arXiv:2206.03097v1 [cs.DS] 7 Jun 2022

Locality-sensitive bucketing functions

Ke Chen[1] and Mingfu Shao[1,2]

[1]Department of Computer Science and Engineering, School of Electronic Engineering and Computer Science, The Pennsylvania State University, United States
[2]Huck Institutes of the Life Sciences, The Pennsylvania State University, United States

Abstract

Many bioinformatics applications involve bucketing a set of sequences where each sequence is allowed to be assigned into multiple buckets. To achieve both high sensitivity and precision, bucketing methods are desired to assign similar sequences into the same buckets while assigning dissimilar sequences into distinct buckets. Existing k-mer-based bucketing methods have been efficient in processing sequencing data with low-error rate, but encounter much reduced sensitivity on data with high-error rate. Locality-sensitive hashing (LSH) schemes are able to mitigate this issue through tolerating the edits in similar sequences, but state-of-the-art methods still have large gap. Here we generalize the LSH function by allowing it to hash one sequence into multiple buckets. Formally, a bucketing function, which maps a sequence (of fixed length) into a subset of buckets, is defined to be (d_1, d_2)-sensitive if any two sequences within an edit distance of d_1 are mapped into at least one shared bucket, and any two sequences with distance at least d_2 are mapped into disjoint subsets of buckets. We construct LSB functions with a variety of values of (d_1, d_2) and analyze their efficiency with respect to the total number of buckets needed as well as the number of buckets that a specific sequence is mapped to. We also prove lower bounds of these two parameters in different settings and show that some of our constructed LSB functions are optimal. These results provide theoretical foundations for their practical use in analyzing sequencing data with high error rate while also providing insights for the hardness of designing ungapped LSH functions.

1 Introduction

Comparing a set of given sequences is a common task involved in many bioinformatics applications, such as homology detection [5], overlap detection and the construction of overlap graphs [9, 4, 20], phylogeny tree reconstruction, and isoform detection from circular consensus sequence (CCS) reads [18], to name a few. The naive all-vs-all comparison gives the most comprehensive information but does not scale well. An efficient and widely-used approach that avoids all-vs-all comparison is *bucketing*: a linear scan is employed to assign each sequence into one or multiple buckets, followed by all-vs-all comparison within each bucket. The procedure of assigning sequences into buckets, which we refer to as a *bucketing function*, is desired to be both "sensitive", i.e., two similar sequences ideally appear in at least one shared bucket so that they can be compared, and "specific", i.e., two dissimilar sequences ideally appear in disjoint buckets so that they can be exempt from comparison. The criteria of similar/dissimilar sequences are application-dependent; in this work we study bucketing functions for the edit distance (Levenshtein distance).

A simple yet popular bucketing function is to put a sequence into buckets labeled with its own k-mers. The popular seed-and-extend strategy [1, 2] implicitly uses this approach. Various sketching methods such as minimizers [16, 19, 17, 10] and universal hitting set [13, 6] reduce the number of buckets a sequence is assigned to by only considering a subset of representative k-mers. These bucketing methods based on exact k-mer matching gain tremendous success in analyzing next-generation sequencing (NGS) data, but are challenged by the third-generation sequencing data represented by PacBio [15] and Oxford Nanopore [7] technologies. Due to the high-error rate, sequences that should be assigned to the same

This work is supported by the US National Science Foundation (DBI-2019797 to M.S.) and the US National Institutes of Health (R01HG011065 to M.S.).

buckets hardly share any identical k-mers (for a reasonably large k such as $k = 21$ with 15% error rate), and therefore results in poor sensitivity.

To address this issue, it is required to be able to recognize similar sequences with edits. A solution is locality-sensitive hashing [11, 12] where with high probability, similar (but not necessarily identical) sequences are sent into the same bucket (i.e., when there is a hash-collision), and with high probability dissimilar sequences are sent into different buckets. However, the design of locality-sensitive hashing functions for edit distance is hard; the state-of-the-art method Order Min Hash (OMH) is proved to be a gapped LSH but admits a large gap [11]. Another related approach is embedding the metric space with edit distance into more well-studied normed spaces [3, 14, 20]. However, such an embedding is also hard; for example, it is known that the embedding into L_1 cannot be distortion-free [8].

It is worth noting that locality-sensitive hashing functions, when interpreted as bucketing functions, assign a sequence into exactly one bucket: buckets are labeled with hash values, and a sequence is put into the single bucket where it is hashed to. In this work, we propose the concept of *locality-sensitive bucketing* (LSB) functions as a generalization of LSH functions by allowing it to assign a sequence into multiple buckets. Formally, a bucketing function, which maps a sequence (of fixed length) into one or more buckets, is defined to be (d_1, d_2)-sensitive if any two sequences within an edit distance of d_1 are mapped into at least one shared bucket, and any two sequences with an edit distance at least d_2 are mapped into disjoint subsets of buckets. While a stochastic definition by introducing a distribution on a family of bucketing functions can be made in a similar way as the definition of LSH functions, here we focus on this basic, deterministic definition. We design several LSB functions for a variety of values of (d_1, d_2) including both ungapped ($d_2 = d_1 + 1$) and gapped ($d_2 > d_1 + 1$) ones. This verifies that allowing one sequence to appear in multiple buckets makes the locality-sensitive properties easier to satisfy. Moreover, our lower bound proof shows that any $(1, 2)$-sensitive bucketing function must put each sequence (of length n) into at least n buckets (see Lemma 2), suggesting that certain ungapped locality-sensitive hashing functions, where each sequence is sent to a single bucket, may not exist.

The rest of this paper is organized as follows. In Section 2, we give the precise definition of LSB functions and propose criteria to measure them. In Sections 3 and 4, we design LSB functions using two different approaches, summarized in Section 5. We show experimental studies in Section 6, with a focus on demonstrating the performance of gapped LSB functions.

2 Basics of locality-sensitive bucketing (LSB) functions

Given an alphabet Σ with $|\Sigma| > 1$ and $k \in \mathbb{N}$, the k-mer space $\mathcal{S}_k = \Sigma^k$ is a metric space of all length k strings equipped with the Levenshtein (edit) distance. Given a set B of buckets, a bucketing function f maps $\mathcal{S}_k$ to $\mathcal{P}(B)$, the power set of B. This can be viewed as assigning a k-mer $\boldsymbol{s}$ to a subset of buckets $f(\boldsymbol{s}) \subset B$. Let $d_1 < d_2$ be two non-negative integers, we say a bucketing function f is (d_1, d_2)-*sensitive* if

$$\operatorname{edit}(\boldsymbol{s}, \boldsymbol{t}) \leq d_1 \implies f(\boldsymbol{s}) \cap f(\boldsymbol{t}) \neq \varnothing, \tag{1}$$
$$\operatorname{edit}(\boldsymbol{s}, \boldsymbol{t}) \geq d_2 \implies f(\boldsymbol{s}) \cap f(\boldsymbol{t}) = \varnothing. \tag{2}$$

We refer to above two conditions as LSB-properties (1) and (2) respectively. Intuitively, the LSB-properties state that, if two k-mers are within an edit distance of d_1, then the bucketing function f guarantees assigning them to at least one same bucket, and if two k-mers have an edit distance at least d_2, then the bucketing function f guarantees not assigning them to the same bucket. In other words, (d_1, d_2)-sensitive bucketing functions perfectly distinguish k-mers within distance d_1 from those with distance at least d_2. It is easy to show that if $f : \mathcal{S}_k \to \mathcal{P}(B)$ is a (d_1, d_2)-sensitive bucketing function, then $f(\boldsymbol{s}) \neq \varnothing$ for all $\boldsymbol{s} \in \mathcal{S}_k$. In fact, since $\operatorname{edit}(\boldsymbol{s}, \boldsymbol{s}) = 0 \leq d_1$, the LSB-property (1) implies that $f(\boldsymbol{s}) = f(\boldsymbol{s}) \cap f(\boldsymbol{s}) \neq \varnothing$. If $d_1 = d_2 - 1$ then we say the bucketing function is ungapped; otherwise it is called gapped.

We note that above definition of LSB functions generalize the (deterministic) locality-sensitive hashing functions: if we require that $|f(\boldsymbol{s})| = 1$ for every k-mer $\boldsymbol{s} \in \mathcal{S}_k$, i.e., f maps a k-mer into an element instead of a set, then $f(\boldsymbol{s}) \cap f(\boldsymbol{t}) \neq \varnothing$ becomes $f(\boldsymbol{s}) = f(\boldsymbol{t})$ and $f(\boldsymbol{s}) \cap f(\boldsymbol{t}) = \varnothing$ becomes $f(\boldsymbol{s}) \neq f(\boldsymbol{t})$.

Two related parameters can be used to measure a locality-sensitive bucketing function: $|B|$, the total number of buckets, and $|f(\boldsymbol{s})|$, the number of different buckets that contain a specific k-mer $\boldsymbol{s}$. From a practical perspective, it is desirable to keep both parameters small. We therefore aim to design LSB functions with small $|B|$ and $|f(\boldsymbol{s})|$. Specifically, in the following sections, we will construct (d_1, d_2)-sensitive bucketing functions with a variety of values of (d_1, d_2), analyze their corresponding $|B|$ and $|f(\boldsymbol{s})|$; we will also prove the bounds of $|B|$ and $|f(\boldsymbol{s})|$ in different settings and show that some of our constructed LSB functions are optimal, in terms of minimizing $|B|$ and $|f(\boldsymbol{s})|$.

The bounds of $|B|$ and $|f(\boldsymbol{s})|$ are closely related to the structure of k-mer space $\mathcal{S}_k$. For a k-mer $\boldsymbol{s} \in \mathcal{S}_k$, its d-neighborhood, denoted by $N_k^d(\boldsymbol{s})$, is the set of all k-mers with edit distance at most d from $\boldsymbol{s}$; formally $N_k^d(\boldsymbol{s}) = \{\boldsymbol{t} \in \mathcal{S}_k \mid \text{edit}(\boldsymbol{s}, \boldsymbol{t}) \leq d\}$. The following simple fact demonstrates the connection between the bounds of $|B|$ and $|f(\boldsymbol{s})|$ and the structure of $\mathcal{S}_k$, which will be used later.

Lemma 1. *If there exists β k-mers in $N_k^{d_1}(\boldsymbol{s})$ such that all the pairwise edit distances are at least d_2, then for any (d_1, d_2)-sensitive bucketing function f we must have $|f(\boldsymbol{s})| \geq \beta$.*

Proof. Let f be an arbitrary (d_1, d_2)-sensitive bucketing function. By the LSB-property (2), these β k-mers must be assigned to distinct buckets by f. On the other hand, since they are all in $N_k^{d_1}(\boldsymbol{s})$, the LSB-property (1) requires that $f(\boldsymbol{s})$ overlaps with $f(\boldsymbol{t})$ for each k-mer t in them. Combined, we have $|f(\boldsymbol{s})| \geq \beta$. $\square$

3 An optimal $(1, 2)$-sensitive bucketing function

In the most general setting of LSB functions, the labels of buckets in B are just symbols that are irrelevant to the construction of the bucketing function. Hence we can let $B = \{1, \ldots, |B|\}$. The remaining of this section studies $(1, 2)$-sensitive bucketing functions on this general case. We first prove lower bounds of $|B|$ and $|f(\boldsymbol{s})|$ in this case; we then give an algorithm to construct an optimal $(1, 2)$-sensitive bucketing function f that matches these bounds.

Lemma 2. *If $f : \mathcal{S}_k \to \mathcal{P}(B)$ is $(1, 2)$-sensitive, then for each $\boldsymbol{s} \in \mathcal{S}_k$, $|f(\boldsymbol{s})| \geq k$.*

Proof. Applying Lemma 1 with $d_1 = 1$ and $d_2 = 2$, we only need to show that $N_k^1(\boldsymbol{s})$ contains a set of k different k-mers with pairwise edit distance at least 2. For $i = 1, \ldots, k$, let $\boldsymbol{t}^i$ be a k-mer obtained from $\boldsymbol{s}$ by a single substitution at position i. Observe that if $i \neq j$, then $\boldsymbol{t}^i$ differs from $\boldsymbol{t}^j$ at two positions, namely i and j. So $\{\boldsymbol{t}^1, \ldots, \boldsymbol{t}^k\}$ forms the required set. $\square$

Lemma 3. *If $f : \mathcal{S}_k \to \mathcal{P}(B)$ is $(1, 2)$-sensitive, then $|B| \geq k|\Sigma|^{k-1}$.*

Proof. Consider the collection of pairs $H = \{(\boldsymbol{s}, b) \mid \boldsymbol{s} \in \mathcal{S}_k \text{ and } b \in f(\boldsymbol{s})\}$. We bound the size of H from above and below. For an arbitrary k-mer $\boldsymbol{s}$, let $b \in f(\boldsymbol{s})$ be a bucket that contains $\boldsymbol{s}$. According to the LSB-property (2), any other k-mer in b has edit distance 1 from $\boldsymbol{s}$, i.e., a substitution. Suppose that b contains two k-mers $\boldsymbol{u}$ and $\boldsymbol{v}$ that are obtained from $\boldsymbol{s}$ by a single substitution at different positions. Then $\text{edit}(\boldsymbol{u}, \boldsymbol{v}) = 2$ and $f(\boldsymbol{u}) \cap f(\boldsymbol{v}) \neq \varnothing$, which contradicts the LSB-property (2). Therefore, all the k-mers in b must be identical to $\boldsymbol{s}$ except at some fixed position i. There are $|\Sigma|$ such k-mers (including $\boldsymbol{s}$ itself). So each $b \in B$ can appear in at most $|\Sigma|$ pairs in H. Thus $|H| \leq |\Sigma| \cdot |B|$.

On the other hand, for a k-mer $\boldsymbol{s}$, its 1-neighborhood $N_k^1(s)$ contains $k(|\Sigma| - 1)$ other k-mers, corresponding to the $|\Sigma| - 1$ possible substitutions at each of the k positions. The LSB-property (1) requires that $\boldsymbol{s}$ shares at least one bucket with each of them. As argued above, each bucket $b \in f(\boldsymbol{s})$ can contain at most $(|\Sigma| - 1)$ k-mers other than $\boldsymbol{s}$. Therefore, $\boldsymbol{s}$ needs to appear in at least $k(|\Sigma| - 1)/(|\Sigma| - 1) = k$ different buckets, and hence at least k pairs in H. So $|H| \geq k|\mathcal{S}_k| = k|\Sigma|^k$. Together we have $|\Sigma| \cdot |B| \geq k|\Sigma|^k$, or $|B| \geq k|\Sigma|^{k-1}$. $\square$

We now construct a bucketing function $f : \mathcal{S}_k \to \mathcal{P}(B)$ that is $(1, 2)$-sensitive using an algorithm given below. It runs in exponential time but primarily serves as an constructive proof that $(1, 2)$-sensitive bucketing function exists. Assign to the alphabet Σ an arbitrary order $\sigma : \{1, \ldots, |\Sigma|\} \to \Sigma$.

This induces a lexicographical order π_σ on $\mathcal{S}_k$. The following algorithm defines the function f:

```
foreach s in S_k do f(s) = ∅
m ← 1   // index of the smallest unused bucket
for n = 1 to |Σ|^k do
    s ← π_σ(n)   // s = s_1 s_2 ··· s_k
    for i = 1 to k do
        if s_i == σ(1) then
            for j = 1 to |Σ| do
                t ← s_1 ··· s_{i−1} σ(j) s_{i+1} ··· s_k
                f(t) ← f(t) ∪ {m}   // add t to bucket m
            end
            m ← m + 1
        end
    end
end
```

Lemma 4. *The constructed bucketing function $f : \mathcal{S}_k \to \mathcal{P}(B)$ satisfies: (i) each bucket contains $|\Sigma|$ k-mers, (ii) $|f(\boldsymbol{s})| = k$ for each $\boldsymbol{s} \in \mathcal{S}_k$, and (iii) $|B| = k|\Sigma|^{k-1}$.*

Proof. Claim (i) follows directly from the construction. In the algorithm, each k-mer $\boldsymbol{s}$ is added to k different buckets, one for each position. Namely, for $i = 1, \dots, k$, there is a bucket m_i that contains $\boldsymbol{s}$ together with all the k-mers obtained from $\boldsymbol{s}$ by a substitution at position i. So $|f(\boldsymbol{s})| = k$. To calculate $|B|$, observe that a new bucket is used whenever we encounter the smallest symbol $\sigma(1)$ in some k-mer $\boldsymbol{s}$. So $|B|$ is the same as the number of occurrences of $\sigma(1)$ among all k-mers in $\mathcal{S}_k$. The total number of symbols in $\mathcal{S}_k$ is $k|\Sigma|^k$. By symmetry, $\sigma(1)$ appears $k|\Sigma|^{k-1}$ times. □

Lemma 5. *The constructed bucketing function f is $(1,2)$-sensitive.*

Proof. We show that for $\boldsymbol{s}, \boldsymbol{t} \in \mathcal{S}_k$, $\mathrm{edit}\,(\boldsymbol{s}, \boldsymbol{t}) \le 1$ if and only if $f(\boldsymbol{s}) \cap f(\boldsymbol{t}) \neq \varnothing$. For the forward direction, $\boldsymbol{s}$ and $\boldsymbol{t}$ can differ by at most one substitution at some position i. Let $\boldsymbol{r}$ be the k-mer that is identical to $\boldsymbol{s}$ except at the i-th position where it is substituted by the smallest character $\sigma(1)$. When processing $\boldsymbol{r}$, both $\boldsymbol{s}$ and $\boldsymbol{t}$ are added to a same bucket m. Therefore, $m \in f(\boldsymbol{s}) \cap f(\boldsymbol{t})$.

For the backward direction, let m be an integer from $f(\boldsymbol{s}) \cap f(\boldsymbol{t})$. Note that all the $|\Sigma|$ k-mers in bucket m differ by a single substitution. Hence, $\mathrm{edit}\,(\boldsymbol{s}, \boldsymbol{t}) \le 1$. □

Combining Lemmas 2–5, we have shown that the above $(1,2)$-sensitive bucketing function is optimal in the sense of minimizing $|B|$ and $|f(\boldsymbol{s})|$. This is summarized below.

Theorem 1. *Let $n = k|\Sigma|^{k-1}$ and $B = \{1, \dots, n\}$, there is a $(1,2)$-sensitive bucketing function $f : \mathcal{S}_k \to \mathcal{P}(B)$ with $|f(\boldsymbol{s})| = k$ for each $\boldsymbol{s} \in \mathcal{S}_k$. No $(1,2)$-sensitive bucketing function exists if $|B|$ is smaller or $|f(\boldsymbol{s})| \ge k$ does not hold for all k-mers.*

4 Mapping to k-mers

We continue to explore LSB functions with different values of d_1 and d_2. Here we focus on a special case where $B \subset \mathcal{S}_k$, namely, each bucket in B is labeled by a k-mer. Then the idea of designing LSB functions is to map a k-mer $\boldsymbol{s}$ into its neighboring k-mers that are in B. Formally, given a subset $B \subset \mathcal{S}_k$ and an integer $r \ge 1$, we define the bucketing function $f_r^B : \mathcal{S}_k \to \mathcal{P}(B)$ by

$$f_r^B(\boldsymbol{s}) = N_k^r(\boldsymbol{s}) \cap B = \{\boldsymbol{v} \in B \mid \mathrm{edit}\,(\boldsymbol{s}, \boldsymbol{v}) \le r\} \text{ for each } \boldsymbol{s} \in \mathcal{S}_k.$$

We now derive the conditions for f_r^B to be an LSB function. Since f_r^B maps a k-mer into neighboring buckets (i.e., k-mers) within radius r, if two k-mers $\boldsymbol{s}$ and $\boldsymbol{t}$ are $2r+1$ edits apart, then they will be mapped to disjoint buckets. Formally, if $\mathrm{edit}\,(\boldsymbol{s}, \boldsymbol{t}) \ge 2r+1$, then $f_r^B(\boldsymbol{s}) \cap f_r^B(\boldsymbol{t}) = \varnothing$. This is true because $(\mathcal{S}_k, \mathrm{edit})$ is a metric space satisfying the triangle inequality. This implies that f_r^B satisfies the LSB-property (2) with $d_2 = 2r+1$. We note that this statement holds regardless of the choice of B.

Hence, to make f_r^B a $(d_1, 2r+1)$-sensitive bucketing function for some integer d_1, we only need to determine a subset B so that f_r^B satisfies the LSB-property (1). Specifically, B should be picked such that for any two k-mers $\boldsymbol{s}$ and $\boldsymbol{t}$ within an edit distance of d_1, we always have

$$f_r^B(\boldsymbol{s}) \cap f_r^B(\boldsymbol{t}) = (N_k^r(\boldsymbol{s}) \cap B) \cap (N_k^r(\boldsymbol{t}) \cap B) = N_k^r(\boldsymbol{s}) \cap N_k^r(\boldsymbol{t}) \cap B \neq \varnothing.$$

For the sake of simplicity, we say a set of buckets B is (d_1, r)-*guaranteed* if and only if the corresponding bucketing function f_r^B is $(d_1, 2r+1)$-sensitive. Equivalently, B is (d_1, r)-guaranteed if $N_k^r(\boldsymbol{s}) \cap N_k^r(\boldsymbol{t}) \cap B \neq \varnothing$ for every pair of k-mers $\boldsymbol{s}$ and $\boldsymbol{t}$ with $\text{edit}(\boldsymbol{s}, \boldsymbol{t}) \leq d_1$. In the following sections, we show several (d_1, r)-guaranteed subsets $B \subset \mathcal{S}_k$ for different values of d_1.

4.1 $(2r, r)$-guaranteed and $(2r-1, r)$-guaranteed subsets

We first consider an extreme case where $B = \mathcal{S}_k$.

Lemma 6. *Let $B = \mathcal{S}_k$. Then B is $(2r, r)$-guaranteed if r is even, and B is $(2r-1, r)$-guaranteed if r is odd.*

Proof. First consider the case that r is even. Let $\boldsymbol{s}$ and $\boldsymbol{t}$ be two k-mers with $\text{edit}(\boldsymbol{s}, \boldsymbol{t}) \leq 2r$. Then there is a sequence of $2r$ edits that transforms $\boldsymbol{s}$ to $\boldsymbol{t}$ (if $\text{edit}(\boldsymbol{s}, \boldsymbol{t}) < 2r$, we can add in trivial edits that substitute a character with itself). Because $\boldsymbol{s}$ and $\boldsymbol{t}$ have the same length, these $2r$ edits must contain the same number of insertions and deletions. Reorder the edits so that each insertion is followed immediately by a deletion (i.e., a pair of indels) and all the indels come before substitutions. Because r is even, in this new order, the first r edits contain an equal number of insertions and deletions. Namely, applying the first r edits on $\boldsymbol{s}$ produces a k-mer $\boldsymbol{v}$. Clearly, $\text{edit}(\boldsymbol{s}, \boldsymbol{v}) \leq r$ and $\text{edit}(\boldsymbol{t}, \boldsymbol{v}) \leq r$, i.e., $\boldsymbol{v} \in N_k^r(\boldsymbol{s}) \cap N_k^r(\boldsymbol{t}) = N_k^r(\boldsymbol{s}) \cap N_k^r(\boldsymbol{t}) \cap B$.

For the case that r is odd. Let $\boldsymbol{s}$ and $\boldsymbol{t}$ be two k-mers with $\text{edit}(\boldsymbol{s}, \boldsymbol{t}) \leq 2r-1$. By the same argument as above, $\boldsymbol{s}$ can be transformed to $\boldsymbol{t}$ by a sequence of $2r-1$ edits and we can assume that all the indels appear in pairs and they come before all the substitutions. Because r is odd, $r-1$ is even. So applying the first $r-1$ edits on $\boldsymbol{s}$ produces a k-mer $\boldsymbol{v}$ such that $\text{edit}(\boldsymbol{s}, \boldsymbol{v}) \leq r-1 < r$ and $\text{edit}(\boldsymbol{t}, \boldsymbol{v}) \leq 2r-1-(r-1) = r$. Therefore, $\boldsymbol{v} \in N_k^r(\boldsymbol{s}) \cap N_k^r(\boldsymbol{t}) = N_k^r(\boldsymbol{s}) \cap N_k^r(\boldsymbol{t}) \cap B$. $\square$

By definition, setting $B = \mathcal{S}_k$ makes f_r^B $(2r, 2r+1)$-sensitive if r is even and $(2r-1, 2r+1)$-sensitive if r is odd. This provides nearly perfect bucketing performance in the sense that there is no gap (when r is even) or the gap is one (when r is odd). It is evident from the proof that the gap at $2r$ exists when r is odd because if $\boldsymbol{s}$ can only be transformed to $\boldsymbol{t}$ by r pairs of indels, then there is no k-mer $\boldsymbol{v}$ with $\text{edit}(\boldsymbol{s}, \boldsymbol{v}) = \text{edit}(\boldsymbol{t}, \boldsymbol{v}) = r$.

4.2 Properties of (r, r)-guaranteed subsets

A natural question is, can we use a proper subset of $\mathcal{S}_k$ to achieve (gapped) LSB functions? This can be viewed as down-sampling $\mathcal{S}_k$ such that if two k-mers $\boldsymbol{s}$ and $\boldsymbol{t}$ are similar, then a k-mer is always sampled from $N_k^r(\boldsymbol{s}) \cap N_k^r(\boldsymbol{t})$.

Here we focus on the case that $d_1 = r$, i.e., we aim to construct B that is (r, r)-guaranteed. Recall that this means for any $\boldsymbol{s}, \boldsymbol{t} \in \mathcal{S}_k$ with $\text{edit}(\boldsymbol{s}, \boldsymbol{t}) \leq r$, we have $N_k^r(\boldsymbol{s}) \cap N_k^r(\boldsymbol{t}) \cap B \neq \varnothing$. In other words, f_r^B is $(r, 2r+1)$-sensitive. To prepare the construction, we first investigate some structural properties of (r, r)-guaranteed subsets. We propose a conjecture that they form a hierarchical structure with decreasing r:

Conjecture 1. *If $B \subset \mathcal{S}_k$ is (r, r)-guaranteed, then B is also $(r+1, r+1)$-guaranteed.*

We prove a weaker statement:

Lemma 7. *If $B \subset \mathcal{S}_k$ is (r, r)-guaranteed, then B is $(r+2, r+2)$-guaranteed.*

Proof. Let $\boldsymbol{s}$ and $\boldsymbol{t}$ be two k-mers with $\text{edit}(\boldsymbol{s}, \boldsymbol{t}) \leq r+2$; we want to show that $N_k^{r+2}(\boldsymbol{s}) \cap N_k^{r+2}(\boldsymbol{t}) \cap B \neq \varnothing$. Consider a sequence of edits that transforms $\boldsymbol{s}$ to $\boldsymbol{t}$: skipping a pair of indels or two substitutions gives a k-mer $\boldsymbol{m}$ such that $\text{edit}(\boldsymbol{s}, \boldsymbol{m}) \leq r$ and $\text{edit}(\boldsymbol{t}, \boldsymbol{m}) = 2$. Because B is (r, r)-guaranteed, we have that $N_k^r(\boldsymbol{s}) \cap N_k^r(\boldsymbol{m}) \cap B \neq \varnothing$, i.e., there exists a k-mer $\boldsymbol{v} \in B$ such that $\text{edit}(\boldsymbol{s}, \boldsymbol{v}) \leq r$ and $\text{edit}(\boldsymbol{m}, \boldsymbol{v}) \leq r$. By triangle inequality, $\text{edit}(\boldsymbol{t}, \boldsymbol{v}) \leq \text{edit}(\boldsymbol{t}, \boldsymbol{m}) + \text{edit}(\boldsymbol{m}, \boldsymbol{v}) \leq r+2$. Hence, we have $\boldsymbol{v} \in N_k^{r+2}(\boldsymbol{t})$. Clearly, $\boldsymbol{v} \in N_k^r(\boldsymbol{s})$ implies that $\boldsymbol{v} \in N_k^{r+2}(\boldsymbol{s})$. Combined, we have $\boldsymbol{v} \in N_k^{r+2}(\boldsymbol{s}) \cap N_k^{r+2}(\boldsymbol{t}) \cap B$. $\square$

The next lemma shows that $(1, 1)$-guaranteed subset has the strongest condition.

Lemma 8. *If $B \subset S_k$ is $(1, 1)$-guaranteed, then B is (r, r)-guaranteed for all $r \geq 1$.*

Proof. According to the previous lemma, we only need to show that B is $(2, 2)$-guaranteed. Given k-mers $\boldsymbol{s}$ and $\boldsymbol{t}$ with $\text{edit}(\boldsymbol{s}, \boldsymbol{t}) = 2$, consider a sequence Q of two edits that transforms $\boldsymbol{s}$ to $\boldsymbol{t}$. There are two possibilities:

- If both edits in Q are substitutions, let i be the position of the first substitution.
- If Q consists of one insertion and one deletion, let i be the position of the character that is going to be deleted from $\boldsymbol{s}$.

In either case, let $\boldsymbol{m}$ be a k-mer obtained by replacing the i-th character of $\boldsymbol{s}$ with another symbol in Σ. Then $\text{edit}(\boldsymbol{s}, \boldsymbol{m}) = 1$. Because B is $(1, 1)$-guaranteed, there is a k-mer $\boldsymbol{v} \in B$ such that $\text{edit}(\boldsymbol{s}, \boldsymbol{v}) \leq 1$ and $\text{edit}(\boldsymbol{m}, \boldsymbol{v}) \leq 1$. Observe that either $\boldsymbol{s} = \boldsymbol{v}$ or $\boldsymbol{v}$ is obtained from $\boldsymbol{s}$ by one substitution at position i. So applying the two edits in Q on $\boldsymbol{v}$ also produces $\boldsymbol{t}$, i.e., $\text{edit}(\boldsymbol{t}, \boldsymbol{v}) \leq 2$. Therefore, $\boldsymbol{v} \in N_k^2(\boldsymbol{s}) \cap N_k^2(\boldsymbol{t}) \cap B$. $\square$

Now we bound the size of a $(1, 1)$-guaranteed subset from below.

Lemma 9. *If B is (1,1)-guaranteed, then*

$$\text{(i) for each } \boldsymbol{s} \in \mathcal{S}_k, \left|N_k^1(\boldsymbol{s}) \cap B\right| \geq \begin{cases} 1 & \text{if } \boldsymbol{s} \in B \\ k & \text{if } \boldsymbol{s} \notin B \end{cases}, \qquad \text{(ii) } |B| \geq |\mathcal{S}_k|/|\Sigma| = |\Sigma|^{k-1}.$$

Proof. Let $B \subset \mathcal{S}_k$ be an arbitrary $(1, 1)$-guaranteed subset. For part (i), because $\boldsymbol{s} \in N_k^1(\boldsymbol{s})$, if $\boldsymbol{s}$ is also in B, then $\boldsymbol{s}$ is in their intersection, hence $\left|N_k^1(\boldsymbol{s}) \cap B\right| \geq 1$. If $\boldsymbol{s} = s_1 s_2 \dots s_k \notin B$, then it must have at least k 1-neighbors $\boldsymbol{v}^i \in B$, one for each position $1 \leq i \leq k$, where $\boldsymbol{v}^i = s_1 \dots s_{i-1} v_i s_{i+1} \dots s_k$, $v_i \neq s_i$. Suppose conversely that this is not the case for a particular i. Let $\boldsymbol{t} = s_1 \dots s_{i-1} t_i s_{i+1} \dots s_k$ where $t_i \neq s_i$. We have $\text{edit}(\boldsymbol{s}, \boldsymbol{t}) = 1$. Also, $N_k^1(\boldsymbol{s}) \cap N_k^1(\boldsymbol{t}) = \{x \in \Sigma \mid s_1 \dots s_{i-1} x s_{i+1} \dots s_k\}$, but none of them is in B (consider the two cases $x = s_i$ and $x \neq s_i$), i.e., $N_k^1(\boldsymbol{s}) \cap N_k^1(\boldsymbol{t}) \cap B = \varnothing$. This contradicts the assumption that B is $(1, 1)$-guaranteed.

For part (ii), consider the collection of pairs $H = \left\{(\boldsymbol{s}, \boldsymbol{v}) \,\middle|\, \boldsymbol{s} \in \mathcal{S}_k \text{ and } \boldsymbol{v} \in N_k^1(\boldsymbol{s}) \cap B\right\}$. For all $\boldsymbol{v} \in B$, the number of k-mers $\boldsymbol{s} \in \mathcal{S}_k$ with $\text{edit}(\boldsymbol{s}, \boldsymbol{v}) \leq 1$ is $k(|\Sigma| - 1) + 1$. So $|H| = (k(|\Sigma| - 1) + 1)|B|$. On the other hand, part (i) implies that $|H| \geq |B| + k\left(|\Sigma|^k - |B|\right)$. Combined, we have $|B| \geq |\Sigma|^{k-1}$, as claimed. $\square$

In Section 4.3, we give an algorithm to construct a $(1, 1)$-guaranteed subset B that achieves the size $|B| = |\Sigma|^{k-1}$; furthermore, the corresponding $(1, 3)$-sensitive bucketing function f_1^B satisfies

$$\left|f_1^B(\boldsymbol{s})\right| = \begin{cases} 1 & \text{if } \boldsymbol{s} \in B \\ k & \text{if } \boldsymbol{s} \notin B \end{cases}.$$

This shows that the lower bounds proved above in Lemma 9 are tight and that the constructed $(1, 1)$-guaranteed subset B is optimal in the sense of minimizing both $|B|$ and $\left|f_1^B(\boldsymbol{s})\right|$. Notice that this result improves Lemma 6 with $r = 1$ where we showed that $\mathcal{S}^k$ is a $(1, 1)$-guaranteed subset of size $|\Sigma|^k$. According to Lemma 8, this constructed B is also (r, r)-guaranteed. So the corresponding bucketing function f_r^B is $(r, 2r + 1)$-sensitive.

4.3 Construction of optimal $(1, 1)$-guaranteed subsets

Let $n = |\Sigma|$ and denote the symbols in Σ by $c_1, c_2, \dots, c_n$. We describe a recursive procedure to construct a $(1, 1)$-guaranteed subset of $\mathcal{S}_k$. In fact, we show that $\mathcal{S}_k$ can be partitioned into n subsets $B_k^1 \sqcup B_k^2 \sqcup \dots \sqcup B_k^n$ such that each B_k^i is $(1, 1)$-guaranteed. Here the notation $\sqcup$ denotes disjoint union. The partition of $\mathcal{S}_k$ is built from the partition of $\mathcal{S}_{k-1}$. The base case is $\mathcal{S}_1 = \{c_1\} \sqcup \dots \sqcup \{c_n\}$.

Suppose that we already have the partition for $\mathcal{S}_{k-1} = B_{k-1}^1 \sqcup B_{k-1}^2 \sqcup \dots \sqcup B_{k-1}^n$. Let

$$B_k^1 = \left(c_1 \circ B_{k-1}^1\right) \sqcup \left(c_2 \circ B_{k-1}^2\right) \sqcup \dots \sqcup \left(c_n \circ B_{k-1}^n\right),$$

where $c \circ B^j_{k-1}$ is the set obtained by prepending the character c to each string in the set B^j_{k-1}. For B^2_k, the construction is similar where the partitions of $\mathcal{S}_{k-1}$ are shifted (rotated) by one such that c_1 is paired with B^2_{k-1}, c_2 is paired with B^3_{k-1}, and so on. In general, for $1 \le i \le n$,

$$B^i_k = \left(c_1 \circ B^i_{k-1}\right) \sqcup \left(c_2 \circ B^{i+1}_{k-1}\right) \sqcup \cdots \sqcup \left(c_{n-i+1} \circ B^n_{k-1}\right) \sqcup \left(c_{n-i+2} \circ B^1_{k-1}\right) \sqcup \cdots \sqcup \left(c_n \circ B^{i-1}_{k-1}\right).$$

Note that each k-mer in $\mathcal{S}_k$ appears in exactly one of the subsets B^i_k, justifying the use of the disjoint union notation. (The induction proof of this claim has identical structure as the following proofs of Lemma 10 and 11, so we leave it out for conciseness.) Now we prove the correctness of this construction.

Lemma 10. *Each constructed B^i_k is a minimum $(1,1)$-guaranteed subset of $\mathcal{S}_k$.*

Proof. By Lemma 9, we only need to show that each B^i_k is $(1,1)$-guaranteed and has size $|\Sigma|^{k-1} = n^{k-1}$. The base case $\mathcal{S}_1 = \{c_1\} \sqcup \cdots \sqcup \{c_n\}$ is easy to verify.

Suppose that $\mathcal{S}_{k-1} = \bigsqcup_{j=1}^n B^j_{k-1}$, each B^j_{k-1} is $(1,1)$-guaranteed and has size n^{k-2}. Consider an arbitrary index $1 \le i \le n$. We have $\left|B^i_k\right| = \sum_{j=1}^n \left|B^j_{k-1}\right| = n^{k-1}$. To show that B^i_k is $(1,1)$-guaranteed, consider two k-mers $\boldsymbol{s}, \boldsymbol{t} \in \mathcal{S}_k$ with $\text{edit}(\boldsymbol{s}, \boldsymbol{t}) = 1$. If the single substitution happens on the first character, let $\boldsymbol{x} \in \mathcal{S}_{k-1}$ be the common $(k-1)$-suffix of $\boldsymbol{s}$ and $\boldsymbol{t}$. Since $\bigsqcup_{j=1}^n B^j_{k-1}$ is a partition of $\mathcal{S}_{k-1}$, $\boldsymbol{x}$ must appear in one of the subsets B^ℓ_{k-1}. In B^i_k, it is paired with one of the characters c_m. Then $\boldsymbol{y} = c_m \circ \boldsymbol{x} \in B^i_k$. Furthermore, $\boldsymbol{s}$ and $\boldsymbol{t}$ can each be transformed to $\boldsymbol{y}$ by at most one substitution on the first character. Thus, $\boldsymbol{y} \in N^1_k(\boldsymbol{s}) \cap N^1_k(\boldsymbol{t}) \cap B^i_k$.

If the single substitution between $\boldsymbol{s}$ and $\boldsymbol{t}$ does not happen on the first position, then they share the common first character c_m. In B^i_k, c_m is paired with one of the subsets B^ℓ_{k-1}. Let $\boldsymbol{s'}$ and $\boldsymbol{t'}$ be $(k-1)$-suffixes of $\boldsymbol{s}$ and $\boldsymbol{t}$, respectively. Clearly, $\text{edit}(\boldsymbol{s'}, \boldsymbol{t'}) = 1$. By the induction hypothesis, B^ℓ_{k-1} is $(1,1)$-guaranteed. So there is a $(k-1)$-mer $\boldsymbol{x} \in B^\ell_{k-1}$ such that $\text{edit}(\boldsymbol{s'}, \boldsymbol{x}) \le 1$ and $\text{edit}(\boldsymbol{t'}, \boldsymbol{x}) \le 1$. Observe that $\boldsymbol{y} = c_m \circ \boldsymbol{x} \in B^i_k$. Furthermore, $\text{edit}(\boldsymbol{s}, \boldsymbol{y}) = \text{edit}(\boldsymbol{s'}, \boldsymbol{x}) \le 1$ and $\text{edit}(\boldsymbol{t}, \boldsymbol{y}) = \text{edit}(\boldsymbol{t'}, \boldsymbol{x}) \le 1$. Thus, $\boldsymbol{y} \in N^1_k(\boldsymbol{s}) \cap N^1_k(\boldsymbol{t}) \cap B^i_k$. Therefore, B^i_k is $(1,1)$-guaranteed. Since the index i is arbitrary, this completes the proof. □

It remains to show that for each $\boldsymbol{s} \in \mathcal{S}_k$, $\left|N^1_k(\boldsymbol{s}) \cap B^i_k\right|$ matches the lower bound in Lemma 9. Recall that $f^B_1(\boldsymbol{s}) = N^1_k(\boldsymbol{s}) \cap B$ is $(1,3)$-sensitive if B is $(1,1)$-guaranteed. Together with Lemma 10, this shows that for each constructed B^i_k, the corresponding $(1,3)$-sensitive bucketing function is optimal in terms of minimizing both the total number of buckets and the number of buckets each k-mer is sent to.

Lemma 11. *For $\boldsymbol{s} \in \mathcal{S}_k$, each constructed B^i_k satisfies* $\left|N^1_k(\boldsymbol{s}) \cap B^i_k\right| = \begin{cases} 1 & \text{if } \boldsymbol{s} \in B^i_k \\ k & \text{if } \boldsymbol{s} \notin B^i_k \end{cases}$.

Proof. We proceed by induction on k. The base case $k = 1$ is trivially true because $|B^i_1| = 1$ and all 1-mers are within 1 edit of each other. Suppose that the claim is true for $k-1$. Consider an arbitrary index i. If $\boldsymbol{s} \in B^i_k$, we show that any other k-mer $\boldsymbol{t} \in B^i_k$ has edit distance at least 2 from $\boldsymbol{s}$, namely $N^1_k(\boldsymbol{s}) \cap B^i_k = \{\boldsymbol{s}\}$. Let $\boldsymbol{s'}$ and $\boldsymbol{t'}$ be the $(k-1)$-suffixes of $\boldsymbol{s}$ and $\boldsymbol{t}$ respectively. According to the construction, if $\boldsymbol{s}$ and $\boldsymbol{t}$ have the same first character, then $\boldsymbol{s'}$ and $\boldsymbol{t'}$ are in the same B^j_{k-1} for some index j. By the induction hypothesis, $\text{edit}(\boldsymbol{s'}, \boldsymbol{t'}) \ge 2$ (otherwise $\left|N^1_{k-1}(\boldsymbol{s'}) \cap B^j_{k-1}\right| \ge 2$), and therefore $\text{edit}(\boldsymbol{s}, \boldsymbol{t}) = \text{edit}(\boldsymbol{s'}, \boldsymbol{t'}) \ge 2$. If $\boldsymbol{s}$ and $\boldsymbol{t}$ are different at the first character, then $\boldsymbol{s'}$ and $\boldsymbol{t'}$ are not in the same B^j_{k-1}, so $\text{edit}(\boldsymbol{s'}, \boldsymbol{t'}) \ge 1$ (recall that $\bigsqcup_{j=1}^n B^j_{k-1}$ is a partition of $\mathcal{S}_{k-1}$). Together with the necessary substitution at the first character, we have $\text{edit}(\boldsymbol{s}, \boldsymbol{t}) = 1 + \text{edit}(\boldsymbol{s'}, \boldsymbol{t'}) \ge 2$.

If $\boldsymbol{s} \notin B^i_k$, Lemma 9 and 10 guarantee that $\boldsymbol{s}$ has k 1-neighbors $\boldsymbol{v}^\ell$ in B^i_k, $\ell = 1, \ldots, k$, where $\boldsymbol{v}^\ell$ is obtained from $\boldsymbol{s}$ by a single substitution at position ℓ. Let $\boldsymbol{t} \ne \boldsymbol{s}$ be a 1-neighbor of $\boldsymbol{s}$. Since $\boldsymbol{t}$ can only differ from $\boldsymbol{s}$ by a single substitution at some position ℓ, we know that either $\boldsymbol{t} = \boldsymbol{v}^\ell$ or the edit distance between $\boldsymbol{t}$ and $\boldsymbol{v}^\ell$ is 1. In the latter case, $\boldsymbol{t}$ cannot be in B^i_k otherwise $\left|N^1_k\left(\boldsymbol{v}^\ell\right) \cap B^i_k\right| \ge 2$, contradicting the result of the previous paragraph. Therefore, $N^1_k(\boldsymbol{s}) \cap B^i_k = \left\{\boldsymbol{v}^1, \ldots \boldsymbol{v}^k\right\}$ which has size k. □

We end this section by showing that a membership query can be done in $O(k)$ time on the $(1,1)$-guaranteed subset B constructed above (i.e., $B = B^i_k$ for some i). Thanks to its regular structure, the query is performed without explicit construction of B. Consequently, the bucketing functions using B can be computed without computing and storing this subset of size $|\Sigma|^{k-1}$.

Specifically, suppose that we choose $B = B_k^i$ for some fixed $1 \le i \le n$. Let $\boldsymbol{s}$ be a given k-mer; we want to query if $\boldsymbol{s}$ is in B or not. This is equivalent to determining if the index of the partition of $\mathcal{S}_k$ that $\boldsymbol{s}$ falls into is i or not. Write $\boldsymbol{s} = s_1 s_2 \ldots s_k$ and let $\boldsymbol{s}' = s_2 \ldots s_k$ be the $(k-1)$-suffix of $\boldsymbol{s}$. Suppose that it has been determined that $\boldsymbol{s}' \in B_{k-1}^j$, i.e., the $(k-1)$-mer $\boldsymbol{s}'$ comes from the j-th partition of $\mathcal{S}_{k-1}$. By construction, the index m for which $\boldsymbol{s} \in B_k^m$ is uniquely determined by the symbol $s_1 = c_\ell \in \Sigma$ and the index j according to the formula $m = (j + n + 1 - \ell) \bmod n$. The base case $k = 1$ is trivially given by the design that $c_m \in B_1^m$ for all $1 \le m \le n$. This easily translates into a linear-time algorithm that scans the input k-mer $\boldsymbol{s}$ backwards and compute the index m such that $\boldsymbol{s} \in B_k^m$. To answer the membership query, we only need to check whether $m = i$.

4.4 A $(3,5)$-sensitive bucketing function

Let $B \subset \mathcal{S}_k$ be one of the constructed $(1,1)$-guaranteed subsets. Recall that the resulting bucketing function f_r^B is $(r, 2r+1)$-sensitive; in particular, f_2^B is $(2,5)$-sensitive. We are able to strengthen this result by showing that f_2^B is in fact $(3,5)$-sensitive.

Theorem 2. *Let $B \subset \mathcal{S}_k$ be a $(1,1)$-guaranteed subset. The bucketing function f_2^B is $(3,5)$-sensitive.*

Proof. As f_r^B is already proved to be $(2,5)$-sensitive, to show it is $(3,5)$-sensitive, we just need to prove that, for any two k-mers $\boldsymbol{s}, \boldsymbol{t} \in \mathcal{S}_k$ with $\text{edit}(\boldsymbol{s}, \boldsymbol{t}) = 3$, $f_2^B(\boldsymbol{s}) \cap f_2^B(\boldsymbol{t}) = N_k^2(\boldsymbol{s}) \cap N_k^2(\boldsymbol{t}) \cap B \neq \varnothing$. If the three edits are all substitutions, then there are k-mers $\boldsymbol{x}$ and $\boldsymbol{y}$ such that $\text{edit}(\boldsymbol{s}, \boldsymbol{x}) = \text{edit}(\boldsymbol{x}, \boldsymbol{y}) = \text{edit}(\boldsymbol{y}, \boldsymbol{t}) = 1$. Since B is $(1,1)$-guaranteed, there is a k-mer $\boldsymbol{z} \in B$ with $\text{edit}(\boldsymbol{x}, \boldsymbol{z}) \le 1$ and $\text{edit}(\boldsymbol{y}, \boldsymbol{z}) \le 1$. By triangle inequality, $\text{edit}(\boldsymbol{s}, \boldsymbol{z}) \le \text{edit}(\boldsymbol{s}, \boldsymbol{x}) + \text{edit}(\boldsymbol{x}, \boldsymbol{z}) \le 2$; $\text{edit}(\boldsymbol{t}, \boldsymbol{z}) \le \text{edit}(\boldsymbol{t}, \boldsymbol{y}) + \text{edit}(\boldsymbol{y}, \boldsymbol{z}) \le 2$. So $\boldsymbol{z} \in N_k^2(\boldsymbol{s}) \cap N_k^2(\boldsymbol{t}) \cap B$.

If the three edits are one substitution and a pair of indels, then there is a k-mer $\boldsymbol{x}$ such that $\text{edit}(\boldsymbol{s}, \boldsymbol{x}) = 1$ and $\text{edit}(\boldsymbol{x}, \boldsymbol{t}) = 2$ where the two edits between $\boldsymbol{x}$ and $\boldsymbol{t}$ can only be achieved by one insertion and one deletion. Let i be the position in $\boldsymbol{x}$ where the deletion between $\boldsymbol{x}$ and $\boldsymbol{t}$ takes place. Let $\boldsymbol{y}$ be a k-mer obtained from $\boldsymbol{x}$ by a substitution at position i, so $\text{edit}(\boldsymbol{x}, \boldsymbol{y}) = 1$. Since B is $(1,1)$-guaranteed, there is a k-mer $\boldsymbol{z} \in B$ with $\text{edit}(\boldsymbol{x}, \boldsymbol{z}) \le 1$ and $\text{edit}(\boldsymbol{y}, \boldsymbol{z}) \le 1$. Then $\text{edit}(\boldsymbol{s}, \boldsymbol{z}) \le \text{edit}(\boldsymbol{s}, \boldsymbol{x}) + \text{edit}(\boldsymbol{x}, \boldsymbol{z}) \le 2$. Observe that $\boldsymbol{x}$ and $\boldsymbol{z}$ differ by at most one substitution at position i, which will be deleted when transforming to $\boldsymbol{t}$. So the two edits from $\boldsymbol{x}$ to $\boldsymbol{t}$ can also transform $\boldsymbol{z}$ to $\boldsymbol{t}$, namely, $\text{edit}(\boldsymbol{t}, \boldsymbol{z}) \le 2$. Thus, $\boldsymbol{z} \in N_k^2(\boldsymbol{s}) \cap N_k^2(\boldsymbol{t}) \cap B$. ☐

5 Summary of proved LSB functions

We proposed two sets of locality-sensitive bucketing functions and studied the efficiency of them in terms of $|B|$, the total number of buckets, and $|f(\boldsymbol{s})|$, the number of buckets a specific k-mer $\boldsymbol{s}$ occupies. The results are summarized in Table 1.

Table 1: Results on (d_1, d_2)-sensitive bucketing functions of k-mers. Entries with $\le$ show the best known upper bounds. Entries marked with a single star cannot be reduced under the specific bucketing method. Entries marked with double stars cannot be reduced in general. In column B, we use B_k^i to refer to the $(1,1)$-guaranteed subset constructed in Section 4.3.

(d_1, d_2)-sensitive	B	$\lvert B\rvert$	$\lvert f(\boldsymbol{s})\rvert$	Ref.
$(1,2)$	$\{1, \ldots, \lvert B\rvert\}$	$k\lvert\Sigma\rvert^{k-1}**$	$k**$	Theorem 1
$(1,3)$	$\mathcal{S}_k$	$\lvert\Sigma\rvert^k$	$\lvert N_k^1(\boldsymbol{s})\rvert = (\lvert\Sigma\rvert - 1)k + 1$	Lemma 6
$(1,3)$	B_k^i	$\lvert\Sigma\rvert^{k-1}*$	$\begin{cases} 1 & \text{if } \boldsymbol{s} \in B \\ k & \text{if } \boldsymbol{s} \notin B \end{cases}*$	Lemma 9–11
$(3,5)$	B_k^i	$\lvert\Sigma\rvert^{k-1}$	$\le \lvert N_k^2(\boldsymbol{s})\rvert$	Theorem 2
$(r, 2r+1)$, $r > 1$	B_k^i	$\lvert\Sigma\rvert^{k-1}$	$\le \lvert N_k^r(\boldsymbol{s})\rvert$	Lemma 8, 10
$(2r-1, 2r+1)$, $r \ge 3$ odd	$\mathcal{S}_k$	$\lvert\Sigma\rvert^k$	$\lvert N_k^r(\boldsymbol{s})\rvert$	Lemma 6
$(2r, 2r+1)$, $r \ge 2$ even	$\mathcal{S}_k$	$\lvert\Sigma\rvert^k$	$\lvert N_k^r(\boldsymbol{s})\rvert$	Lemma 6

6 Experimental results on the gapped LSB functions

Several gapped LSB functions are introduced in Section 4. Now we investigate their behavior at the gap. We pick 3 LSB functions to experiment, corresponding to the rows 2–4 in Table 1. For $d = 1, 2, \ldots, 6$, we generate 100,000 random pairs of 20-mers $(\boldsymbol{s}, \boldsymbol{t})$ with edit distance d. Each one the picked LSB function f_r^B is applied and the number of pairs that share a bucket under f_r^B is recorded. The results are shown in Figure 1.

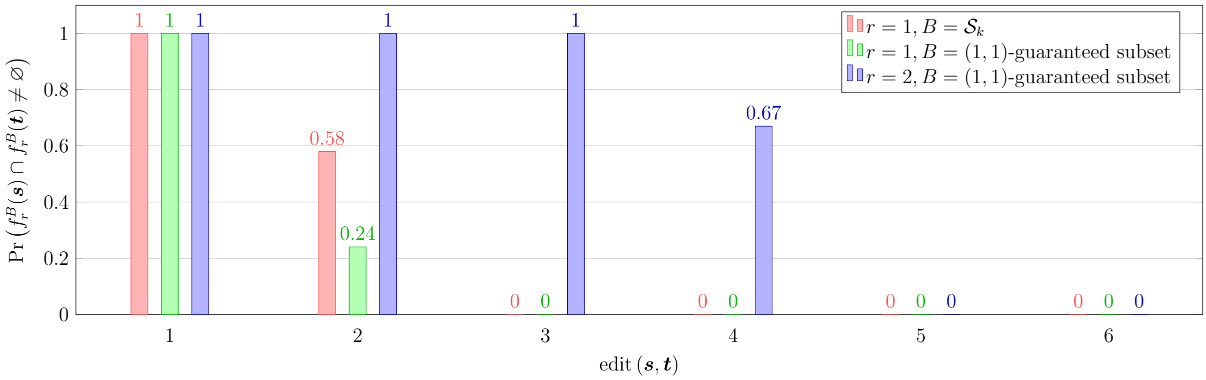
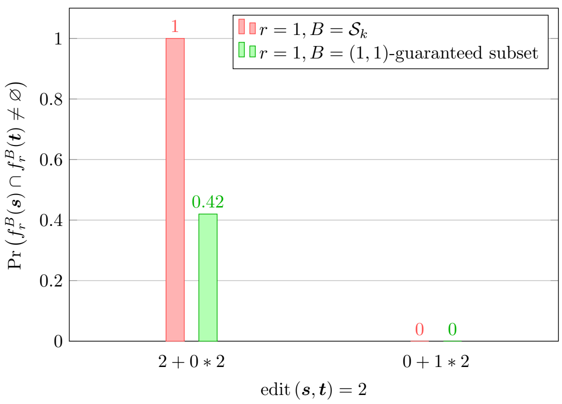


Figure 1: Probabilities that two k-mers share a bucket with respect to their edit distance under three gapped LSB functions (red, green, and blue bars correspond to the rows 2–4 of Table 1).

Recall that Lemma 6 implies $f_r^{\mathcal{S}_k}$ is $(2r-1, 2r+1)$-sensitive when r is odd. The discussion after the proof shows that the gap at $2r$ indeed exists. In particular, if $\boldsymbol{s}$ can only be transformed to $\boldsymbol{t}$ by r pairs of indels, then $N_k^r(\boldsymbol{s}) \cap N_k^r(\boldsymbol{t}) = \varnothing$. On the other hand, if there are some substitutions among the $2r$ edits between $\boldsymbol{s}$ and $\boldsymbol{t}$, then by a similar construction as in the case where r is even, we can find a k-mer $\boldsymbol{v}$ such that edit $(\boldsymbol{s}, \boldsymbol{v}) =$ edit $(\boldsymbol{v}, \boldsymbol{t}) = r$. Motivated by the this observation, we further explore the performance of the LSB functions at the gap for different types of edits. Given a gapped LSB function f, for the gap at d, define categories $0, \ldots, \lfloor d/2 \rfloor$ corresponding to the type of edits: a pair of k-mer with edit distance d is in the i-th category if they can be transformed to each other with i pairs of indels (and $d - 2i$ substitutions) but not $i-1$ pairs of indels (and $d - 2i + 2$ substitutions). Figure 2 shows the results for the three LSB functions in Figure 1 at their respective gaps with respect to different types of edits. Observe that the result for $f_1^{\mathcal{S}_k}$ (in red) agrees with our analysis above.



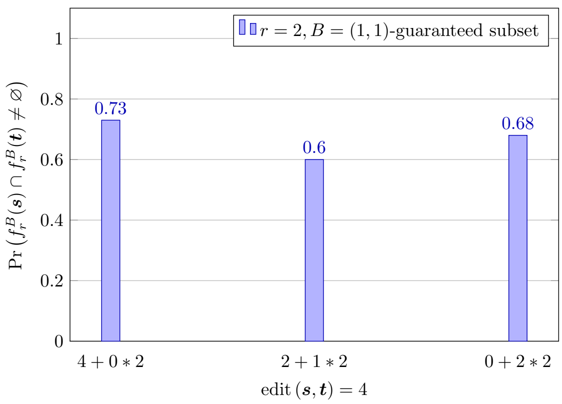


Figure 2: Probabilities that two k-mers share a bucket with respect to their edit type under three gapped LSB functions. The types of edits are labeled in the format $a+b*2$ where a is the number of substitutions and b is the number of pairs of indels. Left: two $(1,3)$-sensitive bucketing functions (rows 2 and 3 of Table 1). Right: the $(3,5)$-sensitive bucketing function (row 4 of Table 1).

7 Conclusion and Discussion

We introduce locality-sensitive bucketing (LSB) functions, that generalize locality-sensitive hashing (LSH) functions by allowing it to map a sequence into multiple buckets. This generalization makes the LSB functions easier to construct, while guaranteeing perfect sensitivity and specificity in a deterministic manner. We construct such functions, prove their properties, and show that some of them are optimal under proposed criteria. We also reveal several properties and structures about the k-mer space, which are of independent interests for studying LSH functions and edit distance.

LSB function has a potential to be widely applicable in comparing sequencing data with high error rate, which is a major motivation for this work. We now give a concrete example on how LSB functions can be used to detect overlaps from a collection of long reads, a well-known problem that is critical in constructing overlap graphs. Given N long reads, we want to find all pairs that have overlaps. The long reads are first split into k-mers (for example, $k = 21$) where each k-mer carries an identifier/pointer to its original read. The k-mers are then sent to buckets according to an LSB function. Reads that are in the same bucket are considered candidate pairs. This process rules out pairs of reads that do not share any similar k-mers (governed by d_2) and therefore is able to avoid many unnecessary comparisons, while at the same time being very sensitive as two overlapping reads are likely to contain some k-mers that are just a few edits apart (governed by d_1).

Our results for LSB functions can be improved in several aspects. An obvious open problem is to design (d_1, d_2)-sensitive functions that are not covered here. For this purpose, one direction is to construct optimal (r, r)-guaranteed subsets for $r > 1$. As an implication of Lemma 11, it is worth noting that the optimal $(1, 1)$-guaranteed subset is a maximal independent set in the undirected graph G_k^1 whose vertex set is $\mathcal{S}_k$ and each k-mer is connected to all its 1-neighbors. It is natural to suspect that similar results hold for larger r. Another approach is to use other more well-studied sets as buckets and define LSB functions based on their connections with $\mathcal{S}_k$. This is closely related to the problem of embedding $\mathcal{S}_k$ which is difficult as noted in the introduction. Our results in Section 3 suggest a new angle to this challenging problem: instead of restricting our attention to embedding $\mathcal{S}_k$ into metric spaces, it may be beneficial to consider a broader category of spaces that are equipped with a non-transitive relation (here we used subsets of integers with the "have a nonempty intersection" relation). Yet another interesting future research direction would be to explore the possibility of improving the practical time and space efficiency of computing and applying LSB functions.

A technique commonly used to boost the sensitivity of a LSH function is known as the OR-amplification. It combines multiple LSH functions in parallel, which can be viewed as sending each sequence into multiple buckets such that the probability of having similar sequences in one bucket is higher than using the individual functions separately. However, as a side effect, the OR-amplification hurts specificity: the chance that dissimilar sequences share a bucket also increases. It is therefore necessary to combine it with other techniques and choosing parameters to balance sensitivity and specificity is a delicate work. On contrast, the LSB function introduced in this paper achieves a provable optimal separation of similar and dissimilar sequences.

References

[1] Stephen F Altschul, Warren Gish, Webb Miller, Eugene W Myers, and David J Lipman. Basic local alignment search tool. *Journal of molecular biology*, 215(3):403–410, 1990.

[2] Stephen F Altschul, Thomas L Madden, Alejandro A Schäffer, Jinghui Zhang, Zheng Zhang, Webb Miller, and David J Lipman. Gapped blast and psi-blast: a new generation of protein database search programs. *Nucleic acids research*, 25(17):3389–3402, 1997.

[3] Z. Bar-Yossef, T.S. Jayram, R. Krauthgamer, and R. Kumar. Approximating edit distance efficiently. In *45th Annual IEEE Symposium on Foundations of Computer Science*, pages 550–559, 2004.

[4] Konstantin Berlin, Sergey Koren, Chen-Shan Chin, James P Drake, Jane M Landolin, and Adam M Phillippy. Assembling large genomes with single-molecule sequencing and locality-sensitive hashing. *Nature Biotechnology*, 33(6):623–630, 2015.

[5] Junjie Chen, Mingyue Guo, Xiaolong Wang, and Bin Liu. A comprehensive review and comparison of different computational methods for protein remote homology detection. *Briefings in Bioinformatics*, 19(2):231–244, 2018.

[6] Dan DeBlasio, Fiyinfoluwa Gbosibo, Carl Kingsford, and Guillaume Marçais. Practical universal k-mer sets for minimizer schemes. In *Proceedings of the 10th ACM International Conference on Bioinformatics, Computational Biology and Health Informatics (BCB'19)*, page 167–176, New York, NY, USA, 2019. Association for Computing Machinery.

[7] Miten Jain, Sergey Koren, Karen H Miga, Josh Quick, Arthur C Rand, Thomas A Sasani, John R Tyson, Andrew D Beggs, Alexander T Dilthey, Ian T Fiddes, et al. Nanopore sequencing and assembly of a human genome with ultra-long reads. *Nature Biotechnology*, 36(4):338–345, 2018.

[8] Robert Krauthgamer and Yuval Rabani. Improved lower bounds for embeddings into l_1. *SIAM Journal on Computing*, 38(6):2487–2498, 2009.

[9] Heng Li. Minimap2: pairwise alignment for nucleotide sequences. *Bioinformatics*, 34(18):3094–3100, 2018.

[10] Guillaume Marçais, Dan DeBlasio, and Carl Kingsford. Asymptotically optimal minimizers schemes. *Bioinformatics*, 34(13):i13–i22, 2018.

[11] Guillaume Marçais, Dan DeBlasio, Prashant Pandey, and Carl Kingsford. Locality-sensitive hashing for the edit distance. *Bioinformatics*, 35(14):i127–i135, 2019.

[12] Samuel McCauley. Approximate similarity search under edit distance using locality-sensitive hashing. In *24th International Conference on Database Theory (ICDT 2021)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2021.

[13] Yaron Orenstein, David Pellow, Guillaume Marçais, Ron Shamir, and Carl Kingsford. Designing small universal k-mer hitting sets for improved analysis of high-throughput sequencing. *PLoS Computational Biology*, 13(10):e1005777, 2017.

[14] Rafail Ostrovsky and Yuval Rabani. Low distortion embeddings for edit distance. *Journal of the ACM (JACM)*, 54(5):23–es, 2007.

[15] Anthony Rhoads and Kin Fai Au. PacBio sequencing and its applications. *Genomics, Proteomics & Bioinformatics*, 13(5):278–289, 2015.

[16] Michael Roberts, Wayne Hayes, Brian R Hunt, Stephen M Mount, and James A Yorke. Reducing storage requirements for biological sequence comparison. *Bioinformatics*, 20(18):3363–3369, 2004.

[17] Michael Roberts, Brian R Hunt, James A Yorke, Randall A Bolanos, and Arthur L Delcher. A preprocessor for shotgun assembly of large genomes. *Journal of Computational Biology*, 11(4):734–752, 2004.

[18] Kristoffer Sahlin, Marta Tomaszkiewicz, Kateryna D Makova, and Paul Medvedev. Deciphering highly similar multigene family transcripts from Iso-Seq data with IsoCon. *Nature Communications*, 9(1):1–12, 2018.

[19] Saul Schleimer, Daniel S Wilkerson, and Alex Aiken. Winnowing: local algorithms for document fingerprinting. In *Proceedings of the 2003 ACM SIGMOD (International Conference on Management of Data)*, pages 76–85, 2003.

[20] Yan Song, Haixu Tang, Haoyu Zhang, and Qin Zhang. Overlap detection on long, error-prone sequencing reads via smooth q-gram. *Bioinformatics*, 36(19):4838–4845, 2020.