mathcal{O}(|\mathcal{E}|)$ time via a direct computation based on Equation 2. This precomputation can be done in $\mathcal{O}(N|\mathcal{E}|)$ time [27].

The algorithm works in two phases. In the first phase, we identify the flattening points α_L and α_H by observing from Lemma 8 that α_L is the smallest value of α such that $\frac{n}{\varepsilon^\alpha b(\varepsilon)} = \frac{n}{b(1)}$ for some $\varepsilon \in \mathcal{E} \setminus \{1\}$. This equality simplifies to $\alpha = \log_\varepsilon \frac{b(1)}{b(\varepsilon)}$, so we obtain $\alpha_L = \min_{\varepsilon \in \mathcal{E} \setminus \{1\}} \log_\varepsilon \frac{b(1)}{b(\varepsilon)}$, which can be computed in $\mathcal{O}(|\mathcal{E}|)$ time. A similar derivation with max instead of min yields α_H.

In the second phase, we run a derivative-free minimization algorithm (golden-section search) within the interval $[\alpha_L, \alpha_H]$, which is guaranteed to contain α^* due to Theorem 9. It converges in $\lceil \log_\phi \frac{\alpha_H - \alpha_L}{\tau} \rceil$ iterations, where τ is the desired tolerance, and $\phi \approx 1.618$ is the golden ratio [29]. Since the cost of each iteration is dominated by the evaluation of $W(\alpha)$, the overall time complexity of finding the pinch point is $\mathcal{O}(|\mathcal{E}| \log \frac{\alpha_H - \alpha_L}{\tau})$. This gives the following theorem.

▶ **Theorem 10.** *Suppose the ribbon width $W(\alpha) = H(\alpha) - L(\alpha)$ is unimodal, $1 \in \mathcal{E}$, and $|\mathcal{E}| > 1$. Suppose the value $b(\varepsilon)$ has been precomputed for all $\varepsilon \in \mathcal{E}$ in $\mathcal{O}(N|\mathcal{E}|)$ time. Then, there exists an algorithm that finds the pinch point α^* in $\mathcal{O}(|\mathcal{E}| \log \frac{\alpha_H - \alpha_L}{\tau})$ time, where τ is the desired tolerance.*

Section 5.1 will show that the runtime of this algorithm is on the order of a few milliseconds in practice.

4 Using PL-approximability bounds in downstream applications

There are a number of PLA-based learned data structures whose space-time performance is bounded in terms of the PLA-size, such as the PGM-index [11, Theorem 1], LeMonHash [8, Theorem 2], and the PLA-index [2, Theorem 1]. In this section, we showcase how our PL-approximability framework can be used to reinterpret such bounds, focusing on the PLA-index as a concrete example particularly relevant in genomic applications. The PLA-index has been applied to substantially improve the run time of suffix array search and decrease the space of a read aligner and direct-access lookup tables [2].

The PLA-index uses a piecewise linear ε-approximation to a multiset of k-mers S to estimate the rank of a query k-mer in S [2]. The tuning parameter ε represents a time/space trade-off, with smaller ε values resulting in faster index query times but more segments.

Theorem 1 in [2] describes the size of the PLA-index as a function of the number of segments used by the PLA index (denoted by s_ε), N, and ε:

$$s_\varepsilon(2k - \lg \frac{s_\varepsilon^{31/16}}{N(1+4\varepsilon)} + \mathcal{O}(1)). \tag{3}$$

As s_ε can be high in the worst case, Equation 3 does not capture the fact that the space is very low in practice.

Here, we show that if S has a power-law bounded PL-approximability, then Equation 3 can be turned into a useful predictor of PLA-index space usage. One of the technical difficulties is that PL-approximability gives bounds on the PLA size $b(\varepsilon)$, whereas Equation 3 is stated in terms of s_ε. Though the PLA-index uses a segmentation based on the optimal one, it requires consecutive segments to share an endpoint and in some rare cases breaks segments. This dependence of s_ε on $b(\varepsilon)$ is difficult to quantify in terms of S. Instead, we isolate the dependence by expressing our results in terms of $\gamma_\varepsilon \triangleq \frac{s_\varepsilon}{b(\varepsilon)}$. Though γ_ε is theoretically unknown (except that $\gamma_\varepsilon \geq 1$), empirical results show that it is close to one on genome spectra. For example, for the 21-spectrum of chromosome 1 of the human genome, γ_ε values are below 1.21 for all tested ε up to $\varepsilon = 1023$.

▶ **Theorem 11.** *Let S be a multiset of k-mers, with n distinct elements, N total elements. Suppose the PL-approximability of S is $(\alpha, \beta_{\text{low}}, \beta_{\text{high}})$ power-law bounded over $\mathcal{E}$. Let B_ε be the number of bits used by the PLA-index on S with tuning parameter ε. Then, for all $\varepsilon \in \mathcal{E}$, we have that*

$$f(\beta_{\text{high}}, \beta_{\text{low}}) \leq B_\varepsilon \leq f(\beta_{\text{low}}, \beta_{\text{high}}),$$

where

$$f(\beta_1, \beta_2) = \frac{n\gamma_\varepsilon}{\beta_1 \varepsilon^\alpha}\left(2k - \frac{15 \lg n}{16} + \left(\frac{31 \log_\varepsilon(\beta_2 \varepsilon^\alpha)}{16} + 1\right)\lg \varepsilon + C\right), \tag{4}$$

and $C = c - \frac{31 \lg \gamma_\varepsilon}{16} + \lg N/n$ and, for large enough $\frac{n}{\beta_{\text{high}}\varepsilon^\alpha}$, c is a number between 5 and 7.

We have isolated and combined some terms into a variable C, in order to convey that they can be thought of as constants in practice. First, N/n is in practice smaller than 2 when avoiding small values of k, for which a suffix array would not make sense as an application anyway. Second, as we mentioned earlier, γ_ε is in practice a number close to 1.

Interpreting Eq. 4 is helped by the observation that given the type of values we see in practice, the dominating term is the leading one, i.e. $1/(\beta_1 \varepsilon^\alpha)$. This intuition is confirmed by Figure 4AB, which plots Eq. 4 using real numbers from human chromosome 1. As suggested by the leading term, increasing the α value of a dataset by δ results in the leading term decreasing by a factor of ε^δ, with the plots confirming this type of exponential decay for Eq. 4 overall. Note that for $\varepsilon = 63$ (Panel A), the decay is slower than for $\varepsilon = 1023$ (Panel B), as expected from the leading term. Figure 4CD shows how the gap between β_{low} and β_{high} affects the gap between the upper and lower bounds of Theorem 11 and gives the scale of the gap at which our bounds become meaningless.

5 Experimental results

The goal of our experiments is threefold. First, we want to understand how much PL-approximability varies among the tree of life. Second, we want to understand the properties of genome spectra that make them different from random k-mer multisets. Finally, we want to evaluate the suitability of CaPLa as a measure of PL-approximability. Our tool for computing CaPLa, as well as reproducibility information, is available through GitHub at `https://github.com/medvedevgroup/CaPLa`.

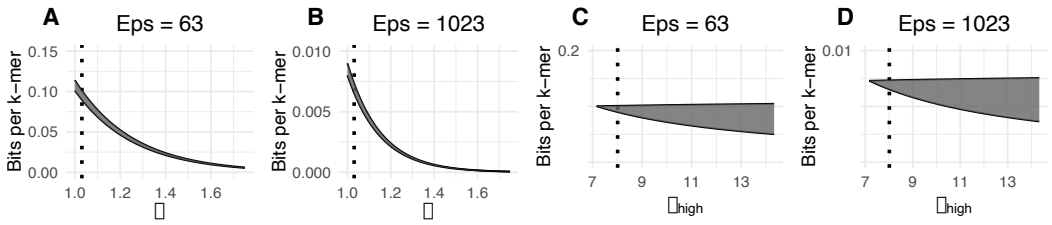

Figure 4 Analysis of space usage of PLA index, as predicted by Theorem 11, plugging in $C = 6$. In each panel, the shaded area is bounded from above by the upper bound curve and below by the lower bound curve. We use values corresponding to human chromosome 1 with $k = 21$: $n = 195,735,278$, $\alpha^* = 1.03$, $\beta^*_{\text{low}} = 7.2$, and $\beta^*_{\text{high}} = 8.0$, $\gamma_{63} = 1.13$ and $\gamma_{1023} = 1.15$. In panels A and B, we fix $\beta_{\text{low}} = \beta^*_{\text{low}}$ and $\beta_{\text{high}} = \beta^*_{\text{high}}$ while varying α. The vertical dotted line corresponds to $\alpha = \alpha^*$. In panels C and D, we fix the values of $\alpha = \alpha^*$ and $\beta_{\text{low}} = \beta^*_{\text{low}}$ and vary β_{high}. The vertical dotted line corresponds to $\beta_{\text{high}} = \beta^*_{\text{high}}$.

5.1 Experimental setup

Data. We downloaded a sample of RefSeq genomes that are complete and full, do not have any missing bases, and are longer than 10,000nt. We further filtered out any genomes for which the number of segments at $\varepsilon = 1024$ was 1 (i.e. $b(1024) = 1$). This excludes genomes that are too trivial to offer meaningful insight and for which the PL-approximability becomes trivially dominated by n. We process the genomes to make everything uppercase, remove any letters that are not A, C, G, or T, and concatenate all contigs into a single string. The resulting dataset contains 513 genomes, representing the kingdoms of virus (91 genomes), bacteria (200 genomes), archaea (200 genomes), and fungi (22 genomes). The median genome length was 2.98 mil and the maximum was 63 mil . We also downloaded six other larger genomes: *C. elegans*, *P. yoelii*, *M. commoda*, *O. lucimarinus*, and the latest high-quality assemblies of the human (T2TCHM13v2.0) and Gorilla genomes .

Tool. Our tool to compute CaPLa takes a genome, a k-mer size, and $\mathcal{E}$, builds a suffix array of the genome using libdivsufsort [24], computes the values $\{b(\varepsilon)\}_{\varepsilon\in\mathcal{E}}$ using O'Rourke's algorithm [27] on the genome's k-mers (extracted in sorted order using the suffix array), and then computes CaPLa using the algorithm of Theorem 10 with tolerance $\tau = 1.49 \times 10^{-8}$ (i.e., the square root of the machine epsilon). We use $\mathcal{E}$ as the set of all integers from 1 to 1024, and $k = 21$ by default. We also construct the PLA-index [2] for $\varepsilon \in \{32, 64\}$ for all genomes, using $k = 21$ as in previous works [2, 17]. Full details, including the accession number of the genomes, their CaPLa values, PLA-index sizes, and other associated information are available at our GitHub repository.

Runtime. For runtime analysis, we used a machine with a 2.10 GHz Intel Xeon CPU E5-2683 v4 processor with 512 GB of memory. To give a sense of the execution time of our pipeline, for a RefSeq genome of median length (GCF_001870125, 2.98M bases), it takes 0.318 s to construct the suffix array, 730.979 s to compute the values $\{b(\varepsilon)\}_{\varepsilon\in\mathcal{E}}$, and 0.006 seconds to run the algorithm of Theorem 10. Thus, the runtime of the algorithm of Theorem 10 is negligible and we do not investigate it further. A more detailed analysis of the runtime of computing $b(\varepsilon)$ values is covered in other papers (e.g. [2]); since it is not a contribution of our work, we do not expand on those studies here.

CaPLa comparison. At first glance, comparing CaPLa values across genomes may seem challenging, as α^* has a larger impact on the PL-approximability than β^*_{low} or β^*_{high}, due to its presence in the exponent of ε in Equation 1. However, the values are easier to compare when the user has a desired query time, i.e. when we think of ε as being fixed. In this case, we define the *ε-mapped CaPLa* as the following two values: $\rho_{\text{low}} = \alpha + \log_\varepsilon \beta_{\text{low}}$ and $\rho_{\text{high}} = \alpha + \log_\varepsilon \beta_{\text{high}}$. The PL-approximability power-law bound (Equation 1) can then be rewritten in terms of ρ_{low} and ρ_{high}, i.e. $\varepsilon^{\rho_{\text{low}}} \leq n/b(\varepsilon) \leq \varepsilon^{\rho_{\text{high}}}$. This replaces the three variables of CaPLa with two variables which, moreover, are now directly comparable between genomes. We can further average ρ_{low} and ρ_{high} to get the *averaged ε-mapped CaPLa*, which becomes useful for looking at CaPLa distributions across large datasets.

Table 1 CaPLa of the RefSeq dataset and of six additional genomes. The first row corresponds to all of the RefSeq dataset, while the later four rows separate the dataset into Kingdoms. Note that percentiles are not reported for the last six rows because each row corresponds to a single genome.

	α^*			β^*_{low}			β^*_{high}		
Category	5th%	Mean	95th%	5th%	Mean	95th%	5th%	Mean	95th%
RefSeq dataset	1.03	1.09	1.16	4.8	7.4	8.6	7.1	9.0	11.1
Virus	1.06	1.12	1.19	3.4	5.6	7.9	7.1	8.6	10.8
Bacteria	1.03	1.08	1.12	6.4	7.8	8.6	7.0	9.1	11.2
Archaea	1.03	1.07	1.12	6.7	7.8	8.6	7.2	8.9	10.7
Fungi	1.08	1.12	1.17	6.5	8.2	9.0	7.3	10.3	12.5
C. elegans		1.06			7.4			8.2	
P. yoelii		1.00			5.5			6.1	
M. commoda		1.06			7.4			8.2	
O. lucimarinus		1.06			7.4			8.1	
Human		1.02			7.1			8.0	
Gorilla		1.00			7.1			7.9	

5.2 Understanding the scale of CaPLa

Table 1 shows the CaPLa values for our data. But, to make sense of the variability, we need to first understand how variations in CaPLa values translate into variations in the space of downstream data structures. The theoretical predictions of Theorem 11 allow such an analysis for the case of the PLA-index. To simplify the analysis, we consider the case that PL-approximability follows a perfect power law, i.e. there is a β^* such that $\beta^* = \beta^*_{\text{low}} = \beta^*_{\text{high}}$. As Theorem 11 and Figure 4 indicated, the total bits per k-mer is approximately $\frac{\gamma_\varepsilon}{\beta^* \varepsilon^{\alpha^*}}$. When α^* increases by an additive factor of δ, then the approximate space decreases by a multiplicative factor of $1 - \varepsilon^{-\delta}$; when β^* increases by a multiplicative factor of δ, then the approximate space decreases by a multiplicative factor of $\frac{\delta}{\delta+1}$. Table 2 shows these numbers for some concrete values that we encounter in our data. For example, in the bacterial genomes, the α^* values increase by 0.1 from the 5th and 95th percentiles (Table 1); Table 2 shows that this corresponds to a decrease in space of 32% for the PLA index with $\varepsilon = 64$. Thus the variability in CaPLa across RefSeq corresponds to a substantial variability in predicted bits per k-mer of the associated PLA-index.

5.3 Variation of CaPLa among the tree of life

To better understand the distributions of CaPLa among different kingdoms, Figure 5AC shows the distributions of the averaged ε-mapped CaPLa. The fungi kingdom has a noticeably higher CaPLa. The difference is substantial, as is reflected in the lower PLA-index space

Table 2 The effect of CaPLa on PLA-index space. We show the decrease in bits per distinct k-mer when adding δ to α (left columns) and multiplying β by $(1+\delta)$ (right columns). The numbers without parenthesis indicate the predicted decrease using $\frac{\gamma_\varepsilon}{\beta^* \varepsilon \alpha^*}$, the leading term of Equation 4. The numbers in parenthesis indicate the change predicted by the full form of Theorem 11, using $C = 6$ and the real values from human chromosome 1 with $k = 21$ ($n = 195,735,278$, $\alpha^* = 1.0309$, $\beta^*_{\text{low}} = 7.1645$, and $\beta^*_{\text{high}} = 8.0119$, $\gamma_{63} = 1.13$ and $\gamma_{32} = 1.13$).

	$\alpha \leftarrow \alpha + \delta$			$\beta \leftarrow \beta(1+\delta)$	
	$\delta = 0.01$	$\delta = 0.05$	$\delta = 0.1$	$\delta = 0.1$	$\delta = 0.5$
$\varepsilon = 32$	3% (3%)	16% (15%)	29% (28%)	9% (9%)	33% (32%)
$\varepsilon = 64$	4% (4%)	19% (18%)	34% (32%)	9% (9%)	33% (32%)
$\varepsilon = 1024$	7% (6%)	29% (28%)	50% (48%)	9% (9%)	33% (32%)

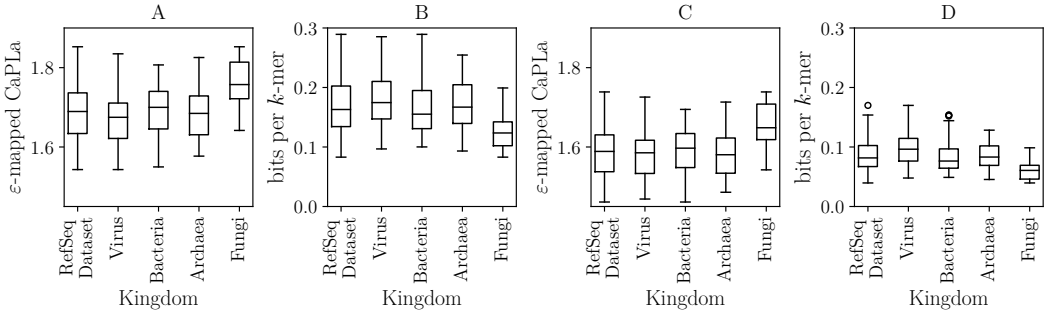

Figure 5 Distribution of averaged ε-mapped CaPLa across RefSeq. Panels A and C show the values for $\varepsilon = 32$ and $\varepsilon = 64$, respectively. Panels B and D show the distribution of bits per distinct k-mer in the constructed PLA indexes. In each panel, the left box plot is for our whole RefSeq dataset, while the remaining four box plots correspond to subsets according to Kingdoms.

usage of fungi genomes compared to the other kingdoms (Figure 5BD). The reasons for this could be due to biological factors but could also be due to biases in the type of genomes that have been sequenced and/or deposited into RefSeq.

Figure 6 shows the space usage of the PLA index and breaks down the relationship between bits per distinct k-mer and the CaPLa separately for each genome. The variation is substantial, with space differences as big as threefold between different RefSeq genomes. Furthermore, the figure shows a strong correlation between CaPLa and bits per distinct k-mer of the PLA-index.

We also split the human genome into its 24 constituent chromosome sequences. Figure 7 shows that CaPLa varies substantially across the chromosomes, indicating that PL-approximability varies not only between different species but also within the genome of a single species. Figure 7 also shows a similar pattern for the chromosomes of the Gorilla genome. Chromosome Y, which is known for being highly repetitive, is an outlier in both species, with substantially lower CaPLa than other chromosomes.

In spite of the CaPLa variability, our results give a general range of CaPLa for genomes that are not part of our dataset. This gives a user with a new genome sequence a robust estimate of the expected efficiency of a PLA-based learned index. This is in contrast with datasets from other domains that have been shown to roughly fit a power-law curve with significantly different α values, such as $\alpha = 1.8$ on book popularity data, $\alpha = 1.4$ on Facebook user IDs, and $\alpha = 2$ on random gaps [9].

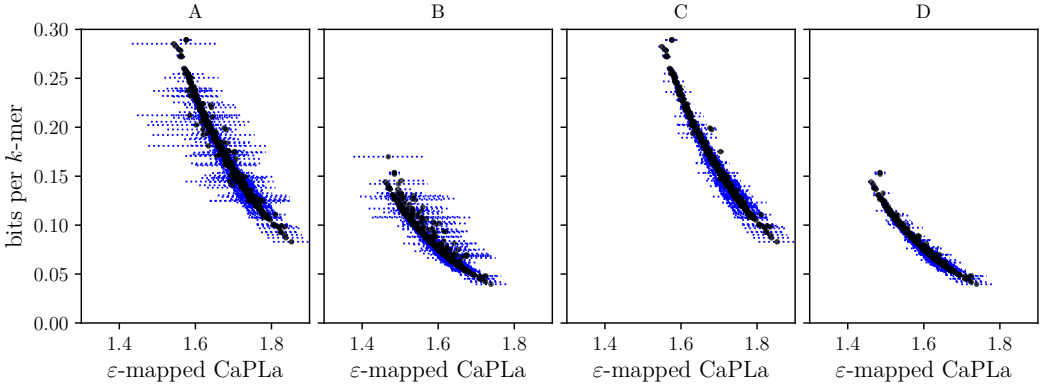

Figure 6 Relation between the bits per distinct k-mer of the PLA-index to the ε-mapped CaPLa for the RefSeq dataset. Panel A shows the result for $\varepsilon = 32$ and Panel B for $\varepsilon = 64$. Each dashed horizontal line represents the range of the ε-mapped CaPLa for a single genome and each black dot represents an averaged ε-mapped CaPLa for a single genome. Panels C and D show the same information as panels A and B, but we additionally filter out all genomes that are shorter than $80,000$nt. For such shorter genomes, the constant overhead of the PLA-index data structures starts to dominate the part that makes use of the PL-approximation; as the plot shows, most of the widest horizontal lines in panels A and B are gone after filtering out the shorter genomes.

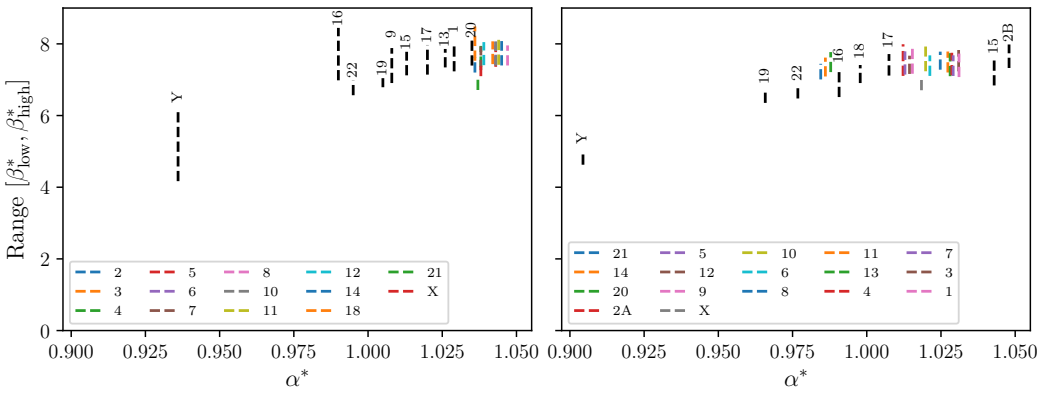

Figure 7 CaPLa of Human (left) and Gorilla (right) chromosomes. Each vertical segment is labeled with the corresponding chromosome, either in the legend or directly above the segment.

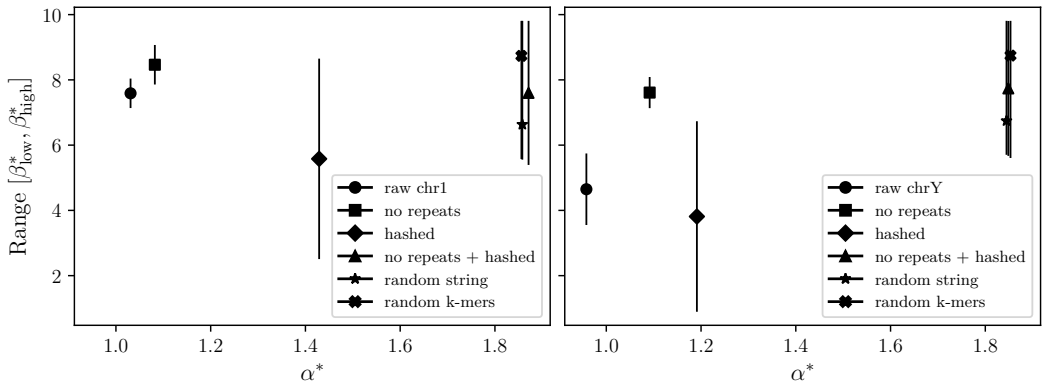

Figure 8 PL-approximability of a genome spectrum versus a random k-mer multisets. The left and right panels show human chromosomes 1 and Y, respectively, for $k = 21$. The y-axis shows the range from β^*_{low} to β^*_{high}. The random string and the random k-mers are generated so as to match the number of k-mers in the raw sequence.

5.4 What makes a genome spectrum special?

Figure 8 shows a huge difference between the CaPLa of human chromosome 1 ($\alpha^* = 1.03$) and a random multiset of k-mers of the same size ($\alpha^* = 1.86$). Theorem 11 predicts a 26-fold difference in the size of the corresponding PLA indexes, at $\varepsilon = 64$. We therefore wanted to tease out the effect of several factors which distinguish a genome spectrum from a random multiset. The first factor is that the k-mers of a genome spectrum must be overlapping, i.e. not all multisets of k-mers can be put together into a continuous genome sequence, even allowing for chromosome breaks. To test the effect of this, we create a random string with the same length as chr1 and find that it also had $\alpha^* = 1.86$. Thus, this factor does not seem to play a role, at least in isolation.

A second effect is that chr1 is not random but is evolved to have a biological function. One consequence is that the genome has k-mers which appear more than once, which is rare in a random string of the same length (for $k = 21$). To test the effect of this, we make a linear scan of chr1 and clip out any encountered k-mer which we have previously seen. In this way, we create a genome-like string which is nearly repeat-free. We found $\alpha^* = 1.08$ for this low-repeat string, suggesting that repeats do negatively affect the PL-approximability.

Another consequence of having a biological function is that the spectrum of a genome has a non-uniform distribution. To test the effect of this, we process the k-mers of chr1 with a hash function and use the hashed value as the basis for the rank function. This has the effect of making the distribution uniform. The resulting $\alpha^* = 1.43$ shows that the non-uniform distribution of the k-mers in a spectrum substantially decreases the PL-approximability.

Finally, we combined the two effects by removing repeats and then hashing the k-mers, getting an $\alpha^* = 1.87$, the same as a random multiset. This indicates that the two factors have an interplay which drives down the PL-approximability a lot when combined.

We repeated the same experiments on the Y chromosome (Figure 8). The effects were similar, though the effect of removing repeats was more pronounced (α^* rose from 0.96 to 1.09). This was expected, as the Y is known to be more repetitive than chromosome 1 [30].

5.5 Stability of CaPLa as a measure

Uniqueness of pinch point. We conducted brute-force tests to investigate whether in our data the pinch point is unique (i.e. CaPLa is well defined), and to verify whether the unimodality assumption required by our algorithm in Theorem 10 holds in practice. Specifically, for each genome in the RefSeq dataset, we sampled the twisted ribbon width $W(\alpha)$ at uniformly spaced α-values within the interval $[\alpha_L, \alpha_H]$ containing a pinch point (Theorem 9), using a step size of 10^{-6}. We then checked for the presence of multiple minima within a numerical tolerance of $\tau = 1.49 \times 10^{-8}$, and verified unimodality by looking for a decreasing and increasing trend before and after the minimum, respectively.

We found that the pinch point was unique in all of the 513 RefSeq genomes. Furthermore, the unimodality assumption was not met in only 11 datasets. Even in those 11 cases, the algorithm of Theorem 10 avoided local minima and matched the brute-force pinch point in all but one instance, where the actual pinch point was nonetheless close (within 3.23×10^{-2}). In summary, up to the tested step size, CaPLa was defined for all genomes and our efficient algorithm returned the correct value in all but one case.

Genome length. A desirable property of a measure of PL-approximability is that it is not causally dependent on genome length and is only an intrinsic property of a genome's rank curve. To verify this, we compare the lengths of our RefSeq genomes with their α^*, β^*_{low}, and β^*_{high} values, respectively, as well as to the ε-mapped CaPLa, for $\varepsilon \in \{2, 4, 8, \ldots, 1024\}$. The Pearson correlation coefficient (R^2) is less than 0.02 for all comparisons, providing evidence that the relationship between CaPLa and genome length is negligible.

***k*-mer size.** The structure of the rank curve is unavoidably dependent on k, because for small k, the number of repeats and the density of S with respect to the universe gets high. Ideally, when k is above the threshold where spurious repeats are rare, CaPLa should stabilize. To validate this, we looked at the behavior of CaPLa with respect to k for the RefSeq dataset and for bigger mammalian genomes. Figure 9 shows six genomes, which we manually selected to capture the phylogenetic diversity of the data. We measure the averaged ε-mapped CaPLa for $\varepsilon = 64$. The plots show that indeed CaPLa plateaus at some point between $k = 16$ and $k = 21$, depending on the genome. The difference between the averaged ε-mapped CaPLa values at $k = 26$ and $k = 31$ was less than 0.01 for all six genomes, both for $\varepsilon = 64$ (Figure 9) and $\varepsilon = 32$ (data not shown). Similarly, between $k = 21$ and $k = 26$, the differences in α^* values were less than 0.4% and the difference in the β^*_{low} and β^*_{high} values were less than 4%. These results provide strong evidence that the CaPLa is relatively stable for large enough values of k, and, in particular, for values of $k \geq 21$.

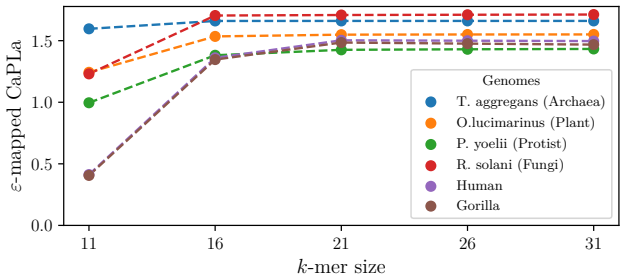

Figure 9 Effect of k-mer size on CaPLa, for $\varepsilon = 64$, using averaged ε-mapped CaPLa

Accuracy of prediction. We compare the space range predicted by plugging in CaPLa into Theorem 11 with the true space taken by the PLA-index (Figure 10). In the majority of cases, the true space falls within the predicted range, as expected. However, for most cases of $\varepsilon = 1024$ and one case of $\varepsilon = 16$, the true space is higher than the predicted range. This is likely due to one of two reasons: 1) the PLA-index uses an implementation of Elias-Fano encoding that has a constant overhead per character, which is not the theoretically optimal data structure used for theoretical analysis, and 2) the results of Theorem 11 hold only asymptotically and do not account for constant factors that contribute to the space, especially for higher ε where the total space is small.

We also compare the true space with the space predicted by fitting a single power-law model to the tabulated PLA-sizes (as we previously discussed in Figure 2) and plugging the fitted parameters into Theorem 1 from [2], with $c = 4$. As Figure 10 shows, the fit-based predictions are often significantly under-predicting the true space, especially at $\varepsilon = 16$.

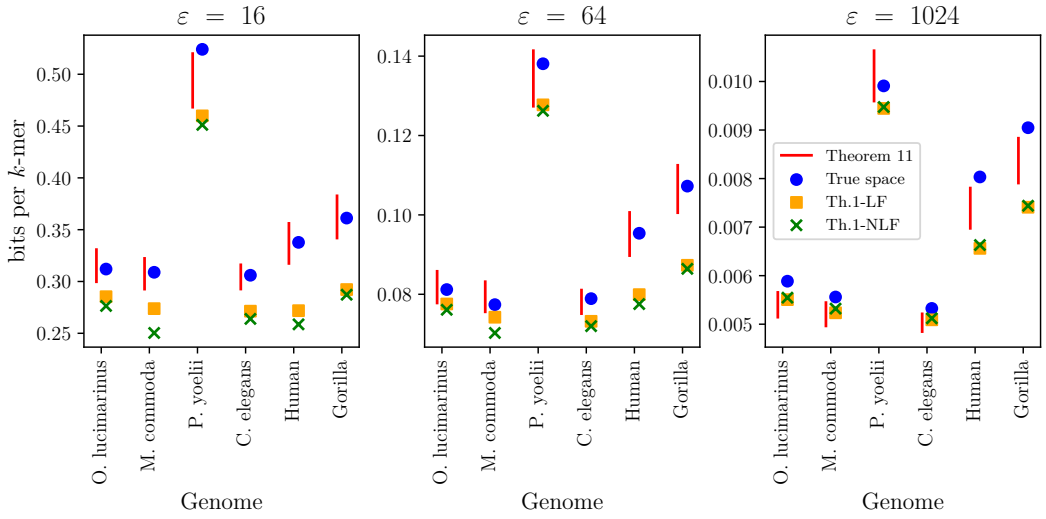

Figure 10 The true space of the PLA-index compared with various forms of predictions, for $\varepsilon \in \{16, 64, 1024\}$. The true space is measured using a version of the PLA-index implementation which encodes the X-array using Elias-Fano, which is the version that is theoretically analyzed in [2] and in Theorem 11. The predicted range (red) corresponds to plugging in the CaPLa into Theorem 11, using $c = 7$. The Th.1-LF and Th.1-NLF correspond to fitting a power-law model with a linear and non-linear fit, respectively, and plugging in the fitted parameters into Theorem 1 in [2] (with $c = 4$). The fitted and non-fitted models are described in the caption of Figure 2.

6 Conclusion

In this paper, we introduced the notion of PL-approximability of a multiset and a corresponding measure called CaPLa (Section 2). Our theoretical results show that CaPLa is well-defined and can be computed efficiently, under reasonable assumptions which we demonstrate hold in almost all tested data (Section 3). Our experimental results show that CaPLa varies substantially not only across genomes and kingdoms, but also within individual genomes, and this variability can lead to differences of up to 50% in the space required by a PLA-based data structure (Sections 5.2 and 5.3). Within our CaPLa framework, the key factors distinguishing a genome spectrum from a random k-mer multiset are the presence of

repeats and the non-uniform distribution of k-mers. The combined effect of these two factors substantially reduces PL-approximability (Section 5.4). CaPLa has several desirable properties, namely, it is well-defined and efficient to compute across our experimented genomes, it is not trivially correlated with genome length thus indicating it captures the intrinsic properties of the genome, and it is robust with respect to k-mer size when $k \geq 21$ (Section 5.5).

Our result has a concrete application to parameter selection in PLA-based indexes such as the PLA-index. When a target space is specified, an approximate estimate of the dataset's CaPLa can be used with Theorem 11 to compute a suitable value for the tuning parameter ε. This is particularly advantageous when the CaPLa is known for a similar dataset (e.g., from another individual of the same species), as it avoids the need to recompute the full set of $(\varepsilon, b(\varepsilon))$ pairs. In the absence of prior CaPLa information, future work could also explore efficient estimation of CaPLa over a small sample of the dataset itself (e.g., a contiguous region of the suffix array).

Our paper presents several future directions. From an application perspective, it will be interesting to connect the variability of PL-approximability to biological properties. Our preliminary results show that PL-approximability is higher in fungi and lower in repeat-rich chromosomes like the Y, but this raises more questions than it answers. From a theory perspective, it would be interesting to find the conditions under which the pinch point is unique. In our experiments, we could not find any instances of non-unique pinch points, suggesting that a theoretical proof may be possible. Finally, Theorem 9 requires that $1 \in \mathcal{E}$ in order to guarantee the existence of a pinch point. This condition is necessary, as in fact one can prove that a pinch point does not exist if $1 \notin \mathcal{E}$. This raises an interesting open question if there is some alternative notion of CaPLa that can be used when $1 \notin \mathcal{E}$.

References

1 Md. Hasin Abrar and Paul Medvedev. PLA-complexity of k-mer multisets. *bioRxiv*, 2024. doi:10.1101/2024.02.08.579510.

2 Md. Hasin Abrar and Paul Medvedev. PLA-index: A k-mer Index Exploiting Rank Curve Linearity. In *Proc. 24th International Workshop on Algorithms in Bioinformatics (WABI)*, volume 312 of *LIPIcs*, pages 13:1–13:18, 2024. doi:10.4230/LIPIcs.WABI.2024.13.

3 Md. Hasin Abrar, Paul Medvedev, and Giorgio Vinciguerra. Efficiency of learned indexes on genome spectra, 2025. doi:10.1101/2025.07.10.664199.

4 Domenico Amato, Giosuè Lo Bosco, and Raffaele Giancarlo. Neural networks as building blocks for the design of efficient learned indexes. *Neural Computing and Applications*, 35(29):21399–21414, 2023. doi:10.1007/S00521-023-08841-1.

5 Naiyong Ao, Fan Zhang, Di Wu, Douglas S. Stones, Gang Wang, Xiaoguang Liu, Jing Liu, and Sheng Lin. Efficient parallel lists intersection and index compression algorithms using graphics processing units. *PVLDB*, 4(8):470–481, 2011. doi:10.14778/2002974.2002975.

6 Rayan Chikhi, Jan Holub, and Paul Medvedev. Data structures to represent a set of k-long DNA sequences. *ACM Computing Surveys*, 54(1):1–22, 2021. doi:10.1145/3445967.

7 Rayan Chikhi, Brice Raffestin, Anton Korobeynikov, Robert Edgar, and Artem Babaian. Logan: Planetary-scale genome assembly surveys life's diversity. *bioRxiv*, July 2024. doi:10.1101/2024.07.30.605881.

8 Paolo Ferragina, Hans-Peter Lehmann, Peter Sanders, and Giorgio Vinciguerra. Learned monotone minimal perfect hashing. In *Proc. 31st Annual European Symposium on Algorithms (ESA)*, volume 274 of *LIPIcs*, pages 46:1–46:17, 2023. doi:10.4230/LIPICS.ESA.2023.46.

9 Paolo Ferragina, Fabrizio Lillo, and Giorgio Vinciguerra. On the performance of learned data structures. *Theoretical Computer Science*, 871:107–120, 2021. doi:10.1016/j.tcs.2021.04.015.

10 Paolo Ferragina and Giorgio Vinciguerra. Learned data structures. In Luca Oneto, Nicolò Navarin, Alessandro Sperduti, and Davide Anguita, editors, *Recent Trends in Learning From Data*, pages 5–41. Springer International Publishing, 2020. doi:10.1007/978-3-030-43883-8_2.

11 Paolo Ferragina and Giorgio Vinciguerra. The PGM-index: a fully-dynamic compressed learned index with provable worst-case bounds. *PVLDB*, 13(8):1162–1175, 2020. doi:10.14778/3389133.3389135.

12 Alex Galakatos, Michael Markovitch, Carsten Binnig, Rodrigo Fonseca, and Tim Kraska. FITing-Tree: A data-aware index structure. In *Proc. 45th International Conference on Management of Data (SIGMOD)*. ACM, 2019. doi:10.1145/3299869.3319860.

13 Darryl Ho, Saurabh Kalikar, Sanchit Misra, Jialin Ding, Vasimuddin Md, Nesime Tatbul, Heng Li, and Tim Kraska. LISA: Learned indexes for sequence analysis. *bioRxiv*, 2021. doi:10.1101/2020.12.22.423964.

14 Youngmok Jung and Dongsu Han. BWA-MEME: BWA-MEM emulated with a machine learning approach. *Bioinformatics*, 38(9):2404–2413, 2022. doi:10.1093/bioinformatics/btac137.

15 Saurabh Kalikar, Chirag Jain, Md Vasimuddin, and Sanchit Misra. Accelerating minimap2 for long-read sequencing applications on modern CPUs. *Nature Computational Science*, 2(2):78–83, 2022. doi:10.1038/s43588-022-00201-8.

16 Andreas Kipf, Ryan Marcus, Alexander van Renen, Mihail Stoian, Alfons Kemper, Tim Kraska, and Thomas Neumann. RadixSpline: a single-pass learned index. In *Proc. 3rd International Workshop on Exploiting Artificial Intelligence Techniques for Data Management (aiDM)*, pages 5:1–5:5, 2020. doi:10.1145/3401071.3401659.

17 Melanie Kirsche, Arun Das, and Michael C Schatz. Sapling: Accelerating suffix array queries with learned data models. *Bioinformatics*, 37(6):744–749, 2021. doi:10.1093/BIOINFORMATICS/BTAA911.

18 Tim Kraska, Alex Beutel, Ed H. Chi, Jeffrey Dean, and Neoklis Polyzotis. The case for learned index structures. In *Proc. 45th International Conference on Management of Data (SIGMOD)*. ACM, 2018. doi:10.1145/3183713.3196909.

19 Meng Li, Huayi Chai, Siqiang Luo, Haipeng Dai, Rong Gu, Jiaqi Zheng, and Guihai Chen. VEGA: an active-tuning learned index with group-wise learning granularity. *Proc. ACM Manag. Data*, 3(1):86:1–86:26, 2025. doi:10.1145/3709736.

20 Liang Liang, Guang Yang, Ali Hadian, Luis Alberto Croquevielle, and Thomas Heinis. SWIX: A memory-efficient sliding window learned index. *Proc. ACM Manag. Data*, 2(1):41:1–41:26, 2024. doi:10.1145/3639296.

21 Chaohong Ma, Xiaohui Yu, Yifan Li, Xiaofeng Meng, and Aishan Maoliniyazi. FILM: a fully learned index for larger-than-memory databases. *PVLDB*, 16(3):561–573, 2022. doi:10.14778/3570690.3570704.

22 Camille Marchet, Christina Boucher, Simon J. Puglisi, Paul Medvedev, Mikaël Salson, and Rayan Chikhi. Data structures based on k-mers for querying large collections of sequencing data sets. *Genome Research*, 31(1):1–12, 2020. doi:10.1101/gr.260604.119.

23 Paul Medvedev. Theoretical analysis of sequencing bioinformatics algorithms and beyond. *Communications of the ACM*, 66(7):118–125, 2023. doi:10.1145/3571723.

24 Yuta Mori. libdivsufsort. https://github.com/y-256/libdivsufsort/.

25 Gonzalo Navarro. *Compact data structures: A practical approach*. Cambridge University Press, 2016.

26 Gonzalo Navarro. Indexing highly repetitive string collections, Part I. *ACM Computing Surveys*, 54(2):1–31, 2021. doi:10.1145/3434399.

27 Joseph O'Rourke. An on-line algorithm for fitting straight lines between data ranges. *Communications of the ACM*, 24(9):574–578, 1981. doi:10.1145/358746.358758.

28 Giulio Ermanno Pibiri, Yoshihiro Shibuya, and Antoine Limasset. Locality-preserving minimal perfect hashing of k-mers. *Bioinformatics*, 39(Supplement_1):i534–i543, 2023. doi:10.1093/bioinformatics/btad219.

29 William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes: The Art of Scientific Computing*. Cambridge University Press, 3rd edition, 2007.

30 Arang Rhie, Sergey Nurk, Monika Cechova, Savannah J Hoyt, Dylan J Taylor, Nicolas Altemose, Paul W Hook, Sergey Koren, Mikko Rautiainen, Ivan A Alexandrov, et al. The complete sequence of a human Y chromosome. *Nature*, 621(7978):344–354, 2023. doi:10.1038/s41586-023-06457-y.

31 Geoffrey X. Yu, Markos Markakis, Andreas Kipf, Per-Åke Larson, Umar Farooq Minhas, and Tim Kraska. TreeLine: an update-in-place key-value store for modern storage. *PVLDB*, 16(1):99–112, 2022. doi:10.14778/3561261.3561270.

32 Sepanta Zeighami and Cyrus Shahabi. On distribution dependent sub-logarithmic query time of learned indexing. In *Proc. 40th International Conference on Machine Learning (ICML)*, volume 202 of *PMLR*, pages 40669–40680, 2023. URL: https://proceedings.mlr.press/v202/zeighami23a.html.