Optimally Resilient Codes for List-Decoding from Insertions and Deletions

Venkatesan Guruswami*
venkatg@cs.cmu.edu
Carnegie Mellon University
Pittsburgh, PA, USA

Bernhard Haeupler†
haeupler@cs.cmu.edu
Carnegie Mellon University
Pittsburgh, PA, USA

Amirbehshad Shahrasbi†
shahrasbi@cs.cmu.edu
Carnegie Mellon University
Pittsburgh, PA, USA

ABSTRACT

We give a complete answer to the following basic question: "What is the maximal fraction of deletions or insertions tolerable by q-ary list-decodable codes with non-vanishing information rate?"

This question has been open even for binary codes, including the restriction to the binary insertion-only setting, where the best-known result was that a $\gamma \leq 0.707$ fraction of insertions is tolerable by some binary code family.

For any desired $\varepsilon > 0$, we construct a family of binary codes of positive rate which can be efficiently list-decoded from any combination of γ fraction of insertions and δ fraction of deletions as long as $\gamma + 2\delta \leq 1 - \varepsilon$. On the other hand, for any γ, δ with $\gamma + 2\delta = 1$ list-decoding is impossible. Our result thus precisely characterizes the feasibility region of binary list-decodable codes for insertions and deletions.

We further generalize our result to codes over any finite alphabet of size q. Surprisingly, our work reveals that the feasibility region for $q > 2$ is *not* the natural generalization of the binary bound above. We provide tight upper and lower bounds that precisely pin down the feasibility region, which turns out to have a $(q-1)$-piece-wise linear boundary whose q corner-points lie on a quadratic curve.

The main technical work in our results is proving the existence of code families of sufficiently large *size* with good list-decoding properties for any combination of δ, γ within the claimed feasibility region. We achieve this via an intricate analysis of codes introduced by [Bukh, Ma; SIAM J. Discrete Math; 2014]. Finally, we give a simple yet powerful concatenation scheme for list-decodable insertion-deletion codes which transforms any such (non-efficient) code family (with vanishing information rate) into an efficiently decodable code family with constant rate.

CCS CONCEPTS

• **Mathematics of computing → Coding theory**.

*Supported in part by NSF grant CCF-1814603.
†Supported in part by NSF grants CCF-1527110, CCF-1618280, CCF-1814603, CCF-1910588, NSF CAREER award CCF-1750808 and a Sloan Research Fellowship.

STOC '20, June 22–26, 2020, Chicago, IL, USA

ACM ISBN 978-1-4503-6979-4/20/06…$15.00
https://doi.org/10.1145/3357713.3384262

KEYWORDS

Coding for Insertions and Deletions, List Decoding, Error Resilience

ACM Reference Format:
Venkatesan Guruswami, Bernhard Haeupler, and Amirbehshad Shahrasbi. 2020. Optimally Resilient Codes for List-Decoding from Insertions and Deletions. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing (STOC '20), June 22–26, 2020, Chicago, IL, USA.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3357713.3384262

1 INTRODUCTION

Error correcting codes have the ability to efficiently correct large fractions of errors while maintaining a large communication rate. The fundamental trade-offs between these two conflicting desiderata have been intensely studied in information and coding theory. Algorithmic coding theory has further studied what trade-offs can be achieved *efficiently*, i.e., with polynomial time encoding and decoding procedures.

This paper studies insdel codes, i.e., error correcting codes with a large minimum edit distance, which can correct synchronization errors such as insertions and deletions. While codes for Hamming errors and the Hamming metric are quite well understood, insdel codes have largely resisted such progress but have attracted a lot of attention recently [3, 5, 7–9, 12, 14, 16–19, 21, 25, 26]. A striking example of a basic question that is open in the context of synchronization errors is the determination of the maximal fraction of deletions or insertions a unique- or list-decodable binary code with non-vanishing rate can tolerate. That is, we do not even know at what fraction of errors the rate/distance tradeoff for insdel codes hits zero rate. These basic and intriguing questions are open even if one just asks about the existence of codes, irrespective of computational considerations, and even when restricted to the insertion-only setting.

In this paper we fully answer these questions for list-decodable binary codes and more generally for codes over any alphabet of a fixed size q. Our results are efficient and work for any combination of insertions and deletions from which list decoding is information-theoretically feasible at all.

1.1 Prior Results and Related Works

The study of codes for insertions and deletions has a long history and goes back to studies of Levenshtein[24] in the 60s. We refer to the surveys by Sloan [31], Mercier et al. [28] and Mitzenmacher [29] for a more extensive background, and focus here on works related to the main thrust of this paper, namely the maximal tolerable fraction of worst-cast deletions or insertions for unique- and list-decodable code families with non-vanishing rate. We stress that our focus is

on *worst-case* patterns of insdel errors subject to bounds on the fraction of insertions and the fraction of deletions allowed. There is also a rich body of work on tackling random insdel errors, which is not the focus of this work.

Unique Decoding. Let us first review the situation for unique decoding, where the decoder must determine the original transmitted codeword. For unique decoding of binary codes, the maximal tolerable fraction of deletions is easily seen to be at most $\frac{1}{2}$ because otherwise either all zeros or all ones in a transmitted codeword can be deleted. (For q-ary codes, this fraction becomes $1 - 1/q$.) On the other hand, for a long time the best (existential) possibility results for unique-decodable binary codes stemmed from analyzing random binary codes.

In the Hamming setting, random codes often achieve the best known parameters and trade-offs, and a lot of effort then goes into finding efficient constructions and decoding algorithms for codes that attempt to come close to the random constructions. However, the edit distance is combinatorially intricate and even analyzing the expected edit distance of two random strings, which is the first step in analyzing random codes, is highly non-trivial.

Lueker [27], improving upon earlier results by Dančík and Paterson [10, 11], proved that the expected fractional length of the longest common subsequence between two random strings lies between 0.788071 and 0.826280 (the exact value is still unknown). Using this, one can show that a random binary code of positive rate can tolerate between 0.23 and 0.18 fraction of deletions or insertions. Edit distance of random q-ary strings were studied by Kiwi, Loebl, and Matoušek[23], leading to positive rate random codes by Guruswami and Wang [14] that correct $1 - \Theta(\frac{1}{\sqrt{q}})$ fraction of deletions for asymptotically large q. Because random codes do not have efficient decoding and encoding procedures these results were purely existential. Computationally efficient binary codes of non-vanishing rate tolerating some small unspecified constant fraction of insertions and deletions were given by Schulman and Zuckerman [30]. Guruswami and Wang [14] gave binary codes that could correct a small constant fraction of deletions with rate approaching 1, and this was later extended to handle insertions as well [12].

In the regime of low-rate and large fraction of deletions, Bukh and Guruswami [4] gave a q-ary code construction that could tolerate up to a $\frac{q-1}{q+1}$ fraction of deletions, which is $\frac{1}{3}$ for binary codes. Note that this beats the performance of random codes. Together with Håstad [5] they later improved the deletion fraction to $1-\frac{2}{q+\sqrt{q}}$ or $\sqrt{2}-1 \approx 0.414$ for binary codes. This remains the best known result for unique-decodable codes and determining whether there exist binary codes capable of correcting a fraction of deletions approaching $\frac{1}{2}$ remains a fascinating open question.

List decoding. The situation for list-decodable codes over small alphabets is equally intriguing. In list-decoding, one relaxes the decoding requirement from having to output the codeword that was sent to having to produce a (polynomially) small list of codewords which includes the correct one. The trivial limit of $1/2$ fraction deletions for unique-decoding binary codes applies equally well for list-decoding. In their paper, Guruswami and Wang [14] showed that this limit can be approached by efficiently list-decodable binary codes. Similarly, q-ary codes list-decodable from a deletion fraction approaching the optimal $1 - 1/q$ bound can be constructed.

However, the situation was not well understood when insertions are also allowed. It had already been observed by Levenshtein [24] that (at least existentially) insertions and deletions are equally hard to correct for unique-decoding, in that if a code can correct t deletions then it can also correct any combination of t insertions and deletions. This turns out to be not true for list-decoding. This was demonstrated pointedly in [20], where it is shown that arbitrary large $\gamma = O(1)$ fractions of insertions (possibly exceeding 1) can be tolerated by list-decodable codes over sufficiently large constant alphabets (see Theorem 2.1), whereas the fraction of deletions δ is clearly bounded by 1. Indeed, the fraction of insertions γ does not even factor into the rate of these list-decodable insertion-deletion codes—this rate can approach the optimal bound of $1 - \delta$ where δ is the deletion fraction. The result in [20], however, applies only to sufficiently large constant alphabet sizes, and it does not shed any light on the list-decodability of *binary* (or any fixed alphabet) insdel codes.

Considering a combination of insertions and deletions, the following bound is not hard to establish.

Proposition 1.1. *For any integer q and any $\delta, \gamma \geq 0$ with $\frac{\delta}{1-\frac{1}{q}} + \frac{\gamma}{q-1} \geq 1$ there is no family of constant rate codes of length n which are list-decodable from δn deletions and γn insertions.*

For the case of insertion-only binary codes, the above limits the maximum fraction of insertions to 100%, which is twice as large as the best possible deletion fraction of $1/2$.

Turning to existence/constructions of list-decodable codes for insertions, recall that the codes of Bukh, Guruswami, Håstad (BGH) could unique-decode (and thus also list-decode) a fraction of 0.414 insertions (indeed any combination of insertions and deletions totaling 0.414 fraction). Wachter-Zeh [32] recently put forward a Johnson-type bound for insdel codes. The classical Johnson bound works in the Hamming metric, and connects unique-decoding to list-decoding (for Hamming errors) by showing that any unique-decodable code must also be list-decodable from an even larger fraction of corruptions. One intriguing implication of Wachter-Zeh's Johnson bound for insdel codes is that any unique-decodable insdel code which tolerates a $\frac{1}{2}$ fraction of deletions (or insertions) would automatically also have to be (existentially) list-decodable from a 100% fraction of insertions. Therefore, even if one is interested in unique-decoding, e.g., closing the above-mentioned gap between $\sqrt{2}-1$ and $\frac{1}{2}$, this establishes the search for maximally list-decodable binary codes from insertions as a good and indeed necessary step towards this goal. On the other hand, proving any non-trivial impossibility result bounding the maximal fraction of insertions of list-decodable binary codes away from 100% would directly imply an impossibility result for unique-decoding binary codes from a deletion fraction approaching $\frac{1}{2}$.

Follow-up work by Hayashi and Yasunaga [22] corrected some subtle but crucial bugs in [32] and reproved a corrected Johnson Bound for insdel codes. They furthermore showed that the BGH codes [5] could be list-decoded from a fraction ≈ 0.707 of insertions. Lastly, via a concatenation scheme used in [12, 14] they furthermore made these codes efficient. A recent work of Liu, Tjuawinata, and

Xing [26] also provides efficiently list-decodable insertion-deletion codes and derives a Zyablov-type bound. In summary, for the binary insertion-only setting, the largest fraction of insertions that we knew to be list-decodable (even non-constructively) was ≈ 0.707.

1.2 Our Results

We close the above gap and show binary codes which can be list-decoded from a fraction $1-\varepsilon$ fraction of insertions, for any desired constant $\varepsilon > 0$. In fact, we give a single family of codes that are list-decodable from any mixed combination of γ fraction of insertions and δ fraction of deletions, as long as $2\delta + \gamma \leq 1 - \varepsilon$.

Theorem 1.2. *For any $\varepsilon \in (0,1)$ and sufficiently large n, there exists a constant rate family of efficient binary codes that are L-list decodable from any δn deletions and γn insertions in* $\mathrm{poly}(n)$ *time as long as $\gamma + 2\delta \leq 1 - \varepsilon$ where n denotes the block length of the code, $L = O_\varepsilon(\exp(\exp(\exp(\log^* n))))$, and the code achieves a rate of* $\exp\left(-\frac{1}{\varepsilon^{10}}\log^2\frac{1}{\varepsilon}\right)$.

Since the computationally efficient codes from Theorem 1.2 match the bounds from Proposition 1.1 for every δ, γ, this nails down the entire feasibility region for list-decodability from insertions and deletions for the binary case. We stress that while we get constructive results, even the existence of inefficiently list-decodable codes, that too just for the insertion-only setting, was not known prior to this work.

In the above result, the rather weird looking bound on the list-size is inherited from results on list-decoding from a huge number insertions over larger alphabets [20], which in turn is inherited from the list-size bounds for the list-recoverable algebraic-geometric code constructions in [15].

We use similar construction techniques to obtain codes with positive rate over any arbitrary alphabet size q that are list-decodable from any fraction of insertions and deletions under which list-decoding is possible. We thus precisely identify the feasibility region for any alphabet size, together with an efficient construction. Again, recall that the existence of such codes was not known earlier, even for the insertion-only case.

Theorem 1.3. *For any positive integer $q \geq 2$, define F_q as the concave polygon defined over vertices $\left(\frac{i(i-1)}{q}, \frac{q-i}{q}\right)$ for $i = 1, \cdots, q$ and $(0,0)$. (An illustration for $q = 5$ is presented in Fig. 1). F_q does not include the border except the two segments $[(0,0),(q-1,0))$ and $[(0,0),(0,1-1/q))$. Then, for any $\varepsilon > 0$ and sufficiently large n, there exists a family of q-ary codes that, as long as $(\gamma,\delta) \in (1-\varepsilon)F_q$, are efficiently L-list decodable from any δn deletions and γn insertions where n denotes the block length of the code, $L = O(\exp(\exp(\exp(\log^* n))))$, and the code achieves a positive rate of* $\exp\left(-\frac{1}{\varepsilon^{10}}\log^2\frac{1}{\varepsilon}\right)$.

We further show in Section 5 that for any pair of positive real numbers $(\gamma,\delta) \notin F_q$, there exists no infinite family of q-ary codes with rate bounded away from zero that can be list decoded from a δ-fraction of deletions plus a γ-fraction of insertions.

1.3 Our Techniques

We achieve these results using two ingredients, each interesting in its own right. The first is a simple new concatenation scheme for

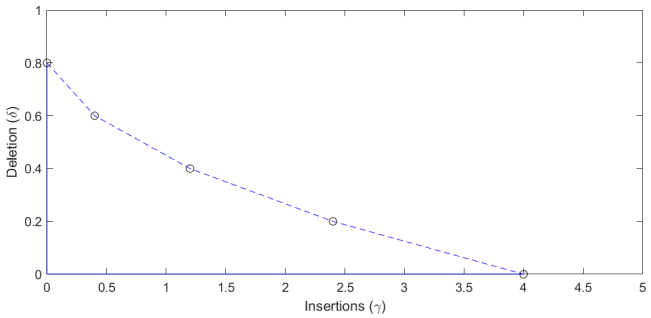

Figure 1: Feasibility region for $q = 5$.

list-decodable insdel codes which can be used to boost the rate of insdel codes. The second component, which constitutes the bulk of this work, is a technically intricate proof of the list-decoding properties of the Bukh-Ma codes [6] which have good (edit) distance properties but a tiny sub-constant rate. We note that these codes were the inner codes in the "clean construction" in the BGH work on codes unique-decodable from a 1/3 insdel fraction [5]. This was driven by a property of these codes called the *span*, which is a stronger form of edit distance that applies at all scales. The Bukh-Ma codes were also used by Guruswami and Li [13] in their existence proof of codes of positive rate for correcting a fraction of *oblivious deletions* approaching 1. In this work, the non-trivial list-decodability property of the Bukh-Ma codes drives our result.

1.3.1 *Concatenating List-Decodable Insdel Codes.*

Our first ingredient is a simple but powerful framework for constructing list-decodable insertion-deletion codes via code concatenation. Recall that code concatenation which composes the encoding of an *outer* code C_{out} with an *inner* code C_{in} whose size equals the alphabet size of C_{out}.

In our approach, the outer code C_{out} is chosen to be a list-decodable insdel code C_{out} over an alphabet that is some large function of $1/\varepsilon$, but which has constant rate and is capable of tolerating a huge number of insertions. The inner code C_{in} is chosen to be a list-decodable insdel code over a fixed alphabet of the desired size q, which has non-trivial list decoding properties for the desired fraction δ, γ of deletions and insertions.

We show that even if C_{in} has an essentially arbitrarily bad sub-constant rate and is not efficient, the resulting q-ary insdel code does have constant rate, and can also be efficiently list decoded from the same fraction of insertions and deletions as C_{in}. For the problem considered in this paper, this framework essentially provides efficiency of codes for free. More importantly, it reduces the problem of finding good *constant-rate* insdel codes over a fixed alphabet to finding a family of good list-decodable insdel codes *with an arbitrarily large number of codewords*, and a list-size bounded by some fixed function of $1/\varepsilon$.

Our decoding procedure for concatenated list-decodable insdel codes is considerably simpler than similar schemes introduced in earlier works [5, 12, 14, 30]. Of course, the encoding is simply given by the standard concatenation procedure. The decoding is done by (i) list-decoding shifted intervals of the received string using the inner code C_{in}, (ii) creating a single string from the symbols in these lists, and (iii) using the list-decoding algorithm of the outer

code on this string (viewed as a version of the outer codeword with some number of deletions and insertions).

The main driving force behind why this simplistic sounding approach actually works is a judicious choice of the outer code C_{out}. Specifically, we use the codes due to Haeupler, Shahrasbi, and Sudan [20] which can tolerate a very large number of insertions. This means that the many extra symbols coming from the list-decodings of the inner code C_{in} and the choice of overlapping intervals does not disrupt the decoding of the outer code.

1.4 Analyzing the Properties of Bukh-Ma Codes

The main technical challenge that remains is to construct or prove the existence of arbitrarily large binary codes with optimal list decoding properties for any γ, δ (and q). For this we turn to a simple family of codes introduced by Bukh and Ma [6], which consist of strings $(0^r 1^r)^{\frac{n}{r}}$ which oscillate between 0's and 1's with different frequencies. (Below we will refer to r as the *period*, and $1/r$ should be thought of as the *frequency* of alternation.)

A simple argument shows that the edit distance between any two such strings with sufficiently different periods is maximal, resulting in a tolerable fraction of edit errors of $\frac{1}{2}$ for unique decoding. The Johnson bound of [22, 32] implies that this code must also be list-decodable from a full fraction 100% of insertions. Therefore, using these codes as the inner codes in the above-mentioned concatenation scheme resolves the list-decoding question for the insertion-only setting. (The deletion-only setting is oddly easier as just random inner codes suffice, and was already resolved in [14].) This also raises hope that the Bukh-Ma codes might have good list-decoding properties for other γ, δ as well. Fortunately, this turns out to be true, though establishing this involves an intricate analysis that constitutes the bulk of the technical work in this paper.

Theorem 1.4. *For any $\varepsilon > 0$ and sufficiently large n, let $C_{n,\varepsilon}$ be the following Bukh-Ma code:*

$$C_{n,\varepsilon} = \left\{ (0^r 1^r)^{\frac{n}{2r}} \,\middle|\, r = \left(\frac{1}{\varepsilon^4}\right)^k, k < \log_{1/\varepsilon^4} n \right\}.$$

For any $\delta, \gamma \geq 0$ where $\gamma + 2\delta < 1 - \varepsilon$, $C_{n,\varepsilon}$ is list-decodable from any δn deletions and γn insertions with $O(\varepsilon^{-3})$ list size.

In order to prove Theorem 1.4 we first introduce a new correlation measure which expresses how close a string is to any given frequency (or Bukh-Ma codeword) if one allows for both insertions and deletions each weighted appropriately. Using this we want to show that it is impossible to have a single string v which is more than ε-correlated with more than $\Theta_\varepsilon(1)$ frequencies.

Intuitively, one might expect that each correlation can be (fractionally) attributed to a (disjoint) part of v which would result in the maximum number of ε-close frequencies to be at most $1/\varepsilon$. This, however, turned out to be false. Instead, we use a proof technique which is somewhat reminiscent of the one used to establish the polarization of the martingale of entropies in the analysis of polar codes [1, 2].

In more detail, we think of recursively sub-sampling smaller and smaller nested substrings of v, and analyze the expectation and variance of the bias between the fraction of 0's and 1's in these substrings. More precisely, we order the run lengths $r_1, r_2, \ldots$ that are ε-correlated with v in decreasing order and first sample a substring v_1 with $r_1 \gg |v_1| \gg r_2$ from v. While the expected zero-one bias in v_1 is the same as in v, we show that the variance of this bias is an increasing function in the correlation with $(0^{r_1}1^{r_1})^{\frac{n}{2r_1}}$. Intuitively, v_1 cannot be too uniform on an scale of length l if it is correlated with r_1.

Put differently, in expectation the sampled substring v_1 will land in a part of v which is either (slightly) correlated to one of the long stretches of zeros in v or in a part which is correlated with a long stretch of ones in v, resulting in at least some variance in the bias of v_1. Because the scales $r_2, r_3, \ldots$ are so much smaller than v_1, this sub-sampling of v_1 furthermore preserves the correlation with these scales intact, at least in expectation.

Next we sample a substring v_2 with $r_2 \gg |v_2| \gg r_3$ within v_1. Again, the bias in v_2 stays the same as the one in v_1 in expectation but the sub-sampling introduces even more variance given that v_1 is still non-trivially correlated with the string with period r_2. The evolution of the bias of the strings $v_1, v_2, \ldots$ produced by this nested sampling procedure can now be seen as a martingale with the same expectation but an ever increasing variance. Given that the bias is bounded in magnitude by 1, the increase in variance cannot continue indefinitely. This limits the number of frequencies a string v can be non-trivially correlated with, which is exactly what we were after.

Our generalization to larger q-ary alphabets follows the same high level blueprint, but is technically even more delicate. Recall that in the non-binary case, there are $(q-1)$ different linear trade-offs between δ, γ depending on the exact regime they lie in.

2 PRELIMINARIES

2.1 List-Decodable Insertion-Deletion Codes

The following list-decodable insertion-deletion codes from [20] will be used as the outer code in our constructions.

Theorem 2.1 (Theorem 1.1 from [20]). *For every $\delta, \varepsilon \in (0,1)$ and constant $\gamma > 0$, there exist a family of list-decodable insdel codes that can protect against δ-fraction of deletions and γ-fraction of insertions and achieves a rate of $1 - \delta - \varepsilon$ or more over an alphabet of size $\left(\frac{\gamma+1}{\varepsilon^2}\right)^{O\left(\frac{\gamma+1}{\varepsilon^3}\right)} = O_{\gamma,\varepsilon}(1)$. These codes are list-decodable with lists of size $L_{\varepsilon,\gamma}(n) = \exp(\exp(\exp(\log^* n)))$, and have polynomial time encoding and decoding complexities.*

2.2 Strings, Insertions, Deletions, and Distances

In this section we provide preliminary definitions on strings, edit operations, and related notions.

Definition 2.2 (Count and Bias). We define $\text{count}_a(w) = |\{i | w[i] = a\}|$ as the number of appearances of symbol a in string w. The bias of a binary string w is the normalized difference between the appearances of zeros and ones in w, i.e., $\text{bias}(w) = \frac{\text{count}_1(w) - \text{count}_0(w)}{|w|}$. With this definition, $\text{count}_0(w) = \frac{1-\text{bias}(w)}{2}|w|$ and $\text{count}_1(w) = \frac{1+\text{bias}(w)}{2}|w|$.

Definition 2.3 (Matching). A matching M of size k between two strings S and S' is defined to be two sequences of k integer positions $0 < i_1 < \ldots < i_k \leq |S|$ and $0 < i'_1 < \ldots < i'_k \leq |S'|$ for which

$S[i_j] = S'[i'_j]$ for all $j \le k$. The subsequence induced by a matching M is simply $S[i_1], \ldots, S[i_k]$. Every common subsequence between S and S' implicitly corresponds to a matching and we use the two interchangeably.

Definition 2.4 (Advantage of a Matching). Let M be a matching between two binary strings a and b. The *advantage of the matching* M is defined as $\mathrm{adv}_M = \frac{3|M|-|a|-|b|}{|a|}$.

Definition 2.5 (Advantage). For a given pair of strings a and b, the *advantage of a to b* is defined as the advantage of the matching M that corresponds to the largest common subsequence between them, i.e., $\mathrm{adv}(a,b) = \mathrm{adv}_{M=\mathrm{LCS}(a,b)}$. It is easy to verify that the longest common subsequence M maximizes the advantage among all matchings from a to b.

We now make the following remark that justifies the notion of advantage as defined above. Note that any matching between two strings a and b implies a set of insertions and deletions to convert b to a which is, to delete all unmatched symbols in b and insert all unmatched symbols in a within the remaining symbols.

Remark 2.6. *Consider strings a and b and matching M between them. Think of a as a distorted version of b and let δ_M and γ_M represent the fraction of deletions and insertions needed to convert b to a as suggested by M, i.e., $\delta_M = \frac{\text{Number of unmatched symbols in } b}{|b|} = \frac{|b|-|M|}{|b|}$, and $\gamma_M = \frac{\text{Number of unmatched symbols in } a}{|b|} = \frac{|a|-|M|}{|b|}$. The adv_M function tracks the value of $|b|(1-2\delta_M-\gamma_M)$ normalized by $|a|$ rather than $|b|$.*

$$adv_M(a,b) = \frac{3|M|-|a|-|b|}{|a|}$$
$$= \frac{3|b|(1-\delta_M)-|b|(1-\delta_M+\gamma_M)-|b|}{|a|} = \frac{|b|}{|a|}\cdot(1-2\delta_M-\gamma_M)$$

We will make use of this unnatural normalization later on.

We now extend the definition of advantage to the case where the second argument is an infinite string.

Definition 2.7 (Infinite Advantage). For a finite string a and infinite string b, the advantage of a to b is defined as the minimum advantage that a has over all substrings of b.

$$\mathrm{adv}(a,b) = \min_{b'=b[i,j]} \mathrm{adv}(a,b').$$

We now define a family of binary strings called *Alternating Strings*.

Definition 2.8 (Alternating Strings). For any positive integer r, we define the infinite alternating string of run-length r as $A_r = (0^r 1^r)^\infty$ and denote its prefix of length l with $A_{r,l} = A_r[1,l]$.

We finish the preliminaries by the following lemma stating some properties of the notions defined through this section.

Lemma 2.9. *The following properties hold true:*

- *For any pair of binary strings S_1, S_2 where $adv(S_1,S_2) > 0$, lengths of S_1 and S_2 are within a factor of two of each other, i.e, $\min(|S_1|,|S_2|) \ge \frac{\max(|S_1|,|S_2|)}{2}$.*
- *For any binary string S and integer r, $adv(S,A_r) \ge -\frac{1}{2}$*

Proof. For the first part, let $M = \mathrm{LCS}(S_1,S_2)$. We have that $\mathrm{adv}(S_1,S_2) \ge 0 \Rightarrow 3|M| \ge |S_1|+|S_2|$, which, as $|M| \le \min(|S_1|,|S_2|)$, implies that $\min(|S_1|,|S_2|) \ge \frac{\max(|S_1|,|S_2|)}{2}$.

For the second part, let $n = |S|$ and assume that $b \in \{0,1\}$ is the most frequent bit in S and there are m occurrences of b in S. Take a substring S' in A_r as the smallest string that starts at the beginning of a b^r block and contains the same number of bs as S. The size of S' is no more than $2m$ and the longest common subsequence between S and S' is at least m. Therefore, $\mathrm{adv}(S,A_r) \ge \mathrm{adv}(S,S') \ge \frac{3|M|-|S|-|S'|}{|S|} \ge \frac{3m-2m-2m}{n} \ge \frac{-m}{n} \ge -\frac{1}{2}$. □

3 PROOF OF THEOREM 1.4: LIST-DECODING FOR BUKH-MA CODES

To prove this theorem, we assume for the sake of contradiction that there exists a string v and $k > \frac{1200}{\varepsilon^3}$ members of $C_{n,\varepsilon}$ like $A_{r_1,n}, A_{r_2,n}, \cdots, A_{r_k,n}$, so that each $A_{r_i,n}$ can be converted to v with I_i insertions and D_i deletions where $I_i+2D_i \le n(1-\varepsilon)$. We define the indices in a way that $r_1 > r_2 > \cdots > r_k$. Given the definition of $C_{n,\varepsilon}$, $r_i \ge \frac{r_{i+1}}{\varepsilon^4}$. We first show that, for $i = 1,2,\cdots,k$, $\mathrm{adv}(v,A_{r_i,n}) \ge \frac{\varepsilon}{2}$.

Lemma 3.1. *For any $1 \le i \le k$, $adv(v,A_{r_i,n}) \ge \frac{\varepsilon}{2}$.*

Proof. Let M_i denotes the matching that corresponds to the set of I_i insertions and D_i deletions that convert $A_{r_i,n}$ to v.

$$I_i + 2D_i \le n(1-\varepsilon) \Rightarrow n - I_i - 2D_i \ge n\varepsilon \Rightarrow 1-\gamma_i-2\delta_i \ge \varepsilon$$

Note that according to Remark 2.6, $\mathrm{adv}(v,A_{r_i,n}) = \frac{n}{|v|}\cdot(1-\gamma_i-2\delta_i)$. Thus, $\mathrm{adv}(v,A_{r_i,n}) \ge \frac{n}{|v|}\varepsilon \ge \frac{\varepsilon}{2}$. The last step follows from the first item of Lemma 2.9. □

Having Lemma 3.1, we are ready to prove Theorem 1.4. We start with defining a couple of sequences of random variables via random sampling of nested substrings of v. We split the string v into substrings of size $l_1 = r_1\varepsilon^2$, pick one uniformly at random and denote it by v_1. We define random variable $A_1 = \mathrm{adv}(v_1,A_{r_1})$ and random variable $B_1 = \mathrm{bias}(v_1)$. Similarly, we split v_1 into substrings of length $l_2 = r_2\varepsilon^2$ and pick v_2 uniformly at random and define $A_2 = \mathrm{adv}(v_2,A_{r_2})$ and $B_2 = \mathrm{bias}(v_2)$. Continuing this procedure, one can obtain the two sequences of random variables $A_1, A_2, \cdots, A_k$ and $B_1, B_2, \cdots, B_k$. We will prove the following.

Lemma 3.2. *The following hold for $A_1,\cdots,A_k$ and $B_1,\cdots,B_k$:* (1) $\mathbb{E}[B_i] = bias(v)$, (2) $\mathbb{E}[A_i] \ge \frac{\varepsilon}{2}$.

Proof. Note that one can think of v_i as a substring of v that is obtained by splitting v into substrings of length l_i and choosing one uniformly at random. Let U denote the set of all such substrings. We have that

$$\mathbb{E}[B_i] = \sum_{\hat{v}\in U}\frac{1}{|U|}\cdot\mathrm{bias}(\hat{v}) = \frac{1}{|U|}\sum_{\hat{v}\in U}\frac{\mathrm{count}_1(\hat{v})-\mathrm{count}_0(\hat{v})}{l_i}$$
$$= \frac{\mathrm{count}_1(v)-\mathrm{count}_0(v)}{|U|\cdot l_i} = \mathrm{bias}(v).$$

A similar argument proves the second item. Take the matching M_i between v and $A_{r_i,n}$ that achieves the advantage $\mathrm{adv}(v,A_{r_i,n})$, i.e., the largest matching between v and $A_{r_i,n}$. Take some $\hat{v} \in U$; $\hat{v}$ is mapped to some substring in $A_{r_i,n}$ under M_i. We call that substring of $\hat{v}$, *the projection of $\hat{v}$ under M_i* and denote it by $\hat{v} \to M_i$. We also

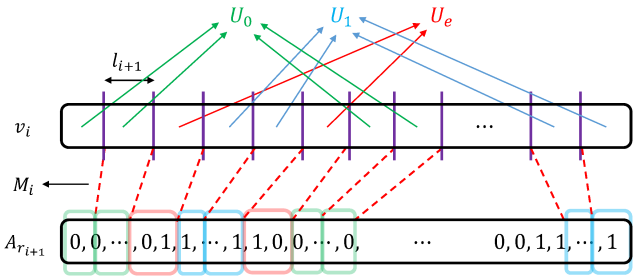

Figure 2: Partitioning substrings of length l_{i+1} into three sets U_0, U_1, U_e

represent the subset of M_i that appears between $\hat{v}$ and $\hat{v} \to M_i$ with $M_i[\hat{v}]$.

For a $\hat{v} \in U$, we define $a(\hat{v})$ as the value for advantage that is yielded by the matching $M_i[\hat{v}]$ between $\hat{v}$ and $\hat{v} \to M_i$. In other words, $a(\hat{v}) = \frac{3|M_i[\hat{v}]| - |\hat{v}| - |\hat{v} \to M_i|}{|\hat{v}|}$. Given the definitions of advantage and infinite advantage, we have that $a(\hat{v}) \le \text{adv}(\hat{v}, \hat{v} \to M_i) \le \text{adv}(\hat{v}, A_{r_i})$. This can be used to prove the second item as follows:

$$\begin{aligned}
\mathbb{E}[A_i] &= \sum_{\hat{v} \in U} \frac{1}{|U|} \cdot \text{adv}(\hat{v}, A_{r_i}) \ge \frac{1}{|U|} \cdot \sum_{\hat{v} \in U} a(\hat{v}) \\
&= \frac{1}{|U|} \cdot \sum_{\hat{v} \in U} \frac{3|M_i[\hat{v}]| - |\hat{v}| - |\hat{v} \to M_i|}{|\hat{v}|} \\
&= \frac{1}{|U| \cdot |\hat{v}|} \cdot \sum_{\hat{v} \in U} \left(3|M_i[\hat{v}]| - |\hat{v}| - |\hat{v} \to M_i|\right) \\
&= \frac{1}{|v|} \cdot \left(3|M_i| - |v| - |A_{r_i,n}|\right) = \text{adv}(v, A_{r_i,n}) \ge \frac{\varepsilon}{2}
\end{aligned}$$

where the last step follows from Lemma 3.1. □

Lemma 3.3. *For the sequence $B_1, B_2, \cdots, B_k$, we have* $\text{Var}(B_{i+1}) \ge \text{Var}(B_i) + \frac{\varepsilon^3}{1200}, \quad \forall 1 \le i < k.$

Proof. To analyze the relation of $\text{Var}(B_i)$ and $\text{Var}(B_{i+1})$, we use the law of total variance and condition the variance of B_{i+1} on v_i, i.e., the substring chosen in the ith step of the stochastic process, from which we sub sample v_{i+1}.

$$\begin{aligned}
\text{Var}(B_{i+1}) &= \text{Var}\left(\mathbb{E}[B_{i+1}|v_i]\right) + \mathbb{E}\left[\text{Var}(B_{i+1}|v_i)\right] \\
&= \text{Var}\left(B_i\right) + \mathbb{E}\left[\text{Var}(B_{i+1}|v_i)\right] \qquad (1)
\end{aligned}$$

Equation (1) comes from the fact that the average bias of substrings of length l_{i+1} in v_i is equal to the bias of v_i. Having this, we see that it suffices to show that $\mathbb{E}\left[\text{Var}(B_{i+1}|v_i)\right] \ge \varepsilon^3/1200$. We remind the reader that v_{i+1} is obtained by splitting v_i into substrings of length $l_{i+1} = r_{i+1}\varepsilon^2$ and choosing one at random. We denote the set of such substrings by U. Also, there is a matching M_i between v_i and $A_{r_{i+1}}$ with advantage ε or more. Any substring of length l_{i+1} is mapped to some substring in $A_{r_{i+1}}$, i.e., its projection of the substring under M_i. Note there are three different possibilities for such projection. It is either an all zeros string, an all one string, or a string that contains both zeros and ones. We partition U into three sets U_0, U_1, and U_e based on which case the projection belongs to. (See Fig. 2)

We partition the sample space into three events E_0, E_1, and E_e based on whether v_{i+1} belongs to U_0, U_1, or U_e respectively. We also define the random variable T over $\{0, 1, e\}$ that indicates which one of E_0, E_1, or E_e happens. Once again, we use the law of total variance to bound $\mathbb{E}\left[\text{Var}(B_{i+1}|v_i)\right]$.

$$\begin{aligned}
\mathbb{E}\left[\text{Var}(B_{i+1}|v_i)\right] &= \mathbb{E}_{v_i}\Big[\text{Var}_T(\mathbb{E}\left[B_{i+1}|v_i, T\right]) \\
&\quad + \mathbb{E}_T\left[\text{Var}(B_{i+1}|v_i, T)\right]\Big] \\
&\ge \mathbb{E}_{v_i}\Big[\text{Var}_T(\mathbb{E}\left[B_{i+1}|v_i, T\right])\Big] \qquad (2)
\end{aligned}$$

Note that the term $\text{Var}_T(\mathbb{E}\left[B_{i+1}|v_i, T\right])$ refers to variance of a 3-valued random variable that takes the value $\mathbb{E}_{v_i}\left[B_{i+1}|v_i, T = t\right]$ with probability $\Pr\{T = t|v_i\}$ for $t \in \{0, 1, e\}$. We use three important facts about this distribution to bound its variance from below.

First, $\Pr\{T = e|v_i\} \le 2\varepsilon^2$. To see this, note that the run length of $A_{r_{i+1}}$ is $r_{i+1} = \frac{l_{i+1}}{\varepsilon^2}$ and the length of the projection of v_i in A_{r_i} under the matching that yields the optimal $\text{adv}(v_i, A_{r_i})$ is no more than $2|v_i| = 2l_i$ (See Lemma 2.9). Therefore, $|U_e| \le \frac{2l_i}{r_{i+1}}$ and consequently no more that a $\frac{2l_i/r_{i+1}}{l_i/l_{i+1}} = 2\varepsilon^2$ fraction of strings in U might be mapped to a substring of $A_{r_{i+1}}$ that crosses the border of some $0^{r_{i+1}}$ and $1^{r_{i+1}}$ intervals.

Secondly, for any $j \in \{0, 1\}$, $\Pr\{T = j|v_i\} \ge \frac{\text{adv}(v_i, A_{r_{i+1}}) - 8\varepsilon^2}{8}$. This can be showed as follows. Let M_i^j represent the subset of pairs of M_i with one end in U_j for $j \in \{0, 1, e\}$ and $v_i \to M_i$ represent the substring of $A_{r_{i+1}}$ where v_i is projected under M_i. Note that $\Pr\{T = j|v_i\} = \frac{|U_j|}{|U|} = \frac{|U_j| \cdot l_i}{|v_i|} \ge \frac{|M_i^j|}{|v_i|} \ge \frac{|M_i^j|}{2|v_i \to M_i|}$. Assume for contradiction that $\Pr\{T = j|v_i\} < \frac{\text{adv}(v_i, A_{r_{i+1}}) - 8\varepsilon^2}{8}$ for some j. Then, $|M_i^j| < |v_i \to M_i| \frac{\text{adv}(v_i, A_{r_{i+1}}) - 8\varepsilon^2}{4}$, which since $|M_i^{j'}| \le \frac{|v_i \to M_i|}{2}$ for $j' \in \{0, 1\}$ and $|M_i^e| \le 2\varepsilon^2 |v_i \to M_i|$, gives that $|M_i| < |v_i \to M_i| \left(\frac{1}{2} + 2\varepsilon^2 + \frac{\text{adv}(v_i, A_{r_{i+1}}) - 8\varepsilon^2}{4}\right) = |v_i \to M_i| \left(\frac{1}{2} + \frac{\text{adv}(v_i, A_{r_{i+1}})}{4}\right)$. However, $\text{adv}_{M_i} = \frac{3|M_i| - |v_i| - |p|}{|v_i|} \Rightarrow 2|M_i| - |p| \ge |v_i|\text{adv}_{M_i} \Rightarrow |M_i| \ge |p|\left(\frac{1}{2} + \frac{\text{adv}_{M_i}}{4}\right)$. This contradiction implies that $\Pr\{T = j|v_i\} \ge \frac{\text{adv}(v_i, A_{r_{i+1}}) - 8\varepsilon^2}{8}$.

The third and final important ingredient is provided by the following lemma that we prove later on.

Lemma 3.4. *The following holds true:*

$$\left|\mathbb{E}\left[B_{i+1}|v_i, T = 0\right] - \mathbb{E}\left[B_{i+1}|v_i, T = 1\right]\right| \ge \frac{adv(v_i, A_{r_{i+1}}) - 5\varepsilon^2}{3}$$

To summarize, the above three properties imply that we have a three-valued random variable where the probability for one value is minuscule and there is at least $[\text{adv}(v_i, A_{r_{i+1}}) - 5\varepsilon^2]/3$ difference between the other two values each occurring with adequately large probabilities. This is enough for us to bound below the variance of such random variable. The following straightforward lemma abstracts this.

Lemma 3.5. *Let X be a random variable that can take values a_0, a_1, and a_2 where $\Pr\{X = a_i\} \ge \xi$ for $i \in \{0, 1\}$. Then, we have that* $\text{Var}(X) \ge \frac{\xi}{2}(a_0 - a_1)^2$.

Applying Lemma 3.5 to our random variable gives that:

$$\text{Var}_T(\mathbb{E}\left[B_{i+1}|v_i, T\right]) \ge \frac{\left(\text{adv}(v_i, A_{r_{i+1}}) - 8\varepsilon^2\right)\left(\text{adv}(v_i, A_{r_{i+1}}) - 5\varepsilon^2\right)^2}{144}$$

Note the right hand side of this inequality is negative when $\text{adv}(v_i, A_{r_{i+1}}) \leq 8\varepsilon^2$. Therefore, we define function $g(x)$ as a function that takes value of $\frac{(x-8\varepsilon^2)(x-5\varepsilon^2)}{144}$ when $x > 8\varepsilon^2$ and zero otherwise. Note that g is a convex function. We have that

$$\text{Var}_T(\mathbb{E}[B_{i+1}|v_i, T]) \geq g(\text{adv}(v_i, A_{r_{i+1}})) \tag{3}$$

Plugging (3) into (2) gives that

$$\begin{aligned}
\mathbb{E}[\text{Var}(B_{i+1}|v_i)] &\geq \mathbb{E}_{v_i}\left[\text{Var}_T(\mathbb{E}[B_{i+1}|v_i, T])\right] \\
&\geq \mathbb{E}_{v_i}\left[g(\text{adv}(v_i, A_{r_{i+1}}))\right] \\
&\geq g\left(\mathbb{E}_{v_i}\left[\text{adv}(v_i, A_{r_{i+1}})\right]\right) = g(\mathbb{E}[A_{i+1}]) \quad (4) \\
&\geq g(\varepsilon/2) = \varepsilon^3/1152 + o(\varepsilon^3) \quad (5)
\end{aligned}$$

where (4) follows from the Jensen inequality and (5) follows from Lemma 3.2 and the fact that g is an increasing function. Note that the right hand side is at least $\frac{\varepsilon}{1200}$ for sufficiently small ε. This completes the proof of Lemma 3.3 (With the exception of Lemma 3.4). □

With Lemma 3.3 proved, one can easily prove Theorem 1.4.

Proof of Theorem 1.4. Since $\text{Var}(B_{i+1}) \geq \text{Var}(B_i) + \varepsilon^3/1200$, we have that $\text{Var}(B_k) \geq \text{Var}(B_1) + (k-1)\frac{\varepsilon^3}{1200} \geq \frac{(k-1)\varepsilon^3}{1200}$. If $k > \frac{1200}{\varepsilon^3}$, the above inequality implies that $\text{Var}(B_k) > 1$ which is impossible since B_k takes value in $[-1, 1]$. This contradiction implies that the list size $k \leq \frac{1200}{\varepsilon^3}$. □

We now proceed to the proof of Lemma 3.4.

3.1 Proof of Lemma 3.4

Consider v_i and the matching that yields the optimal advantage from v_i to $A_{r_{i+1}}$, denoted by M_i. We denote the substring of $A_{r_{i+1}}$ that is identified by the projection of v_i under M_i as $p = v_i \rightarrow M_i$. To simplify the analysis, we perform a series of transformations on v_i, M_i, and p that does not decrease adv_{M_i} except by a small quantity. Fig. 3 depicts the steps of this transformation described below.

(1) First, we delete all substrings of U_e—i.e., substrings of length l_i in v_i whose projection contain both 0s and 1s—from v_i.
(2) We reorder the substrings of length l_{i+1} in v_i by shifting all U_0 substrings together and all U_1 substrings together. We accordingly shift the projections of these strings in p to the similar order. This was, the remainder of M_i from step 1 will be preserved as a valid matching between reordered strings.
(3) At this point, string p consists of a stretch of zeros followed by a stretch of ones. If the length of two stretches are not equal, we add adequate zeros or ones to the smaller stretch to make p have the form of 0^t1^t.

To track the changes in adv_{M_i} during this transformation, we track how $|M_i|$, $|v_i|$ and $|p|$ change throughout the three steps mentioned above.

In the first step, a total of up to $|U_e|l_{i+1}$ elements are removed from v_i and M_i. Note that since the run length of $A_{r_{i+1}}$ is r_{i+1}, there can only be $\frac{|p|}{r_{i+1}}$ substrings in U_e. Therefore, $|U_e|l_{i+1} \leq \frac{|p|l_{i+1}}{r_{i+1}} = |p|\varepsilon^2 \leq 2\varepsilon^2|v_i|$.

The second step preserves $|M_i|$, $|v_i|$ and $|p|$.

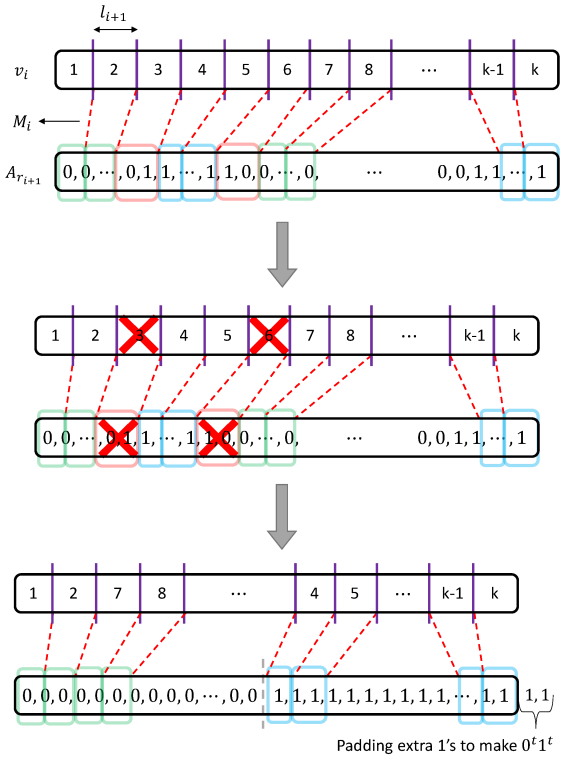

Figure 3: Three steps of transformation in Lemma 3.4.

Finally, since p is a substring of $A_{r_{i+1}}$, the third step increases $|p|$ only by up to r_{i+1}. Note the run length of the $A_{r_{i+1}}$s and consequently l_{i+1}s are different by a multiplicative factor of at least $\frac{1}{\varepsilon^4}$ by the definition of the code C. Therefore, $r_{i+1} = \frac{l_{i+1}}{\varepsilon^2} = \frac{l_{i+1}|v_i|}{\varepsilon^2|v_i|} = \frac{l_{i+1}|v_i|}{\varepsilon^2 l_i} \leq \varepsilon^2|v_i|$.

Overall, the value of the $\text{adv}_{M_i} = \frac{3|M|-|p|-|v_i|}{|v_i|}$ can be affected by a maximum of $(3-1) \times 2\varepsilon^2|v_i| + \varepsilon^2|v_i| = 5\varepsilon^2|v_i|$ decrease in the numerator and $\varepsilon^2|v_i|$ decrease in the denominator. Therefore, the eventual advantage does not drop below $\text{adv}_{M_i} - 5\varepsilon^2$. Let us denote the transformed versions of v_i, p, and M_i by $\bar{v}_i$, $\bar{p}$, and $\bar{M}_i$ respectively. We have shown that

$$\text{adv}_{\bar{M}_i} \geq \text{adv}_{M_i} - 5\varepsilon^2. \tag{6}$$

Further, let $\bar{v}_i = (\bar{v}_i^0, \bar{v}_i^1)$ so that $\bar{v}_i^0$ and $\bar{v}_i^1$ respectively correspond to the part of $\bar{v}_i$ that is mapped to 0^t and 1^t under $\bar{M}_i$. Consider the matching between $\bar{v}_i$ and $\bar{p}$ that connects as many zeros as possible between the $\bar{v}_i^0$ and 0^t and as many ones as possible between the $\bar{v}_i^1$ to 1^t portion of $\bar{p}$. Clearly, the size of $\bar{M}_i$ cannot exceed the size of this matching and therefore,

$$\text{adv}_{\bar{M}_i} \leq \frac{3\left[\min\{t, \text{count}_0(\bar{v}_i^0)\} + \min\{t, \text{count}_1(\bar{v}_i^1)\}\right] - |\bar{v}_i| - 2t}{|\bar{v}_i|} \tag{7}$$

Note that as long as $t < \text{count}_0(\bar{v}_i^0)$ or $t < \text{count}_1(\bar{v}_i^1)$, increasing t in the right hand side term does not make it smaller. Therefore, the inequality (7) holds for $t = \max_{j\in\{0,1\}}\{\text{count}_j(\bar{v}_i^j)\}$. Without loss of generality, assume that $\text{count}_0(\bar{v}_i^0) \leq \text{count}_1(\bar{v}_i^1)$ and set

$t = \text{count}_1(\bar{v}_i^1)$. Then we have the following.

$$\begin{aligned}
&\text{adv}_{\bar{M}_i} \le \frac{3\text{count}_0(\bar{v}_i^0) + \text{count}_1(\bar{v}_i^1) - |\bar{v}_i|}{|\bar{v}_i|} \\
\Rightarrow\ &\text{adv}_{\bar{M}_i} \le \frac{3\frac{1-\text{bias}(\bar{v}_i^0)}{2}|\bar{v}_i^0| + \frac{1+\text{bias}(\bar{v}_i^1)}{2}|\bar{v}_i^1| - (|\bar{v}_i^0| + |\bar{v}_i^1|)}{|\bar{v}_i|} \\
\Rightarrow\ &2\text{adv}_{\bar{M}_i}|\bar{v}_i| \le 3(1-\text{bias}(\bar{v}_i^0))|\bar{v}_i^0| \\
&\quad + (1+\text{bias}(\bar{v}_i^1))|\bar{v}_i^1| - 2(|\bar{v}_i^0| + |\bar{v}_i^1|) \\
\Rightarrow\ &2\text{adv}_{\bar{M}_i}|\bar{v}_i| \le \left[1-3\text{bias}(\bar{v}_i^0)\right]|\bar{v}_i^0| - \left[1-\text{bias}(\bar{v}_i^1)\right]|\bar{v}_i^1| \qquad (8)
\end{aligned}$$

We claim that the above inequality leads to the fact that $|\text{bias}(\bar{v}_i^1) - \text{bias}(\bar{v}_i^0)| \ge \text{adv}_{\bar{M}_i}/3$. Assume for contradiction that this is not the case. Therefore, replacing the term $\text{bias}(\bar{v}_i^0)$ with $\text{bias}(\bar{v}_i^1)$ in (8) does not change the value of the right hand side by any more than $|\bar{v}_i| \cdot \text{adv}_{\bar{M}_i}$. Same holds true with replacing the term $\text{bias}(\bar{v}_i^1)$ with $\text{bias}(\bar{v}_i^0)$ in (8). This implies that, with $b^* = \max\{\text{bias}(\bar{v}_i^0), \text{bias}(\bar{v}_i^1)\}$, we have that $\text{adv}_{\bar{M}_i}|\bar{v}_i| \le (1-3b^*)\cdot|\bar{v}_i^0| - (1-b^*)|\bar{v}_i^1|$ and, therefore,

$$\left(1-b^*\right)|\bar{v}_i^1| < \left(1-3b^*\right)|\bar{v}_i^0| \qquad (9)$$

On the other hand, we assumed earlier (without loss of generality) that $\text{count}_0(\bar{v}_i^0) \le \text{count}_1(\bar{v}_i^1)$. Therefore,

$$\begin{aligned}
&\left(1-\text{bias}(\bar{v}_i^0)\right)|\bar{v}_i^0| \le \left(1+\text{bias}(\bar{v}_i^1)\right)|\bar{v}_i^1| \\
\Rightarrow\quad &\left(1-b^*\right)|\bar{v}_i^0| \le \left(1+b^*\right)|\bar{v}_i^1| \qquad (10)
\end{aligned}$$

Note that since $|b^*| \le 1$, $(1-b^*)^2 > (1+b^*)(1-3b^*) \Rightarrow \frac{1-3b^*}{1-b^*} < \frac{1-b^*}{1+b^*}$. Multiplying the two sides of this inequality to the sides of (10) gives that $(1-3b^*)|\bar{v}_i^0| \le (1+b^*)|\bar{v}_i^1|$ which contradicts (9). Therefore, we must have $|\text{bias}(\bar{v}_i^1) - \text{bias}(\bar{v}_i^0)| \ge \text{adv}_{\bar{M}_i}/3$. Note that $\text{bias}(\bar{v}_i^j) = \mathbb{E}\left[B_{i+1}|v_i, T=j\right]$ since $\text{bias}(\bar{v}_i^j)$ is the average bias of all strings in U_j. Therefore, combining with (6), we have that

$$\left|\mathbb{E}\left[B_{i+1}|v_i, T=0\right] - \mathbb{E}\left[B_{i+1}|v_i, T=1\right]\right| \ge \frac{\text{adv}(v_i, A_{r_{i+1}}) - 5\varepsilon^2}{3}. \quad \square$$

4 PROOF OF THEOREM 1.2: CONCATENATED INSDEL CODES

We recall that the concatenation of an inner insdel code C_{in} over an alphabet of size $|\Sigma_{\text{in}}|$ and an outer insdel code, C_{out}, over an alphabet of size $|\Sigma_{\text{out}}| = |C_{\text{in}}|$ as a code over alphabet Σ_{in}, is obtained by taking each codeword $x \in C_{\text{out}}$, encoding each symbol of x with C_{in}, and appending the encoded strings together to obtain each codeword of the concatenated code.

In this section, we will show that, concatenating an inner code C_{in} from Theorem 1.4 that can L_{in}-list decode from any γ fraction of insertions and δ fraction deletions when $2\delta + \gamma < 1 - \varepsilon_{\text{in}}$ along with an appropriately chosen outer code C_{out} from Theorem 2.1, one can obtain an infinite family of constant-rate insertion-deletion codes that are efficiently list-decodable from any γ fraction of insertions and δ fraction of deletions as long as $2\delta + \gamma < 1 - \varepsilon$ for $\varepsilon = \frac{16}{5}\varepsilon_{\text{in}}$.

4.1 Construction of the Concatenated Code

We start by fixing some notation. Let C_{out} be able to L_{out}-list decode from δ_{out} fraction of deletions and γ_{out} fraction of insertions.

$$\varepsilon \longrightarrow \varepsilon_{\text{in}} \longrightarrow L_{\text{in}} \longrightarrow \delta_{out}, \gamma_{\text{out}} \longrightarrow \Sigma_{out} \longrightarrow |\mathcal{C}_{in}|$$

Figure 4: The order of determining parameters in the proof of Theorem 1.2.

Further, let us indicate the block sizes of C_{out} and C_{in} with n_{out} and $n_{\text{in}} = \lceil \log |\Sigma_{\text{out}}| \rceil$.

To construct our concatenated codes, we utilize Theorem 2.1 to obtain an efficient family of codes C_{out} over alphabet Σ_{out} of size $O_{\gamma_{\text{out}}, \delta_{\text{out}}}(1)$ that is L_{out}-list decodable from any δ_{out} fraction of deletions and γ_{out} fraction of insertions for appropriate parameters δ_{out} and γ_{out} that we determine later. We then concatenate any code in C_{out} with an instance of the binary list-decodable codes from Theorem 1.4, C_{in}, with parameter $n_{\text{in}} = \lceil \log |\Sigma_{\text{out}}| \rceil$ and a properly chosen ε_{in}. We will determine appropriate values for all these parameters given ε when describing the decoding procedure in Section 4.2. Fig. 4 shows the order of determining all parameters. We remark that the following two properties for the utilized inner and outer codes are critical to this order of fixing parameters:

(1) The alphabet size of the family of codes used as the outer code only depends on δ_{out} and γ_{out} and is independent of the outer block size n_{out}. (See Theorem 2.1)
(2) The list size of the family of codes used as the inner code, L_{in}, merely depends on parameter ε_{in} in Theorem 1.4 and is independent of the size of the code or its block length, i.e., $|C_{\text{in}}|$ or n_{in}.

4.2 Decoding Procedure and Parameters

We now analyze the resulting family of codes and choose the undetermined parameters along the way of describing the decoding procedure. A pseudo-code of the decoding procedure is available in Algorithm 1. Let C be a binary code with block length n that is obtained from the above-mentioned concatenation. Take the codeword $x \in C$ and split it into *blocks* of length n_{in}. Note that each such block corresponds to the encoding of some symbol in Σ_{out} under C_{in}. Let x' be a string obtained by applying $n\gamma$ insertions and $n\delta$ deletions into x where $n = n_{\text{in}}n_{\text{out}}$ and $\gamma + 2\delta < 1 - \varepsilon$. For each block of x, we define the error count to be the total number of insertions that have occurred in that block plus twice the number of deleted symbols in it. Clearly, the average value of error count among all blocks is $n_{\text{in}}(\gamma + 2\delta) < n_{\text{in}}(1-\varepsilon)$. By a simple averaging, at least $\left(1 - \frac{1-\varepsilon}{1-\varepsilon/4}\right)n_{\text{out}} \ge \frac{3\varepsilon}{4}\cdot n_{\text{out}}$ of those blocks have an error count of $n_{\text{in}}(1-\frac{\varepsilon}{4})$ or less. Let us call the set of all such blocks S.

Further, we partition S into smaller sets based on the number of deletions occurring in the blocks of S. Let $S_i \in S$ be the subset of blocks in S for which the number of deletions is in $\left[n_{\text{in}}\cdot\frac{\varepsilon}{16}\cdot(i-1), n_{\text{in}}\cdot\frac{\varepsilon}{16}\cdot i\right)$ for $i = 1, 2, \cdots, 8/\varepsilon$[1]. The following hold true:

(1) All blocks in S_i suffer from at least $n_{\text{in}}\cdot\frac{\varepsilon}{16}\cdot(i-1)$ deletions. Further, they can suffer from up to $n_{\text{in}}\cdot\left(1 - \frac{\varepsilon}{4} - \frac{2\varepsilon}{16}\cdot(i-1)\right)$ insertions. Therefore, they all appear as substrings of length $n_{\text{in}}\cdot\left(2 - \frac{\varepsilon}{4} - \frac{3\varepsilon}{16}\cdot(i-1)\right)$ or less in x'.

[1] Note that the fraction of deletions cannot exceed $\frac{1}{2}$ assuming $n_{\text{in}}(\gamma+2\delta) < n_{\text{in}}(1-\varepsilon)$.

(2) We have that $S = \dot{\bigcup}_{i=1}^{8/\varepsilon} S_i$. By the Pigeonhole principle, for some $i^* \in [1, 8/\varepsilon]$, $|S_{i^*}| \geq \frac{3\varepsilon^2}{32} n_{\text{out}}$.

Our decoding algorithm consists of $8/\varepsilon$ rounds each consisting of two phases of inner and outer decoding. During the first phase of each round $i = 1, 2, \cdots, 8/\varepsilon$, the algorithm uses the decoder of the inner code on x' to construct a string T_i over alphabet Σ_{out} and then, in the second phase, uses the decoder of the outer code on input T_i to obtain a list $List_i$ of size L_{out}. In the end, the decoding algorithm outputs the union of all such lists $\bigcup_i List_i$.

Algorithm 1 Decoder of the Concatenated Code

procedure CONCAT'D-DEC($x', \varepsilon, n_{\text{in}}, n_{\text{out}}, \text{Dec}_{C_{\text{in}}}, \text{Dec}_{C_{\text{out}}}$)
  Output $\leftarrow \emptyset$
  for $i \in \left\{1, 2, \cdots, \frac{8}{\varepsilon}\right\}$ **do** ▷ Round i
    $w \leftarrow \left\lfloor \frac{n_{\text{in}}(2-\varepsilon/4-3\varepsilon(i-1)/16)}{n_{\text{in}}\varepsilon/16} \right\rfloor + 1$
    $T_i \leftarrow$ empty string
    for $j \in \left\{1, 2, \cdots, \frac{|x'|}{n_{\text{in}}\varepsilon/16} - w\right\}$ **do** ▷ Phase I
      $List \leftarrow \text{Dec}_{C_{\text{in}}}\left(x'\left[\frac{n_{\text{in}}\varepsilon}{16} \cdot j, \frac{n_{\text{in}}\varepsilon}{16} \cdot (j+w)\right]\right)$
      Pad symbols of Σ_{out} corresponding to the elements of $List$ to the right of T_i.
    Output $\leftarrow$ Output $\cup\, \text{Dec}_{C_{\text{out}}}(T_i)$ ▷ Phase II
  return Output

Description of Phase I (Inner Decoding). We now proceed to the description of the first phase in each round $i \in \{1, 2, \cdots, 8/\varepsilon\}$. In the construction of T_i, we aim for correctly decoding the blocks in S_i. As mentioned above, all such blocks appear in x' in a substring of length $n_{\text{in}} \cdot \left(2 - \frac{\varepsilon}{4} - \frac{3\varepsilon}{16} \cdot (i-1)\right)$ or less.

Having this observation, we run the deocoder of the inner code on substrings of x' of form $x'\left[\frac{n_{\text{in}}\varepsilon}{16} \cdot j, \frac{n_{\text{in}}\varepsilon}{16} \cdot (j+w)\right]$ for all $j = 1, 2, \cdots, \frac{|x'|}{n_{\text{in}}\varepsilon/16} - w$ where $w = \left\lfloor \frac{n_{\text{in}}(2-\varepsilon/4-3\varepsilon(i-1)/16)}{n_{\text{in}}\varepsilon/16} \right\rfloor + 1$. One can think of such substrings as a *window* of size $w \cdot \frac{n_{\text{in}}\varepsilon}{16}$ that slides in $\frac{n_{\text{in}}\varepsilon}{16}$ increments.

Note that each block B in S_i appears within such window and is far from it by, say, D_B deletions and no more than $n_{\text{in}}\left(1 - \frac{\varepsilon}{4}\right) - 2D_B + \frac{n_{\text{in}}\varepsilon}{16}$ insertions where the additional $\frac{n_{\text{in}}\varepsilon}{16}$ term in insertion count comes from the extra symbols around the block in the fixed sized window. As long as the fraction of insertions plus twice the fraction of deletions that are needed to convert a block of S_i into its corresponding window does not exceed $1 - \varepsilon_{\text{in}}$, the output of the inner code's decoder for input $x'\left[\frac{n_{\text{in}}\varepsilon}{16} \cdot j, \frac{n_{\text{in}}\varepsilon}{16} \cdot (j+w)\right]$ will contain the block B of S_i. So, we choose ε_{in} such that

$$n_{\text{in}}\left(1 - \frac{\varepsilon}{4}\right) - 2D_B + \frac{n_{\text{in}}\varepsilon}{16} + 2D_B \leq n_{\text{in}}(1 - \varepsilon_{\text{in}}) \quad (11)$$

$$\Leftrightarrow \quad n_{\text{in}}(1 - 3\varepsilon/16) \leq n_{\text{in}}(1 - \varepsilon_{\text{in}}) \Leftrightarrow \varepsilon_{\text{in}} \leq \frac{3}{16}\varepsilon$$

Now, each element in the output list corresponds to some codeword of the inner code and, therefore, some symbol in Σ_{out}. For each run of the decoder of the inner code, we take the corresponding symbols of Σ_{out} and write them back-to-back in arbitrary order. Then, we append all such strings in the increasing order of j to obtain T_i.

Description of Phase II (Outer Decoding). Note that the length of T_i is at most $\frac{|x'|}{n_{\text{in}}\varepsilon/16} L_{\text{in}} \leq \frac{2n_{\text{in}}n_{\text{out}}}{n_{\text{in}}\varepsilon/16} L_{\text{in}} = n_{\text{out}} \cdot \frac{32}{\varepsilon} L_{\text{in}}$. Further, T_i contains symbols corresponding to all blocks of S_i as a subsequence (i.e., in the order of appearance) except possibly the ones that appear in the same run of the inner decoder together. Since the fraction of deletions happening to each block in S_i is less than $\frac{1}{2}$ and the size of the inner decoding sliding window is no more than $2n_{\text{in}}$, the number of blocks of S_i that can appear in the same window in the first phase is at most 4. This gives that T_i has a common subsequence of size at least $\frac{|S_i|}{4}$ with the codeword of the outer code.

We mentioned earlier that for some i^*, $|S_{i^*}| \geq \frac{3\varepsilon^2}{32} n_{\text{out}}$. Therefore, for such i^*, T_{i^*} is different from x by up to a $1 - \frac{3\varepsilon^2}{128}$ fraction of deletions and $\frac{32}{\varepsilon} L_{\text{in}}$ fraction of insertions. Therefore, by taking $\delta_{\text{out}} = 1 - \frac{3\varepsilon^2}{128}$, $\gamma_{\text{out}} = \frac{32}{\varepsilon} L_{\text{in}} = O\left(\frac{1}{\varepsilon^4}\right)$, and using each T_i as an input to the decoder of the outer code in the second phase, x will certainly appear in the outer output list for some T_i. (Specifically, for $i = i^*$.)

4.3 Remaining Parameters

As shown in Section 4.2, we need a list-decodable code as outer code that can list-decode from $\delta_{\text{out}} = 1 - \frac{3\varepsilon^2}{128}$ fraction of deletions and $\gamma_{\text{out}} = \frac{32}{\varepsilon} L_{\text{in}} = O\left(\frac{1}{\varepsilon^4}\right)$ fraction of insertions. To obtain such codes we use Theorem 2.1 with parameters $\gamma = \frac{32}{\varepsilon} L_{\text{in}}$ and $\epsilon = \frac{3\varepsilon^2}{256}$. This implies that the rate of the outer code is $r_{\text{out}} = \frac{3\varepsilon^2}{256} = O(\varepsilon^2)$, it is $L_{\text{out}} = O_\varepsilon(\exp(\exp(\exp(\log^* n))))$ list-decodable, and can be defined over an alphabet size of $|\Sigma_{\text{out}}| = e^{O\left(\frac{1}{\varepsilon^{10}} \log \frac{1}{\varepsilon^8}\right)}$.

Consequently, $|C_{in}| = \log|\Sigma_{\text{out}}| = O\left(\frac{1}{\varepsilon^{10}} \log \frac{1}{\varepsilon}\right)$. Note that in Theorem 1.4, the block length of the inner code can be chosen independently of its list size as the list size only depends on ε_{in}. This is a crucial quality in our construction since in our analysis ε_{in} and L_{in} are fixed first and then $|C_{\text{in}}|$ is chosen depending on the properties of the outer code.

As the decoder of the outer code is used $\frac{8}{\varepsilon}$ times in the decoding of the concatenated code, the list size of the concatenated code will be $L = \frac{8}{\varepsilon} \cdot L_{\text{out}} = O_\varepsilon(\exp(\exp(\exp(\log^* n))))$. The rate of the concatenated code is $r = r_{\text{out}} r_{\text{in}} = O\left(\varepsilon^2 \cdot \frac{\log\log|C_{\text{in}}|}{n_{\text{in}}}\right) = e^{-O\left(\frac{1}{\varepsilon^{10}} \log^2 \frac{1}{\varepsilon}\right)}$.

Finally, since the outer code is efficient and the inner code is explicit and can be decoded by brute-force in $O_\varepsilon(1)$ time, the encoding and decoding procedures run in polynomial time. This concludes the proof of Theorem 1.2.

5 EXTENSION TO LARGER ALPHABETS

In this section we extend the results presented so far to q-ary alphabets where $q > 2$.

5.1 Feasibility Region: Upper Bound

For an alphabet of size q, no positive-rate family of deletion codes can protect against $1 - \frac{1}{q}$ fraction of errors since, with that many deletions, an adversary can simply delete all but the most frequent

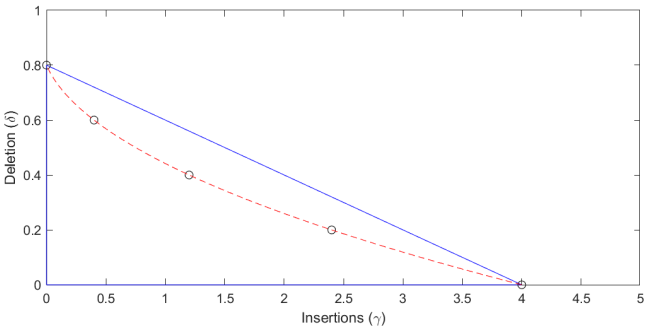

Figure 5: Infeasible points inside the conjectured feasibility region. (Illustrated for $q = 5$)

symbol of any codeword. Similarly, for insertion codes, it is not possible to achieve resilience against $q-1$ fraction of errors as adversary would be able to turn any codeword $x \in q^n$ to $(1, 2, \cdots, q)^n$.

The findings of the previous sections on binary alphabets might suggest that the feasibility region for list-decoding is the region mapped out by these two points, i.e., $\frac{\delta}{1-\frac{1}{q}} + \frac{\gamma}{q-1} < 1$. However, this conjecture turns out to be false. The following theorem provides a family of counterexamples.

Theorem 5.1. *For any alphabet size q and any $i = 1, 2, \cdots, q$, no positive-rate q-ary infinite family of insertion-deletion codes can list-decode from $\delta = \frac{q-i}{q}$ fraction of deletions and $\gamma = \frac{i(i-1)}{q}$ fraction of insertions.*

Proof. Take a codeword $x \in [q]^n$. With $\delta n = \frac{q-i}{q} \cdot n$ deletions, the adversary can delete the $q - i$ least frequent symbols to turn x into $x' \in \Sigma_d^{n(1-\delta)}$ for some $\Sigma_d = \{\sigma_1, \cdots, \sigma_i\} \subseteq [q]$. Then, with $\gamma n = n(1-\delta)(i-1) = n\frac{i(i-1)}{q}$ insertions, it can turn x' into $[\sigma_1, \sigma_2, \cdots, \sigma_i]^{n(1-\delta)}$. Such adversary only allows $O(1)$ amount of information to pass to the receiver. Hence, no such family of codes can yield a positive rate. □

Note that all points $(\gamma, \delta) = \left(\frac{i(i-1)}{q}, \frac{q-i}{q}\right)$ are located on a second degree curve inside the conjectured feasibility region $\frac{\delta}{1-\frac{1}{q}} + \frac{\gamma}{q-1} < 1$ (see Fig. 5). In the extended version of this paper, we use a simple time-sharing argument to show that the actual feasibility region is a subset of the polygon outlined by these points.

Theorem 5.2. *For any positive integer $q > 2$, define F_q as the concave polygon defined over vertices $\left(\frac{i(i-1)}{q}, \frac{q-i}{q}\right)$ for $i = 1, \cdots, q$ and $(0, 0)$. (see Fig. 1). F_q does not include the border except the two segments $[(0, 0), (q-1, 0))$ and $\left[(0, 0), \left(0, 1-\frac{1}{q}\right)\right)$. Then, for any pair of positive real numbers $(\gamma, \delta) \notin F_q$, there exists no infinite family of q-ary codes with positive rate that can correct from δ fraction of deletions and γ fraction of insertions.*

5.2 Feasibility Region: Exact Characterization

Finally, we will show that the feasibility region is indeed equal to the region F_q described in Theorem 5.2. The proof closely follows the steps taken for the binary case but is significantly more technical. We first formally define q-ary Bukh-Ma codes and show they are

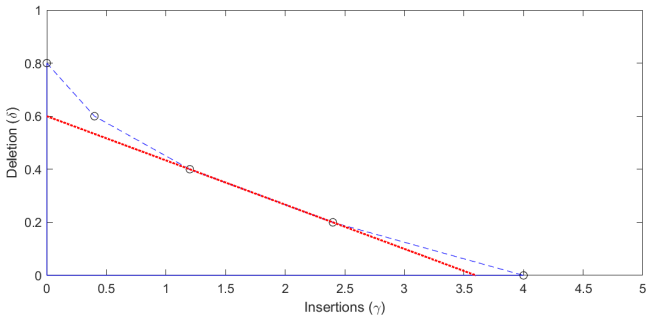

Figure 6: In the feasibility region for $q = 5$, the line passing through $(1.2, 0.4)$ and $(1.8, 0.3)$ (indicated with red dotted line) is characterized as $\gamma + 6\delta \leq 3.6$. (Corresponding to $i = 3$ in Eq. (12))

list-decodable as long as the error rate lies in F_q and then use the concatenation in Section 4 to obtain Theorem 1.3.

Theorem 5.3. *For any integer $q \geq 2$, $\varepsilon > 0$, and sufficiently large n, let $C^q_{n,\varepsilon}$ be the following Bukh-Ma code:*

$$C^q_{n,\varepsilon} = \left\{ \left(0^r 1^r \cdots q^r\right)^{\frac{n}{qr}} \middle| r = \left(\frac{1}{\varepsilon^4}\right)^k, k < \log_{1/\varepsilon^4} n \right\}.$$

For any $(\gamma, \delta) \in (1-\varepsilon)F_q$ it holds that $C^q_{n,\varepsilon}$ is list decodable from any δn deletions and γn insertions with a list size of $O\left(\frac{q^5}{\varepsilon^2}\right)$.

We remark that in the case of $q = 2$, Theorem 5.3 improves over Theorem 1.4 in terms of the dependence of the list size on ε.

5.2.1 Proof Sketch for Theorem 5.3. To prove Theorem 5.3, we show that Bukh-Ma codes are list-decodable as long as the error rate (γ, δ) lies beneath the line that connects a pair of consecutive non-zero vertices of F_q.

In other words, for pairs $\left(\frac{i(i-1)}{q}, \frac{q-i}{q}\right)$ and $\left(\frac{i(i+1)}{q}, \frac{q-i-1}{q}\right)$ we consider the line passing through them (see Fig. 6), i.e.,

$$\gamma + (2i)\delta = \frac{(2q-1)i - i^2}{q}, \quad i = 1, \cdots, q-1 \tag{12}$$

and show that as long as $\gamma + (2z)\delta \leq (1-\varepsilon)\frac{(2q-1)z - z^2}{q}$ for some $z \in \{1, \cdots, q-1\}$, Bukh-Ma codes are list-decodable. Note that the union of such areas is equal to $(1-\varepsilon)F_q$.

The analysis for each line follows the arguments for the binary case. Namely, we assume that k codewords can be converted to some center string v via (γ, δ) fraction of errors. Then, using an appropriate advantage notion and considering some coupled statistic processes obtained by sampling substrings, we show that k is bounded above by some $O_q(\text{poly}(1/\varepsilon))$.

The only major difference is that the notion of bias cannot be directly used for q-ary alphabets. In this general case, instead of keeping track of the variance of the bias, we keep track of the sum of the variances of the frequency of the occurrence of each symbol. We show that this quantity increases by some constant after each substring sampling (analogous to Lemma 3.3) by showing that a positive advantage requires that the frequency of occurrence of at least one of the symbols to be ε-different for two different values of the random variable T (analogous to Lemma 3.4). The rest of this

section contains more formal description of generalized notions and proofs for generalized q-ary claims.

5.3 Generalized Notation and Preliminaries

To prove Theorem 5.3, we need to generalize some of the notions and respective preliminary lemmas for the binary case. We start with defining ith order advantage.

Definition 5.4 (ith order q-ary advantage of matching M). For a pair of positive integers $i < q$, a pair of q-ary strings a and b, and a matching M between a and b, we define *ith order q-ary advantage of a to b* as follows: $\text{adv}_M^{q,i}(a,b) = \frac{(2i+1)|M|-|a|-\frac{i+i^2}{q}\cdot|b|}{|a|}$.

Note that the notion of advantage utilized for the binary case is obtained for $q = 2$ and $i = 1$ in the above definition. The notions of ith order advantage between two strings (that is independent of a specific matching, i.e., $\text{adv}^{q,i}(a,b)$) and infinite ith order advantage are defined in a similar manner to the binary case.

Remark 5.5. *In the same spirit as of the binary case, $adv_M^{q,i}(a,b)$ is simply the value of $|b|\left(\frac{(2q-1)i-i^2}{q} - (2i)\delta_M - \gamma_M\right)$ normalized by the length of a.*

Lemma 5.6. *If for strings a and b, $adv^{q,i}(a,b) \geq 0$, then $|a|$ and $|b|$ are within a q factor of each other.*

The proof of this lemma is similar to the binary case and can be found in the extended version of this argument.

Definition 5.7 (q-ary Alternating Strings). For any positive integer r, we define the infinite q-ary alternating string of run-length r as $A_r^q = (1^r 2^r \cdots q^r)^\infty$ and denote its prefix of length l by $A_{r,l}^q = A_r^q[1,l]$.

5.4 Proof of Theorem 5.3

As mentioned before, Theorem 5.3 can be restated as follows.

Theorem 5.8 (Restatement of Theorem 5.3). *For any integer $q \geq 2$, $\varepsilon > 0$, sufficiently large n, and any $z \in \{1,2,\cdots,q-1\}$, the Bukh-Ma code $C_{n,\varepsilon}^n$ from Theorem 5.3 is list decodable from any δn deletions and γn insertions with a list size $O\left(q^5/\varepsilon^2\right)$ as long as $\gamma + (2z)\delta \leq (1-\varepsilon)\frac{(2q-1)z-z^2}{q}$.*

To prove this restated version, once again, we follow the steps taken for the proof of Theorem 1.4 and assume for the sake of contradiction that there exists a string v and $k = \Omega\left(\frac{q^5}{\varepsilon^2}\right)$ members of $C_{n,\varepsilon}^q$ like $A_{r_1,n}^q, A_{r_2,n}^q, \cdots, A_{r_k,n}^q$, so that each $A_{r_i,n}^q$ can be converted to v with I_i insertions and D_i deletions where $I_i + (2z)D_i \leq (1-\varepsilon)\frac{(2q-1)z-z^2}{q}\cdot n$. We define the indices in a way that $r_1 > r_2 > \cdots > r_k$. Given the definition of $C_{n,\varepsilon}^q$, $r_i \geq \frac{r_{i+1}}{\varepsilon^4}$.

Given Remark 5.5 and Lemma 5.6, an argument similar to the one presented in Lemma 3.1 shows that for all these codewords, $\text{adv}^{q,z}(v, A_{r_i,n}^q) \geq \frac{\varepsilon}{q}$.

We define the following stochastic processes similar to the binary case. We split the string v into substrings of size $l_1 = r_1\varepsilon^2$, pick one uniformly at random and denote it by v_1. We define random variable $A_1 = \text{adv}^{q,z}(v_1, A_{r_1}^q)$ and random variables F_1^p for $p = 1,2,\cdots,q$ as the frequency of the occurrence of symbol p in v_1. In other words, $F_1^p = \frac{\text{count}_p(v_1)}{|v_1|}$. We continue this process for $j = 2,3,\cdots,k$ by splitting each v_{j-1} into substrings of length $l_j = r_j\varepsilon^2$, picking v_j uniformly at random, and defining $A_j = \text{adv}^{q,z}(v_j, A_{r_j}^q)$ and $F_j^p = \frac{\text{count}_p(v_j)}{|v_j|}$ for all $p \in \{1,2,\cdots,q\}$. We then define the sequence of real numbers $f_1, f_2, \cdots, f_k$ as follows: $f_i = \sum_{p=1}^q \text{Var}(F_i^p)$. This series of real numbers will play the role of $\text{Var}(B_i)$ in the binary case.

Lemma 5.9. *The following hold for $A_1,\cdots,A_k$ and $F_1^p,\cdots,F_k^p$ for all $p \in \{1,2,\cdots,q\}$: (1) $\mathbb{E}[F_i^p] = F_{i-1}^p$, and (2) $\mathbb{E}[A_i] \geq \frac{\varepsilon}{q}$.*

Proof. Since v_i is a substring of v_{i-1} chosen uniformly at random, the overall frequency of symbol p is equal to the average frequency of its occurrence in each substrings. The second item can be derived as in Lemma 3.2. □

The next lemma mimics Lemma 3.3 for the binary case.

Lemma 5.10. *For the sequence $f_1, f_2, \cdots, f_k$, we have that $f_{i+1} \geq f_i + \Omega(\varepsilon^2/q^4)$.*

Using Lemma 5.10, Theorem 5.8 can be simply proved as follows.

Proof of Theorem 5.8. Note that each f_i is the summation of the variance of q random variables that take values in $[0,1]$. Therefore, their value cannot exceed q. Since $f_{i+1} \geq f_i + \Omega(\varepsilon^2/q^4)$, the total length of the series, k, may not exceed $O\left(\frac{q^5}{\varepsilon^2}\right)$. This implies that the list size is $O\left(\frac{q^5}{\varepsilon^2}\right)$. □

We now present the proof of Lemma 5.10.

Proof of Lemma 5.10. To relate f_i and f_{i+1}, we utilize the law of total variance as follows:

$$\begin{aligned}\text{Var}(F_{i+1}^p) &= \text{Var}\left(\mathbb{E}[F_{i+1}^p|v_i]\right) + \mathbb{E}\left[\text{Var}(F_{i+1}^p|v_i)\right] \\ &= \text{Var}\left(F_i^p\right) + \mathbb{E}\left[\text{Var}(F_{i+1}^p|v_i)\right] \qquad (13)\end{aligned}$$

Equation (13) comes from the fact that the average frequency of symbol p in substrings of length l_{i+1} of v_i is equal to the frequency of p in v_i. Having this, we see that it suffices to show that $\mathbb{E}\left[\text{Var}(F_{i+1}^p|v_i)\right] \geq \Omega\left(\varepsilon^2/q^4\right)$. Similar to Lemma 3.3 we define E_j for $j = 1,2,\cdots,q$ and E_e respectively as the event that the projection of v_{i+1} falls inside a $j^{r_{i+1}}$ in $A_{r_{i+1}}$ or a string containing multiple symbols. We also define the random variable T out of $\{e,1,2,\cdots,q\}$ that indicates which one of these events is realized. Once again, we use the law of total variance to bound $\mathbb{E}\left[\text{Var}(F_{i+1}^p|v_i)\right]$.

$$\begin{aligned}\mathbb{E}\left[\text{Var}(F_{i+1}^p|v_i)\right] &= \mathbb{E}_{v_i}\left[\text{Var}_T\left(\mathbb{E}\left[F_{i+1}^p|v_i,T\right]\right)\right. \\ &\quad \left. + \mathbb{E}_T\left[\text{Var}(F_{i+1}^p|v_i,T)\right]\right] \\ &\geq \mathbb{E}_{v_i}\left[\text{Var}_T\left(\mathbb{E}\left[F_{i+1}^p|v_i,T\right]\right)\right] \qquad (14)\end{aligned}$$

Combining (13) and (14) gives

$$\mathrm{Var}(F_{i+1}^p) \geq \mathrm{Var}\left(F_i^p\right) + \mathbb{E}_{v_i}\left[\mathrm{Var}_T\left(\mathbb{E}\left[F_{i+1}^p|v_i,T\right]\right)\right]$$

$$\Rightarrow \sum_{p=1}^{q}\mathrm{Var}(F_{i+1}^p) \geq \sum_{p=1}^{q}\mathrm{Var}\left(F_i^p\right) + \sum_{p=1}^{q}\mathbb{E}_{v_i}\left[\mathrm{Var}_T\left(\mathbb{E}\left[F_{i+1}^p|v_i,T\right]\right)\right]$$

$$\Rightarrow f_{i+1} \geq f_i + \mathbb{E}_{v_i}\left[\sum_{p=1}^{q}\mathrm{Var}_T\left(\mathbb{E}\left[F_{i+1}^p|v_i,T\right]\right)\right] \tag{15}$$

Note that the term $\mathrm{Var}_T(\mathbb{E}[F_{i+1}^p|v_i,T])$ refers to the variance of a $(q+1)$-valued random variable that takes the value $\mathbb{E}_{v_i}[F_{i+1}^p|v_i,T=t]$ with probability $\Pr\{T=t|v_i\}$ for $t\in\{e,1,2,\cdots,q\}$. Once again, we present a crucial lemma that bounds from below the sum of variances of frequencies with respect to T assuming that the overall advantage is large enough.

Lemma 5.11. *For any realization of v_i, the following holds true if $adv^{q,z}(v_i,A_{r_{i+1}}) \geq 3q\varepsilon^2$:*

$$\sum_{p=1}^{q}\mathrm{Var}_T\left(\mathbb{E}\left[F_{i+1}^p|v_i,T\right]\right) \geq \left(\frac{adv^{q,z}(v_i,A_{r_{i+1}})-3q\varepsilon^2}{2z+1}\right)^2$$

We defer the proof of Lemma 5.11 to Section 5.6. Using Jensen inequality, the fact that $z\leq q$, and Lemma 5.11 along with (15) give that $f_{i+1}\geq f_i+\mathbb{E}_{v_i}\left[\left(\frac{\mathrm{adv}^{q,z}(v_i,A_{r_{i+1}})-3q\varepsilon^2}{2z+1}\right)^2\right] = f_i+\Omega\left(\frac{\varepsilon^2}{q^4}\right)$ for sufficiently small $\varepsilon>0$. □

5.5 Proof of Theorem 1.3

To establish Theorem 1.3, we closely follow the concatenation scheme presented in Section 4. In the following, we provide a high-level description of the proof skipping the details mentioned in Section 4 and highlighting the necessary extra steps.

The construction of the concatenated code is exactly as in Section 4 with the exception that the inner code is defined over an alphabet of size q. Note that if $(\gamma,\delta)\in(1-\varepsilon)F_q$, then (γ,δ) lies underneath one of the lines in the set of lines represented by (12). In other words, there exists some $z\in\{1,2,\cdots,q-1\}$ for which $\gamma+(2z)\delta\leq(1-\varepsilon)\left(\frac{(2q-1)z-z^2}{q}\right)$. Similar to Section 4, we define the notion of *error count* for each block in the codewords of the concatenated code as $(I+2z\cdot D)\cdot\frac{q}{(2q-1)z-z^2}$ where D and I denote the number of deletions and insertions occurred in the block respectively. As in Section 4 one can show that at least $\frac{3\varepsilon}{4}\cdot n_{\text{out}}$ of the blocks contain no more than $\left(1-\frac{\varepsilon}{4}\right)n_{\text{in}}$ error count. We denote the set of all such blocks by S. Once again, we partition S into subsets $S_1,S_2,\cdots$ depending on the number of deletions occurred in the set. More precisely, we define $S_i\subseteq S$ as the set of blocks in S that contain a number of deletions that is in the range $\left[n_{\text{in}}\cdot\frac{\varepsilon}{16q}\cdot(i-1),n_{\text{in}}\cdot\frac{\varepsilon}{16q}\cdot i\right)$ for $i=1,2,\cdots,16q/\varepsilon$. Once again, the following hold true:

(1) We have that $S=\dot{\bigcup}_{i=1}^{16q/\varepsilon}S_i$. By the Pigeonhole principle, for some $i^*\in[1,16q/\varepsilon]$, $|S_{i^*}|\geq\frac{3\varepsilon^2}{64q}n_{\text{out}}$.
(2) Take some $i\in\{1,2,\cdots,16q/\varepsilon\}$ and some block in S_i. Say D deletions have occurred in that block. Then, the total number of insertions is at most $(1-\varepsilon/4)\frac{(2q-1)z-z^2}{q}n_{\text{in}}-2zD$. Therefore, the total length of the block is $n_{\text{in}}-D(1-\varepsilon/4)\frac{(2q-1)z-z^2}{q}n_{\text{in}}-2zD$

$$= n_{\text{in}}\cdot\left[1+\left(1-\frac{\varepsilon}{4}\right)\frac{(2q-1)z-z^2}{q}\right]-(2z+1)D \tag{16}$$

which is no more than

$$n_{\text{in}}\cdot\left[1+\left(1-\frac{\varepsilon}{4}\right)\frac{(2q-1)z-z^2}{q}-\frac{\varepsilon}{16q}(i-1)(2z+1)\right] \tag{17}$$

Based on these observations, it is easy to verify that the decoding algorithm and analysis as presented in Section 4 and Algorithm 1 work for the q-ary case with the following minor modifications:

(a) Based on (17), the parameter w determining the length of the window should be

$$w=\left\lfloor\frac{n_{\text{in}}\cdot\left[1+\left(1-\frac{\varepsilon}{4}\right)\frac{(2q-1)z-z^2}{q}-\frac{\varepsilon}{16q}(i-1)(2z+1)\right]}{n_{\text{in}}\varepsilon/16}\right\rfloor+1. \tag{18}$$

(b) As in (11), parameter ε_{in} has to be chosen such that the error count in decoding windows does not exceed $n_{\text{in}}(1-\varepsilon_{\text{in}})$. Note that the choice of shifting steps for the decoding window from (18) may add up to $\frac{n_{\text{in}}\varepsilon}{16}$ additional insertions to the decoding window. Further, there is up to $n_{\text{in}}\frac{\varepsilon}{16q}$ uncertainty in the total length of the block from (16) since $D\in\left[n_{\text{in}}\cdot\frac{\varepsilon}{16q}\cdot(i-1),n_{\text{in}}\cdot\frac{\varepsilon}{16q}\cdot i\right)$. This can also add up to $n_{\text{in}}\frac{\varepsilon}{16q}(2z+1)\leq\frac{\varepsilon}{8}$ insertions. Therefore, we need $n_{\text{in}}(1-\varepsilon/4)+n_{\text{in}}\left(\frac{\varepsilon}{16}+\frac{\varepsilon}{8}\right)\cdot\frac{q}{(2q-1)z-z^2}\leq n_{\text{in}}(1-\varepsilon_{\text{in}})$. Note that $\frac{q}{(2q-1)z-z^2}\leq\frac{q}{2q-2}\leq 1$. Hence, it suffices that $1-\frac{\varepsilon}{4}+\frac{\varepsilon}{8}+\frac{\varepsilon}{16}\leq 1-\varepsilon_{\text{in}}$ or equivalently, $\varepsilon_{\text{in}}\leq\frac{\varepsilon}{16}$.

(c) Some modifications are necessary to the parameters of the outer code. Notably, for alphabet size q, $|S_{i^*}|\geq\frac{3\varepsilon^2}{64q}n_{\text{out}}$ and the fraction of deletions can be as high as $1-\frac{1}{q}$. This requires $\delta_{\text{out}}=1-\frac{3\varepsilon^2}{128q^2}$.

(d) Finally, note the the value of z is not know to the decoder. So the decoder has to run the algorithm with modifications mentioned above for all possible values of $z=1,2,\cdots,q-1$ and the output the union of all lists produced.

5.6 Proof of Lemma 5.11

We break down this proof into four steps. In the first step, similar to Lemma 3.4, we modify v_i and $A_{r_{i+1},n}$ into a simpler structure without significantly changing the advantage. In the second step, we provide an upper bound for the advantage in this modified version that depends on the local frequencies of symbols, more specifically, on what we refer to as $\mathbb{E}\left[F_{i+1}^j|v_i,T=j\right]$. In Step 3, we show that these upper-bounds would yield a non-positive value on the advantage if one replaces the local frequencies with the overall frequency of symbols in v_i, i.e., F_i^j. In the fourth and last step, we show that this means that the local frequencies have to significantly deviate from global ones to attain the advantage achieved by $\bar{M}_i$ (i.e., $\mathrm{adv}_{\bar{M}_i}^{q,z}$), so much that the lower-bound promised in the lemma's statement is achieved.

Step 1. Modifying v_i and $A_{r_{i+1},n}$ for the sake of simplicity: The proof starts with modifying v_i, $A_{r_{i+1},n}$, and the advantage-yielding matching M_i between them in a way that only slightly changes the value of advantage taking steps identical to the one in Lemma 3.4. Similar to Lemma 3.4, we denote the projection of v_i under M_i by $g = v_i \to M_i$. (See Fig. 3 for a depiction of the steps in binary case.)

(1) First, we delete all substrings of U_e–i.e., substrings of length l_{i+1} in v_i whose projection does not entirely fall into some stretch of $j^{r_{i+1}}$–from v_i.
(2) We reorder the substrings of length l_{i+1} in v_i by shifting all U_j substrings together and the projections in g to preserve the remainder of M_i from step 1.
(3) At this point, string g consists of a stretch of symbol 1 followed by a stretch of symbol 2, etc. If the length of all stretches are not equal, we add adequate symbols to each stretch to make g have the form of $1^t 2^t \cdots q^t$.

To track the changes in $\mathrm{adv}^{q,z}_{M_i}$ during this transformation, we track how $|M_i|$, $|v_i|$ and $|g|$ change throughout the three steps mentioned above.

In the first step, a total of up to $|U_e| l_{i+1}$ elements are removed from v_i and M_i. Note that since the run length of $A_{r_{i+1}}$ is r_{i+1}, there can only be $\frac{|g|}{r_{i+1}}$ substrings in U_e. Therefore, $|U_e| l_{i+1} \le \frac{|g| l_{i+1}}{r_{i+1}} = |g|\varepsilon^2 \le 2\varepsilon^2 |v_i|$.

The second step preserves $|M_i|$, $|v_i|$ and $|g|$.

Finally, since g is a substring of $A_{r_{i+1}}$, the third step increases $|g|$ only by up to $q r_{i+1}$. Note the run length of the $A_{r_{i+1}}$s and consequently l_{i+1}s are different by a multiplicative factor of at least $\frac{1}{\varepsilon^4}$ by the definition of the code C. Therefore, $q r_{i+1} = \frac{q l_{i+1}}{\varepsilon^2} = \frac{q l_{i+1} |v_i|}{\varepsilon^2 |v_i|} = \frac{q l_{i+1} |v_i|}{\varepsilon^2 l_i} \le \varepsilon^2 q |v_i|$.

Overall, the value of the $\mathrm{adv}^{q,z}_{M_i} = \frac{(2z+1)|M| - |v_i| - \frac{z+z^2}{q} \cdot |g|}{|v_i|}$ can be affected by a maximum of $2z \times 2\varepsilon^2 |v_i| + q\varepsilon^2 |v_i| = (2z+q)\varepsilon^2 |v_i| \le 3q\varepsilon^2 |v_i|$ decrease in the numerator and $\varepsilon^2 |v_i|$ decrease in the denominator. Therefore, the eventual advantage does not drop below $\mathrm{adv}^{q,z}_{M_i} - 3q\varepsilon^2$. Let us denote the transformed versions of v_i, g, and M_i by $\bar{v}_i$, $\bar{g}$, and $\bar{M}_i$ respectively. We have shown that

$$\mathrm{adv}^{q,z}_{\bar{M}_i} \ge \mathrm{adv}^{q,z}_{M_i} - 3q\varepsilon^2. \tag{19}$$

Step 2. Bounding Above $\mathrm{adv}^{q,z}_{\bar{M}_i}$ with f^*: Let $\bar{v}_i = (\bar{v}_i^1, \bar{v}_i^2, \cdots, \bar{v}_i^q)$ so that $\bar{v}_i^j$ corresponds to the part of $\bar{v}_i$ that is mapped to j^t under $\bar{M}_i$. Further, let $f^*_j = \mathbb{E}\left[F^j_{i+1} | v_i, T = j\right]$ represent the frequency of the occurrence of symbol j in $\bar{v}_i^j$ as a shorthand, i.e., $f^*_j = \frac{\mathrm{count}_j(\bar{v}_i^j)}{|\bar{v}_i^j|}$ and p_j be the relative length of $\bar{v}_i^j$, i.e., $p_j = \frac{|\bar{v}_i^j|}{|\bar{v}_i|}$. In this section, we compute an upperbound for $\mathrm{adv}^{q,z}_{\bar{M}_i}$ that depends on f^*_js. For the sake of simplicity, from now on we assume, without loss of generality, that $\mathrm{count}_1(\bar{v}_i^1) \ge \mathrm{count}_2(\bar{v}_i^2) \ge \cdots \ge \mathrm{count}_q(\bar{v}_i^q)$ or equivalently, $f^*_1 p_1 \ge f^*_2 p_2 \ge \cdots \ge f^*_q p_q$.

Consider the matching between $\bar{v}_i$ and $\bar{p}$ that, for any $j \in \{1, 2, \cdots, q\}$ matches as many js as possible from j^t to $\bar{v}_i^j$. This matching clearly yields the largest possible advantage between the two that is an upperbound for the $\mathrm{adv}^{q,z}_{\bar{M}_i}$. Similar to the binary case, we find a t that maximizes this advantage and use its advantage as an upper-bound for $\mathrm{adv}^{q,z}_{\bar{M}_i}$.

Let c be so that $f^*_c |\bar{v}_i^c| > t \ge f^*_{c+1} |\bar{v}_i^{c+1}|$. Then, increasing t by one would increase the length of $\bar{p}$ by q and increases the size of the matching by c. To see the effect of this increment on the advantage, note that the denominator does not change and the numerator changes by $c(2z+1) - \frac{z+z^2}{q} \cdot q$. This change in advantage is positive as long as $c(2z+1) - (z+z^2) \ge 0 \Leftrightarrow c \ge \frac{z+z^2}{2z+1} = \frac{z}{2} + \left(\frac{1}{4} - \frac{1}{4(2z+1)}\right)$. Note that the term $\frac{1}{4} - \frac{1}{4(2z+1)}$ is always between $\left[0, \frac{1}{4}\right]$. Hence, incrementing t increases the advantage as long as $c \ge \lfloor \frac{z}{2} \rfloor + 1$. This means that the highest possible advantage is derived when $t = f^*_w |\bar{v}_i^w|$ for $w = \lfloor \frac{z}{2} \rfloor + 1$. With this value for t, the matching contains $f^*_j |\bar{v}_i^j|$ edges between j^t and $|\bar{v}_i^j|$ for all $j > w$ and t edges between j^t and $|\bar{v}_i^j|$ for $j \le w$. Therefore, the size of this matching is $tw + \sum_{j=w+1}^{q} f^*_j |\bar{v}_i^j|$. This yields the following advantage

$$\begin{aligned}
&\frac{(2z+1)\left[tw + \sum_{j=w+1}^{q} f^*_j |\bar{v}_i^j|\right] - |\bar{v}_i| - \frac{z+z^2}{q} \cdot qt}{|\bar{v}_i|} \\
= \;& (2z+1)\left[f^*_w p_w w + \sum_{j=w+1}^{q} f^*_j p_j\right] - 1 - (z+z^2) \cdot f^*_w p_w \\
= \;& \left[(2z+1)w - (z+z^2)\right] \cdot f^*_w p_w + (2z+1) \sum_{j=w+1}^{q} f^*_j p_j - 1
\end{aligned}$$

We remind that this is an upper-bound on the $\mathrm{adv}^{q,z}_{\bar{M}_i}$. Next, we plug in $w = \lfloor \frac{z}{2} \rfloor + 1$ into this bound. Note that $(2z+1)w - (z+z^2) = z(2w - z) + w - z$ which is equal to $\frac{3z+2}{2}$ if z is even and $\frac{z+1}{2}$ if z is odd.

Therefore, we have the following set of upper-bounds on the advantage

$$\mathrm{adv}^{q,z}_{\bar{M}_i} \le \frac{3z+2}{2} \cdot f^*_w p_w + (2z+1) \sum_{j=w+1}^{q} f^*_j p_j - 1, \text{If } z \text{ is even} \tag{20}$$

$$\mathrm{adv}^{q,z}_{\bar{M}_i} \le \frac{z+1}{2} \cdot f^*_w p_w + (2z+1) \sum_{j=w+1}^{q} f^*_j p_j - 1, \text{If } z \text{ is odd} \tag{21}$$

Step 3. Proving Non-positivity of the Bound from Step 3 for Unit Sum Vectors: In this step, we show that the bounds (20) and (21) on advantage that were presented in Step 2 are necessarily non-positive for any vector $(f^*_1, \cdots, f^*_q)$ with unit sum including the vector of overall frequencies $\bar{f} = (\bar{f}_1, \cdots, \bar{f}_q)$ where $\bar{f}_j = \frac{\mathrm{count}_j(\bar{v}_i)}{|\bar{v}_i|} = F^j_i$. In Step 4, we use this fact to show that f^* needs to deviate noticeably from $\bar{f}$ which gives that the variance of frequencies with respect to T is large enough, thus finishing the proof.

Proposition 5.12. *Let $(p_1, \cdots, p_q)$ and $(f^*_1, \cdots, f^*_q)$ be two positive real vectors with unit sum that satisfy $f^*_1 p_1 \ge f^*_2 p_2 \ge \cdots \ge f^*_q p_q$. Then, for all integers $1 \le z < q$, the following hold for $w = \lfloor \frac{z}{2} \rfloor + 1$:*

(1) If z is even, $\frac{3z+2}{2} \cdot f^*_w p_w + (2z+1) \sum_{j=w+1}^{q} f^*_j p_j \le 1$.
(2) If z is odd, $\frac{z+1}{2} \cdot f^*_w p_w + (2z+1) \sum_{j=w+1}^{q} f^*_j p_j \le 1$.

The proof of Proposition 5.12 can be found in the extended version of this article.

Step 4. Large Deviation of f^*s from $\bar{f}$s and Large Variance: Here we finish the proof assuming z is odd. The even case can be proved in the same way. Note that Proposition 5.12 gives that for the overall frequency vector $\bar{f}$ which has a unit sum,

$$\frac{z+1}{2} \cdot \bar{f}_w p_w + (2z+1) \sum_{j=w+1}^{q} \bar{f}_j p_j - 1 \leq 0. \tag{22}$$

However, (19) and (21) imply that for local frequency vector f^*

$$\frac{z+1}{2} \cdot f^*_w p_w + (2z+1) \sum_{j=w+1}^{q} f^*_j p_j - 1 \geq \mathrm{adv}^{q,z}_{M_i} - 3q\varepsilon^2. \tag{23}$$

Subtracting (22) from (23) gives that

$$\frac{z+1}{2} \cdot p_w(f^*_w - \bar{f}_w) + (2z+1) \sum_{j=w+1}^{q} (f^*_j - \bar{f}_j)p_j \geq \mathrm{adv}^{q,z}_{M_i} - 3q\varepsilon^2.$$

$$\Rightarrow \frac{z+1}{2} \cdot p_w|f^*_w - \bar{f}_w| + (2z+1) \sum_{j=w+1}^{q} |f^*_j - \bar{f}_j|p_j \geq \mathrm{adv}^{q,z}_{M_i} - 3q\varepsilon^2.$$

$$\Rightarrow (2z+1) \sum_{j=w}^{q} |f^*_j - \bar{f}_j|p_j \geq \mathrm{adv}^{q,z}_{M_i} - 3q\varepsilon^2.$$

$$\Rightarrow \sum_{j=w}^{q} |f^*_j - \bar{f}_j|p_j \geq \frac{\mathrm{adv}^{q,z}_{M_i} - 3q\varepsilon^2}{2z+1}.$$

This means that there exists some j_0 for which $|f^*_{j_0} - \bar{f}_{j_0}|p_{j_0} \geq \frac{\mathrm{adv}^{q,z}_{M_i} - 3q\varepsilon^2}{2z+1} \Rightarrow (f^*_{j_0} - \bar{f}_{j_0})^2 p_{j_0} \geq (f^*_{j_0} - \bar{f}_{j_0})^2 p^2_{j_0} \geq \left(\frac{\mathrm{adv}^{q,z}_{M_i} - 3q\varepsilon^2}{2z+1}\right)^2$.

Note that

$$\begin{aligned}
&\sum_{p=1}^{q} \mathrm{Var}_T\left(\mathbb{E}\left[F^p_{i+1}|v_i, T\right]\right) \\
&\quad = \sum_{p=1}^{q}\sum_{j=1}^{q} \left(\mathbb{E}\left[F^p_{i+1}|v_i, T=j\right] - F^p_i\right)^2 \Pr\{T=j|v_i\} \\
&\quad \geq \left(\mathbb{E}\left[F^{j_0}_{i+1}|v_i, T=j_0\right] - F^{j_0}_i\right)^2 \Pr\{T=j_0|v_i\} \\
&\quad = (f^*_{j_0} - \bar{f}_{j_0})^2 p_{j_0} \geq \left(\frac{\mathrm{adv}^{q,z}_{M_i} - 3q\varepsilon^2}{2z+1}\right)^2.
\end{aligned}$$

□

REFERENCES

[1] Erdal Arikan. 2008. Channel polarization: A method for constructing capacity-achieving codes. In *2008 IEEE International Symposium on Information Theory*. IEEE, 1173–1177.

[2] Jaroslaw Blasiok, Venkatesan Guruswami, Preetum Nakkiran, Atri Rudra, and Madhu Sudan. 2018. General strong polarization. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*. 485–492.

[3] Joshua Brakensiek, Venkatesan Guruswami, and Samuel Zbarsky. 2018. Efficient Low-Redundancy Codes for Correcting Multiple Deletions. *IEEE Trans. Information Theory* 64, 5 (2018), 3403–3410.

[4] Boris Bukh and Venkatesan Guruswami. 2016. An improved bound on the fraction of correctable deletions. In *Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. 1893–1901.

[5] Boris Bukh, Venkatesan Guruswami, and Johan Håstad. 2017. An improved bound on the fraction of correctable deletions. *IEEE Transactions on Information Theory* 63, 1 (2017), 93–103.

[6] Boris Bukh and Jie Ma. 2014. Longest common subsequences in sets of words. *SIAM Journal on Discrete Mathematics* 28, 4 (2014), 2042–2049.

[7] Kuan Cheng, Bernhard Haeupler, Xin Li, Amirbehshad Shahrasbi, and Ke Wu. 2019. Synchronization strings: highly efficient deterministic constructions over small alphabets. In *Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*.

[8] Kuan Cheng, Zhengzhong Jin, Xin Li, and Ke Wu. 2018. Deterministic document exchange protocols, and almost optimal binary codes for edit errors. In *Proceedings of the Annual Symposium on Foundations of Computer Science (FOCS)*.

[9] Kuan Cheng, Zhengzhong Jin, Xin Li, and Ke Wu. 2019. Block Edit Errors with Transpositions: Deterministic Document Exchange Protocols and Almost Optimal Binary Codes. In *International Colloquium on Automata, Languages, and Programming (ICALP)*.

[10] Vladimír Dančík. 1994. *Expected length of longest common subsequences*. Ph.D. Dissertation. University of Warwick.

[11] Vlado Dančík and Mike Paterson. 1995. Upper bounds for the expected length of a longest common subsequence of two binary sequences. *Random Structures & Algorithms* 6, 4 (1995), 449–458.

[12] Venkatesan Guruswami and Ray Li. 2016. Efficiently decodable insertion/deletion codes for high-noise and high-rate regimes. In *Information Theory (ISIT), 2016 IEEE International Symposium on*. IEEE, 620–624.

[13] Venkatesan Guruswami and Ray Li. 2018. Coding against deletions in oblivious and online models. In *Proceedings of the 29th Annual ACM-SIAM Symposium on Discrete Algorithms*. 625–643.

[14] Venkatesan Guruswami and Carol Wang. 2017. Deletion codes in the high-noise and high-rate regimes. *IEEE Transactions on Information Theory* 63, 4 (2017), 1961–1970.

[15] Venkatesan Guruswami and Chaoping Xing. 2013. List decoding Reed-Solomon, Algebraic-Geometric, and Gabidulin subcodes up to the Singleton bound. In *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*. ACM, 843–852.

[16] Bernhard Haeupler. 2019. Optimal document exchange and new codes for insertions and deletions. In *Proceedings of the Annual Symposium on Foundations of Computer Science (FOCS)*. 334–347.

[17] Bernhard Haeupler, Aviad Rubinstein, and Amirbehshad Shahrasbi. 2019. Near-linear time insertion-deletion codes and $(1+\varepsilon)$-approximating edit distance via indexing. In *Proceedings of the Annual Symposium on Theory of Computing (STOC)*. 697–708.

[18] Bernhard Haeupler and Amirbehshad Shahrasbi. 2017. Synchronization Strings: Codes for Insertions and Deletions Approaching the Singleton Bound. In *Proceedings of the Annual Symposium on Theory of Computing (STOC)*. 33–46.

[19] Bernhard Haeupler and Amirbehshad Shahrasbi. 2018. Synchronization Strings: Explicit Constructions, Local Decoding, and Applications. In *Proceedings of the Annual Symposium on Theory of Computing (STOC)*. 841–854.

[20] Bernhard Haeupler, Amirbehshad Shahrasbi, and Madhu Sudan. 2018. Synchronization Strings: List Decoding for Insertions and Deletions. In *45th International Colloquium on Automata, Languages, and Programming (ICALP)*. 76:1–76:14.

[21] Bernhard Haeupler, Amirbehshad Shahrasbi, and Ellen Vitercik. 2018. Synchronization Strings: Channel Simulations and Interactive Coding for Insertions and Deletions. In *45th International Colloquium on Automata, Languages, and Programming (ICALP)*. 75:1–75:14.

[22] Tomohiro Hayashi and Kenji Yasunaga. 2018. On the List Decodability of Insertions and Deletions. In *2018 IEEE International Symposium on Information Theory (ISIT)*. IEEE, 86–90.

[23] Marcos Kiwi, Martin Loebl, and Jiří Matoušek. 2005. Expected length of the longest common subsequence for large alphabets. *Advances in Mathematics* 197, 2 (2005), 480–498.

[24] Vladimir Levenshtein. 1965. Binary codes capable of correcting deletions, insertions, and reversals. *Doklady Akademii Nauk SSSR 163* 4 (1965), 845–848.

[25] Shu Liu, Ivan Tjuawinata, and Chaoping Xing. 2019. Explicit Constructions of Two-Dimensional Reed-Solomon Codes in High Insertion and Deletion Noise Regime. *arXiv preprint arXiv:1909.03426* (2019).

[26] Shu Liu, Ivan Tjuawinata, and Chaoping Xing. 2019. List Decoding of Insertion and Deletion Codes. *arXiv preprint arXiv:1906.09705* (2019).

[27] George S Lueker. 2009. Improved bounds on the average length of longest common subsequences. *Journal of the ACM (JACM)* 56, 3 (2009), 17.

[28] Hugues Mercier, Vijay K Bhargava, and Vahid Tarokh. 2010. A survey of error-correcting codes for channels with symbol synchronization errors. *IEEE Communications Surveys & Tutorials* 12, 1 (2010).

[29] Michael Mitzenmacher. 2009. A survey of results for deletion channels and related synchronization channels. *Probability Surveys* 6 (2009), 1–33.

[30] Leonard J. Schulman and David Zuckerman. 1999. Asymptotically good codes correcting insertions, deletions, and transpositions. *IEEE transactions on information theory* 45, 7 (1999), 2552–2557.

[31] Neil J. A Sloane. 2002. On single-deletion-correcting codes. *Codes and designs* 10 (2002), 273–291.

[32] Antonia Wachter-Zeh. 2018. List Decoding of Insertions and Deletions. *IEEE Trans. Information Theory* 64, 9 (2018), 6297–6304.