Subsets and Supermajorities: Unifying Hashing-based Set Similarity Search

Thomas D. Ahle
IT University of Copenhagen and BARC
thdy@itu.dk

Jakob B. T. Knudsen
University of Copenhagen and BARC
jakn@di.ku.dk

November 4, 2019

Abstract

We give a simple data structure for Approximate Set Similarity Search, which includes important problems such as Partial Match, Subset Search, and Maximum Inner Product Search over $\{0,1\}^d$, that generalizes and outperforms all previous algorithms such as MinHash [J. Discrete Algorithms 1998], SimHash [STOC 2002], Spherical LSF [WADS 2007, SODA 2017] and Chosen Path [STOC 2017]. We show a number of lower bounds in the model of Locality Sensitive Hashing (LSH) / Filters (LSF), using (p-biased) hypercontractive inequalities, for a wide range of parameters, including space/time trade-offs. This answers the main open question in Christiani and Pagh [STOC 2017] on unifying the landscape of (non-data-dependent) Locality Sensitive (LSH/LSF [STOC 1998]) set similarity search; and provides the first such data structure with smooth time/space trade-off.

Previous algorithms considered different measures of set similarity, such as Jaccard, Cosine and Braun-Blanquet, which made the algorithms hard to compare. However fixing the weights of the sets these measures all become equivalent. Building separate data structures for sets with different norms, was shown by Yan et. al [NeurIPS 2018] to also be the fastest in practice when solving the Maximum Inner Product Search problem.

We thus define the generalized problem of (w_q, w_u, w_1, w_2)-Gap Set Similarity Search, GapSS, on a universe U, where $1 \geq w_q, w_u \geq w_1 > w_2 \geq 0$; to be constructing a data structure with n sets of weight $w_u|U|$ such that given a query, $q \subseteq U$, with weight $|q| = w_q|U|$ we can efficiently either find y in the dataset such that $|q \cap y| > w_2|U|$ or decide that intersection with all data sets is smaller than $w_1|U|$.

The specific case $w_1 = w_q$ corresponds to a "subset query" (also known as partial match) studied at least since Ronald Rivest's [PhD thesis 1974]. Our data structure gives a complete space/time trade-off, as an example for $w_q = 0.02, w_u = 0.1, w_1 = 0.02, w_2 = 0.002$ we give the first data structures for each of the following:

- Space $n^{1.283}$ and query time $n^{0.283}$. (MinHash uses space $n^{1.394}$ and query time $n^{0.394}$)
- Space $n^{1.599}$ and query time $n^{o(1)}$. (Spherical uses space $n^{4.44}$ and query time $n^{o(1)}$)
- Space $n^{1+o(1)}$ and query time $n^{0.808}$. (Spherical uses space $n^{1+o(1)}$ and query time $n^{0.816}$).

At all three trade-offs, the results in the parenthesis describes the currently best known. In fact, to get the stated results with Spherical LSF[1] one needs a new optimal affine embedding first described in this paper.

[1]In this particular case data dependent and non data dependent LSF gives the same bound.

Contents

1 Introduction

Sparse boolean vectors arises from the classical representation of documents as “bags of words”, where non-zero vector entries correspond to occurrences of words (or shingles). Another example is one-hot encoding of categorical data, such as which movies a user has watched. See e.g. [31] for a recent survey of applications.

Data structures for such data has been constructed for queries such as Superset/Subset/Containment, Partial Match, Jaccard similarity and maximum inner product search (MIPS). Early work goes back to Ronald Rivest’s thesis [45] with much follow up work, e.g. [17, 24, 30]. Unfortunately these problems are equivalent to the Orthogonal Vectors problem [19], which means that we can’t do much better than a brute force search through the database.

Hence recent research has focused on approximate versions of the problem, with MinHash (a.k.a. min-wise hashing) by Broder et al.[15, 14] being a landmark result. These problems are usually defined over some “similarity measure” like Jaccard similarity or inner product, with Braun Blanquet similarity being a recent breakthrough [23]. One can observe, however, that knowing the size of the sets in the database and queries makes all of these equivalent, including more than 76 binary similarity (and distance) measures defined in the survey [21]. This method, sometimes known as “norm ranging” is also practical, giving state of the art results [50].

We thus define the Gap Similarity Search problem, as the approximate set similarity search problem that is aware of the set weights:

Definition 1 (The (w_q, w_u, w_1, w_2)-GapSS problem)**.** *Given some universe U and a collection $Y \subseteq \binom{U}{w_u|U|}$ of sets of size $w_u|U|$, build a data structure that for any query set $q \subseteq \binom{U}{w_q|U|}$ of size $w_q|U|$: either returns $y' \in Y$ with $|y' \cap q| > w_2|U|$; or determines that there is no $y \in Y$ with $|y \cap q| \geq w_1|U|$.*

GapSS includes approximate subset search by setting $w_1 = w_u$ (the overlap between the sets must equal the size of the stored sets) and superset search by setting $w_1 = w_q$. The (j_1, j_2)-Jaccard Similarity Search problem corresponds to setting $w_1 = \frac{j_1(w_q+w_u)}{1+j_1}$ and $w_2 = \frac{j_2(w_q+w_u)}{1+j_2}$; and so on for other similarity measures.

Results in this area are usually phrased in terms of two quantities: 1) The "query exponent" $\rho_q \in [0, 1]$ which determines the query time by bounding it by $O(n^{\rho_q})$. 2) The "update exponent" $\rho_u \in [0, 1]$ which determines the time required to update the data structure when a point is inserted or deleted in Y and is given by $O(n^{\rho_u})$. The update exponent also bounds the space usage as $O(n^{1+\rho_u})$. Given a set of parameters (w_q, w_u, w_1, w_2), the important question is for which pairs of (ρ_q, ρ_u) there exists data structures. E.g. given a space budget imposed by ρ_u, we ask how small can one make ρ_q?

The three current algorithms which solves the above problem with the smallest exponents are MinHash [14], Chosen Path [23] and Spherical [48, 7]. These algorithms are all so called Locality Sensitive Hashing, LSH, algorithms, where the two later use the expanded Locality Sensitive Filter, LSF, framework [12]. We give a more thorough review of related work in Section 1.3 at the end of the introduction, but these three will be our main points of comparison throughout the article.

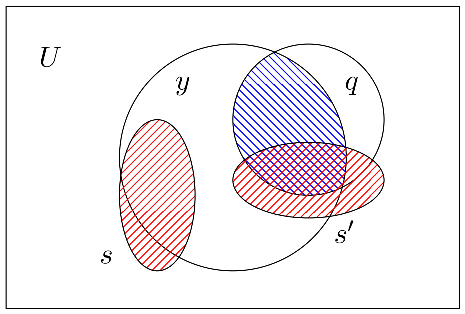
An important distinction is between data dependent and independent LSH algorithms. Both types are based on space partitions, but the former allows the space partition to depend on the dataset to be stored. In this article we mostly ignore this issue. That makes sense because the data dependency usually works by reducing to the case special case $w_2 = w_q w_u$ of the data independent problem, known as the "random setting". Hence *by solving the general problem we also pave the way for later data dependent algorithms.* For this reason, we also focus our lower bounds on the $w_2 = w_q w_u$ case, and we use it for all comparisons we make, such that the effect of data dependency doesn't show up.

The three algorithms above all solve the GapSS problem, with different algorithms being more efficient for different ranges of parameters. While MinHash has been thoroughly studied for half a century, it is unclear whether there is a single point (ρ_q, ρ_u) in which it is optimal. Only Spherical LSF provides supports (ρ_q, ρ_u) pairs where $\rho_q \neq \rho_u$. In [23] it was left as the main open problem to unify the above methods, ideally finding the optimal LSH algorithm for set data.

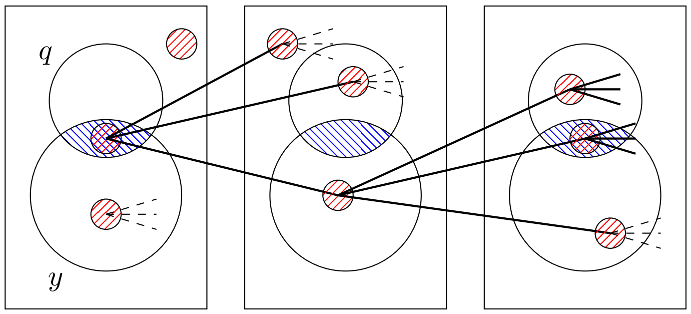
This is the problem we solve in this paper. *Our algorithm provably dominates all previous algorithms at all trade-offs and parameters* for data independent GapSS; and we provide lower bounds showing that this is finally the right solution to this generalised classical problem.

A Simple Algorithm The data structure partitions the sets in Y in such a way that queries only have to look at a few parts to find sets with high overlaps. While our real data structure takes the shape of a tree, the partitions can be understood purely geometrically, or in terms of Boolean functions and social choice theory.

The new (abstract) algorithm: We sample a number of representative sets $R \subseteq \binom{U}{k}$ uniformly independent among the size k subsets of U. We say a representative, $r \in R$, is t_u-"in favour of" a



(a) Two cohorts and two representative sets. The first representative favours y, while the second favours both y and q.



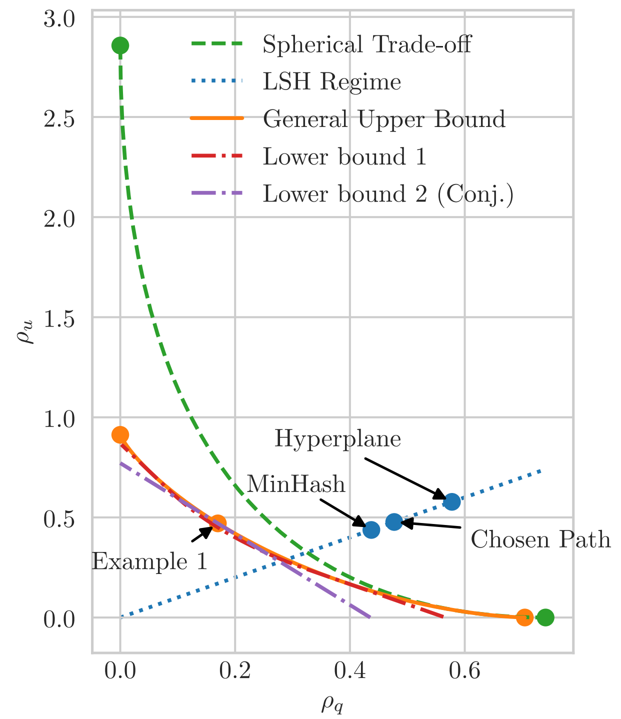
(b) Our decoding algorithm run in parallel on two cohorts q and y. The bold lines illustrates paths considered by sets, while the dashed lines adorn paths only considered by only one of x or y. Here q has a higher threshold ($t_q = 2/3$) than y ($t_u = 1/2$), so q only considers paths starting with two favourable representatives.

Figure 1: The representative sets, coloured in red, are scattered in the universe to provide an efficient space partition for the data.

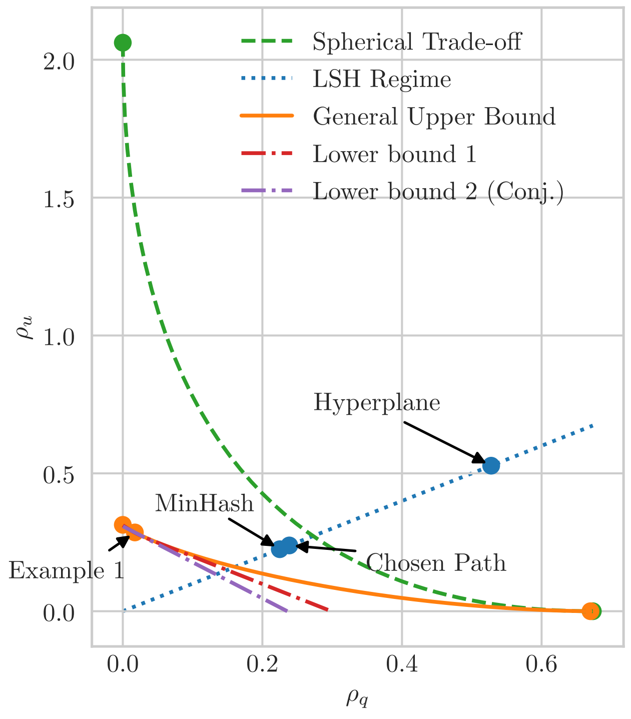
cohort $y \in Y$ if a Supermajority [2] of r is subsumed by the y – that is $|r \cap y|/k \geq t_u$ for some $t_u \in [0, 1]$ which is (usually) greater than w_u (the expected size of the overlap). We similarly pick a value $t_q \in [0, 1]$; and a query $q \in \binom{U}{w_q|U|}$ is compared against all cohorts y favoured by representatives s which also t_q-favour q (that is $|r \cap q|/k \geq t_q$). The intuition is that it is unlikely for a representative to favour a given cohort, and it is even more unlikely for it to favour two given cohorts, so if it does, it's probably because the cohorts are mostly the same (they have a big overlap). Figure 1a has a simple illustration of this principle.

The challenge of turning the abstract algorithm into a real algorithm is to design an oracle able to efficiently yield all such representatives in favour of a given cohort. We augment the above sampling procedure of representatives by carefully adding just enough structure that this is possible without destroying the properties of the algorithm. Instead of independent sampling, we sample a random height k tree of elements from the universe and each path from the root to a leaf is defined to be a representative set. Hence some sets share a common prefix, but mostly they are still independent. To make "decoding" a cohort efficient, we add the extra constraint that each of the prefixes of a representative has to be close to being in favour, rather than only having this requirement on the total set. This allows us to prune the tree during the decoding so we only spend time on representative sets that end up being in favour of our cohort, and only weakens the geometric properties negligibly. Figure 1b has a simple illustration of this algorithm and Algorithm 1 has pseudo-code.

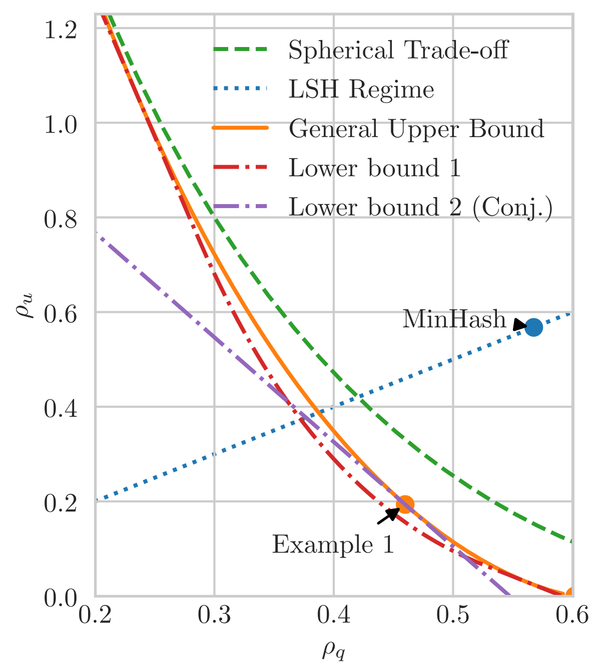
Previous LSF algorithms [23, 8] have used trees in their decoding algorithms, but they have had the advantage that each level of the tree was independent. Hence they were basically forming representatives as a Cartesian product of smaller promising representatives. We do not have this property, so we have to intimately study the "branching random walk", which is a main technical challenge in the paper.



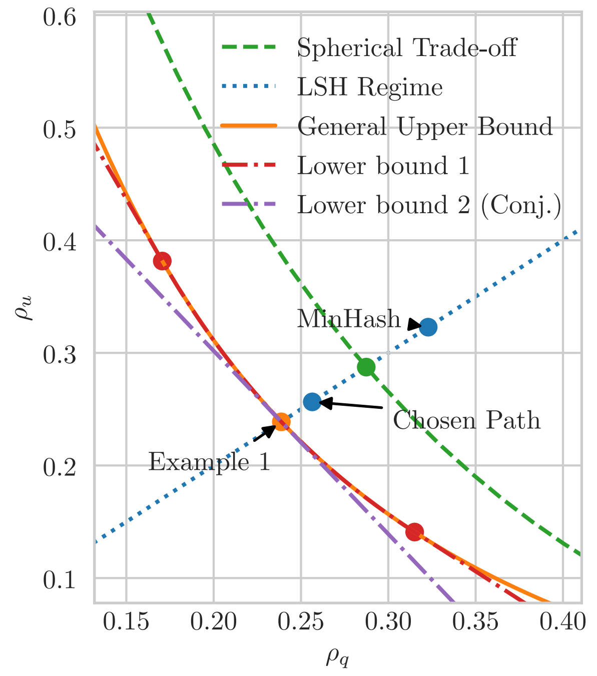
(a) Superset queries with $w_q = 0.1$, $w_u = 0.3$, $w_1 = 0.1$ and $w_2 = w_q w_u$. As the sets are relatively large, Spherical LSH beats MinHash and Chosen Path.



(b) Superset queries at smaller with smaller sets $w_q = 0.01$, $w_u = 0.03$, $w_1 = 0.01$ and $w_2 = w_q w_u$. Here MinHash does better compared to Spherical.



(c) Zoom in on $w_q = 0.4$, $w_u = 0.1$ $w_1 = 0.1$ $w_2 = w_q w_u$. We see that Theorem 3 is not tight when $w_q \neq w_u$. However, the conjectured Lower bound 2 matches the upper bound exactly.



(d) Zoom in on a "regular" example $w_q = w_u = 0.16$ with $w_1 = 0.1$, $w_2 = w_q w_u$. The red dots show the segment within where r, s can be chosen optimally without exceeding $1/2$.

Figure 2: Examples of ρ-values obtained from Theorem 1 for various parameter settings, compared to that of other algorithms, and to the lower bounds Theorem 3 and Conjecture 1. See the appendix Figure 4 for more comparisons.

1.1 Upper and Lower bounds

Below we first describe the time and space complexity of our algorithm, and then give matching lower bounds on most of the parameter space. In order to describe the results, we need to introduce a bit of notation regarding the sharp 2d-Chernoff Bound. This may be surprising, but our lower bounds show that this is indeed the right bound. Finally we present some results directly comparing Supermajorities to the MinHash, Chosen Path and Spherical LSH.

Chernoff bounds in $\mathbb{R}$ are well known, and for iid. $X_i \sim \text{Bernoulli}(p) \in \{0,1\}$ the sharpest form uses the binary KL-Divergence $D(t \mid p) = t \log \frac{t}{p} + (1-t)\log\frac{1-t}{1-p}$ for $t \geq p$ to give $\Pr[\sum X_i \geq tn] \leq \exp(-nD(t \mid p))$ with a matching lower bound. The inequality $D(p+\varepsilon \mid p) \geq 2\varepsilon^2$ gives a special case of Hoeffding's inequality. The Chernoff bound for $\mathbb{R}^2$ is a bit less known, but can likewise be described in terms of the KL-Divergence between two discrete distributions: $D(P \mid Q) = \sum_i p_i \log(p_i/q_i)$. In particular define $P_1 = \begin{bmatrix} w_1 & w_q - w_1 \\ w_u - w_1 & 1-w_q-w_u+w_1 \end{bmatrix}$ and let $X_i \sim B(P_1)$ be iid. $\in \{0,1\}^2$, then $\Pr[\sum X_i \geq \binom{t_q}{t_u} n] \approx \exp(-nD(T_1 \mid P_1))$ where $T_1 = \begin{bmatrix} t_1 & t_q - t_1 \\ t_u - t_1 & 1-t_q-t_u+t_1 \end{bmatrix}$ and $t_1 \in [0, \min\{t_u, t_q\}]$ minimizes $D(T_1 \mid P_1)$. (We say $x \geq y$ for $x, y \in \mathbb{R}^2$ if $x_1 \geq y_1$ and $x_2 \geq y_2$.)

We can now present the general performance bound, after which we will present examples of the performance it yields in more specific situations.

Theorem 1 (Informal Upper Bound). *For any choice of constants $w_q, w_u \geq w_1 \geq w_2 \geq 0$ and $1 \geq t_q, t_u \geq 0$ we can solve the (w_q, w_u, w_1, w_2)-GapSS problem over universe U with query time $\tilde{O}(n^{\rho_q}) + n^{o(1)}$ and space usage $\tilde{O}(n^{1+\rho_u}) + n^{o(1)}$ plus the size of storing the dataset. where*

$$\rho_q = \frac{D(T_1 \mid P_1) - D(t_q \mid w_q)}{D(T_2 \mid P_2) - D(t_q \mid w_q)}, \quad \rho_u = \frac{D(T_1 \mid P_1) - D(t_u \mid w_u)}{D(T_2 \mid P_2) - D(t_q \mid w_q)}.$$

Here P_2 and T_2 are like P_1 and T_1 but with w_1 and t_1 substituted for w_2 and $t_2 = \arg\inf D(T_2 \mid P_2)$. Note that the two bounds differ only in the $D(t_q \mid w_q)$ vs $D(t_u \mid w_u)$ terms in the denominator.

Varying t_q and t_u allows a full space/time trade-off with $\rho_q = 0$ in one end and $\rho_u = 0$ (and $\rho_q < 1$) in the other. Lagrange's condition $\nabla_t \rho_q = \lambda \nabla_t \rho_u$ provides for easy optimization over (t_q, t_u). For convenience we provide Figure 2 to guide the intuition, as well as some corollaries with results for interesting parametrizations.

We note that from the way the (w_q, w_u, w_1, w_2)-GapSS problem is defined, the parameters never depend on n or $|U|$. The full version of our theorem however also covers these cases and give a precise bound on ρ_q and ρ_u even for very small sets where $w_q|U|$ and $w_u|U|$ are constants.

Example 1: Near balanced ρ values. Setting $t_q = 1 - w_u, t_u = 1 - w_q$ always gives an optimal point on the trade-off with ρ values

$$\rho_q = \frac{H(w_1) - D(w_u \mid 1 - w_q)}{H(w_2) - D(w_u \mid 1 - w_q)}, \quad \rho_u = \frac{H(w_1) - D(w_q \mid 1 - w_u)}{H(w_2) - D(w_u \mid 1 - w_q)},$$

where $H(w_i) = (1 - w_q - w_u)\log(\frac{1-w_q-w_u+w_i}{w_i})$. This point is special because the optimal values of t_u and t_q depend only on w_u and w_q, while in general they will also depend on w_1 and w_2. In the case $w_q = w_u = w$ this simplifies further to $\rho_q = \rho_u = \log(\frac{w_1}{w} / \frac{1-w}{1-2w+w_1}) / \log(\frac{w_2}{w} / \frac{1-w}{1-2w+w_2})$ in which case it is finally simple to compare with Chosen Path's ρ value of $\log(\frac{w_1}{w}) \big/ \log(\frac{w_2}{w})$. Chosen Path on balanced sets was shown in [23] to be optimal for w, w_1, w_2 small enough, and we see that Super Majorities do indeed recover this value for that range.

[2]A Supermajority or a qualified majority is a requirement for a proposal to gain a specified greater level of support than a simple majority.

Example 2: Subset/superset queries. If $w_1 = \min\{w_u, w_q\}$ and $w_2 = w_u w_q$ we can take $t_q = -\frac{w_u(1-w_u)w_q(1-w_q)}{w_q-w_u}\alpha + \frac{w_q(1-w_u)}{w_q-w_u}$ and $t_u = \frac{1}{w_q-w_u}\alpha^{-1} - \frac{w_u(1-w_q)}{w_q-w_u}$ for any $\alpha \in [w_1 - w_q w_u, \max\{w_u, w_q\} - w_q w_u]$. Theorem 1 then gives data structures with

$$\rho_q = \frac{t_q \log\frac{1-t_u}{1-w_u} - t_u \log\frac{1-t_q}{1-w_q}}{D(t_u, w_u)} \qquad \rho_u = \frac{(1-t_u)\log\frac{t_q}{w_q} - (1-t_q)\log\frac{t_u}{w_u}}{D(t_u, w_u)} \qquad \text{if } w_1 = w_u,$$

$$\rho_q = \frac{-(1-t_u)\log\frac{t_q}{w_q} + (1-t_q)\log\frac{t_u}{w_u}}{D(t_u, w_u)} \qquad \rho_u = \frac{-t_q \log\frac{1-t_u}{1-w_u} + t_u \log\frac{1-t_q}{1-w_q}}{D(t_u, w_u)} \qquad \text{if } w_1 = w_q.$$

This represents one of the cases where we can solve the Lagrangian equation to get a complete characterization of the t_q, t_u values that give the optimal trade-offs. Note that when $w_1 = w_u$ or w_q, the P matrix as used in the theorem has 0's in it. The only way the KL-divergence $D(T \mid P)$ can then be finite is by having the corresponding elements of T be 0 and use the fact that $0\log(0/q)$ is defined to be 0 in this context.

Example 3: Linear space/constant time. Setting respectively $t_1/w_1 = (t_q - t_1)/(w_q - w_1)$ or $t_1/w_1 = (t_u - t_1)/(w_u - w_1)$ we get $D(T_1 \mid P_1) = D(t_q \mid w_q)$ or $D(T_1 \mid P_1) = D(t_u \mid w_u)$. Theorem 1 then yields algorithms with either $\rho_q = 0$ or $\rho_u = 0$ corresponding to respectively a data structure with $\tilde{O}(1)$ query time, or $\tilde{O}(n)$ space.

Lower bounds Because the general upper bound is somewhat more complicated than previous results, tight lower bounds are an important part of our article, showing that Super Majorities are "the right" approach to data structures for sets.

We split our lower bounds in two types: (1) Bounds that require $w_2 = w_q w_u$ and (2) those that don't. While the second type is more general, our results of the first type stronger. In particular, the type 1 lower bounds also holds for so called *data dependent* LSH data structures, which is a much stronger and extremely successful model for similarity search. This shows that even using data dependence our algorithm can't be improved on random data. Better still, our bounds of this type also give 1-probe lower bounds for cell-probe data structures. This means that our memory requirement, $n^{1+\rho_u}$, is optimal in the case $\rho_q = 0$ and no data structure using size $n^{o(1)}$ memory cells can use less memory while answering a query in a single probe.

The type 2 lower bound is however more simple to state, informally:

Theorem 2 (Lower bound 3, Informal)**.** *If $w_q = w_u = w$ and $\rho_u = \rho_q = \rho$, any LSF data structure must use space $n^{1+\rho}$ and have query time n^ρ where $\rho \ge \log(\frac{w_1-w^2}{w(1-w)})/\log(\frac{w_2-w^2}{w(1-w)})$.*

Note the similarity with our upper bound in the same regime of $\rho = \log(\frac{w_1(1-2w+w_1)}{w(1-w)})/\log(\frac{w_2(1-2w+w_2)}{w(1-w)})$ in Example 1. As expected they match for $w, w_1, w_2 \to 0$, and curiously also for $w_1 \to w$ since $w_1(1-2w+w_1) = w_1 - w^2 + (w - w_1)^2$. However in contrast to the previous bounds of this type [40, 23] our lower bound holds generally rather than just asymptotically.

Our lower bounds

Theorem 3 (Lower bound 1)**.** *Given $\alpha \ge 0$ and $0 \le r, s \le 1/2$, let $u_q = \log\frac{1-w_q}{w_q}$, $u_u = \log\frac{1-w_u}{w_u}$, If r and s are such that $\sqrt{\frac{\sinh(u_q(1-1/r))}{\sinh(u_q/r)}\frac{\sinh(u_u(1-1/s))}{\sinh(u_u/s)}} = \frac{w_1 - w_q w_u}{\sqrt{w_q(1-w_q)w_u(1-w_u)}}$, then any list-of-points data structure must use space $n^{1+\rho_u}$ and have query time n^{ρ_q} where $\rho_q \ge (1/r-1)\alpha + 1/s$ or $\rho_u \ge \alpha/r + 1/s - 1$,* For $w_u = w_q$ we can take $\alpha = 1$ and $r = s$ to get $\rho_q, \rho_u \ge \log \frac{w_q}{w_1} \frac{1-2w_q+w_1}{1-w_q} \big/ \log \frac{1-w_q}{w_q}$ which exactly matches our upper bound in the $w_2 = w_q w_u$ case. It is not immediately obvious, but this bound shows that *Theorem 1 is optimal when $w_q = w_u$* for nearly the entire space/time trade-off where the requirement $r, s \le 1/2$ doesn't stop us. Again we refer Figure 2 for intuition.

The theorem, based on a hypercontractive inequality by Oleszkiewicz [39], also gives a strong bound when $w_u \ne w_q$, but to get a completely tight match with our upper bound, we have to conjecture the simple inequality that for any $a, b \in \mathbb{R}$:

$$(abw_1 + a(w_q - w_1) + b(w_u - w_1) + (1 - w_q - w_u + w_1))^s \le (w_q a^s + (1 - w_q))(w_u b^s + (1 - w_u)), \quad (1)$$

where $s = \log \frac{(1-w_q)(1-w_u)}{w_q w_u} \big/ \log \frac{1-w_q-w_u+w}{w}$. (If $w_1 \le w_q w_u$ the inequality goes the other direction.) The state of the art in this regime is [49] which shows similar inequalities tight up to constants. However equations as strong as eq. (1) is a big open problem. Yet, assuming this inequality we can prove the following lower bound:

Conjecture 1 (Lower bound 2)**.** *Let $\alpha \ge 0$ then any LSF data structure must use space $n^{1+\rho_u}$ and have query time n^{ρ_q} where $\rho_q \ge (\alpha + 1)/s - \alpha \quad$ or $\quad \rho_u \ge (\alpha + 1)/s - 1$.*

Setting $\alpha = D(w_u \mid 1 - w_q)/D(w_q \mid 1 - w_u)$ gives exactly the bounds stated in Example 1 of $\rho_q = \frac{H(w_1)-D(w_u|1-w_q)}{D(w_q,1-w_u)}$ and $\rho_u = \frac{H(w_1)-D(w_q|1-w_u)}{D(w_q,1-w_u)}$ for $w_2 = w_q w_u$. Showing that this is indeed optimal for all choices of w_1, w_q and w_u.

Comparison to previous approaches Since our lower bounds don't cover the entire range of parameters w_q, w_u, w_1, w_2 (no LSH lower bounds do), we need to compare our ρ values with those achieved by previous methods and show that we get lower values on the entire range.

The approach bears similarity to Chosen Path [23], and indeed the algorithms are nearly identical when in the case $t_q = t_u = 1$. For this reason it is clear that we can always get at least as good an exponent as Chosen Path, though in the case where $w_q = w_u$ and $w_q, w_u \to 0$ the results coincide.

We show two more results of this sort: (1) For Spherical LSH we show how to most optimally embed GapSS onto the sphere, and that the resulting exponent is equivalent to the case of Supermajorities $t_q = w_q + o(1)$, $t_u = w_u + o(1)$ — the representative should favour the cohort just slightly above the expectation. (2) We show MinHash can be seen as an average of a family of Chosen Path like algorithms, that an average is always worse than simply using the best, and that each algorithm has a worse exponent than Supermajorities.

The first result is quite interesting on its own right, since many previous algorithms for Maximum Inner Product Search consisted of various embeddings onto the sphere. In particular wee show, that if an algorithm has exponent $\rho(\alpha, \beta) = f(\alpha)/f(\beta)$ where α is the cosine similarity between good points and β is the similarity between bad points on the sphere; then assuming some light properties on f, which contain both Spherical and Hyperplane LSH, two affine embedding of sets $x \in \{0,1\}^d$ to S^{d-1} that minimizes ρ once the new cosine similarities are calculated, is $x \mapsto (x - w)/\sqrt{w(1-w)}$ where $w = |x|/d$. While the mapping is allowed to depend on any of the GapSS parameters, it curiously only cares about the weight of the set itself. For fairness, all our plots, such as Figure 2, uses this embedding when comparing Super Majority to Spherical LSH.

For result (2), consider the classical MinHash scheme: A permutation $h : [d] \to [d]$ is sampled at random, and $y \subseteq \{0,1\}^d$ is placed in bucket $i \in [m]$ if $h(i) \in y$ and $\forall_{j<i} h(j) \notin y$. The probability for a collision between two sets q, y is then $|q \cap y|/(|q| + |y| - |q \cap y|)$ by a standard argument which implies an exponent of $\rho_{\mathrm{mh}} = \log \frac{w_1}{w_q+w_u-w_1} \big/ \log \frac{w_2}{w_q+w_u-w_2}$. Now consider building

multiple such MinHash tables, but keeping only the kth bucket in each one. This gives a LSF scheme, which combined with a symmetrization technique implicit in [23] has exponent $\rho_k = \log \frac{(1-w_q-w_u+w_1)^k w_1}{\max\{(1-w_q)^k w_q,(1-w_u)^k w_u\}} \big/ \log \frac{(1-w_q-w_u+w_2)^k w_2}{\max\{(1-w_q)^k w_q,(1-w_u)^k w_u\}}$. We show in Appendix B.2 that $\rho_{\text{mh}} \geq \min_{i \geq 0} \rho_i$. The point is, that once w_q, w_u, w_1 and w_2 are explicit, a simple variation of Chosen Path can be shown to consistently beat MinHash, which is an open question raised in [23] — and so Supermajorities always dominate MinHash.

1.2 Technical overview

The contributions of the paper are conceptual as well as technical. To a large part, what enables tight upper and lower parts is defining the right problem to study and the geometry to apply. However, even once that is settled a number of tricky algorithmic problems arise, requiring deep mathematical analysis of branching random walks of exponentially twisted variables.

Where do the improvements come from? The first question to ask is why there is even a difference between LSH schemes for Boolean data and for Spherical data. This is perhaps best understood by considering a figure like Figure 5a. When the Jaccard similarity between good sets go to 0 as $m^{-1/2}$ and between bad sets as m^{-1} the Boolean methods like MinHash and Super Majorities still get a ρ value below 1. However as the angles, α, β between good and bad points, go to zero the exponent $\rho \approx \frac{1-2\alpha}{1+2\beta}$ of Spherical LSH always goes to 1.

This has to do with the bounded and positive nature of the Set Similarity problem. A filter (or hash function or representative set) can get a lot of information about the overlap of two sets by considering only a few coordinates. If two sets both contain k out of k randomly sampled universe elements, it gives a huge amount of information. This is essentially the observation of Chosen Path [23]. However requiring k out of k means that large sets will be captured much more often than smaller sets, leading to imbalanced ρ values between queries and updates. This was attempted smoothened out in the paper, but not optimally so. Our insight is essentially finding the optimal way to balance the information gained between sets of different sizes.

Many other methods could (and have been tried): (1) Using thresholds with sparse Gaussian, ± 1, or exponential vectors. (2) Using asymmetric threshold functions like $\sum_k \alpha^k x_k \geq 1$. (3) Using polynomial thresholds. (4) Other hashing based methods in the style of MinHash. As it turns out all of these methods give suboptimal exponents.

Techniques and analysis Once we have decided on a geometry for our space partitions, the obviously important question is how to decode them efficiently. In order to extract as much information as possible, each representative sets has a very small chance of favouring a particular set, and iterating through them all would take more than $n^{\log n}$ comparisons. We solve this by adding a carefully balanced amount of structure to the random sampling process. Namely, we change build representatives using a 2d-branching random walk. This allows the decoder to prune the paths that are unlikely to ever accumulate enough votes in favour of a cohort.

A pruned branching random walk on the real line can be described in the following way. An initial ancestor is created with value 0 and form the zeroth generation. The people in the ith generation give birth Δ times each and independently of one another to form the $(i+1)$th generation. The people in the $(i+1)$th generation inherit the value, v, of their parent plus an independent random variable X. If ever $v + X < 0$, the child doesn't survive. After k generations, we expect by linearity $n^k \Pr[\forall_{i \leq k} \sum_{j \in [i]} X_i \geq 0]$ people to be alive, where X_i are iid. random variables as used in the branching. A pruned 2d-branching random walk is simply one using values $\in \mathbb{R}^2$.

Previous LSF algorithms have used trees to efficiently prune the decoding process [23, 8]. They use hash functions to partition vectors in Y in a tree structure, where at every level ℓ they sample a number of Δ such functions and repeat this for k levels. To answer a given query the algorithm performs a linear scan through the points stored at the leaves that capture the query. If at any level a vector is not captured by the space partition, it is pruned. This means that these algorithms have $\Delta^k p^k$ expected survivors, where p is the probability of survival at a particular level. Unfortunately there are no probability distribution such that this form coincides with what we are after.

Branching random walks are usually analysed using the second moment method. However as noted by Bramson [13]: "an immediate frontal assault using moment estimates, but ignoring the branching structure of the process, will fail." The issue is that the probability that a given pair of paths in the branching process survives is too large. If the shared part of the paths manage to accumulate much more than the expected value, its children will have a much too high chance of surviving. For this reason we have to *counterintuitively add extra pruning when proving the lower bound* that a representative set survives. More precisely, we prune all the paths that accumulate much more than the expected value. We show that this does not lower the probability that a representative set is favour by much, while simultaneously decreasing the variance of the branching random walk a lot. Unfortunately, this adds further complications since ideally we would prune every path that gets below the expectation, but then our branching random walk would have contained in a too narrow band for a sufficient number of paths to survive. Hence instead, we allow the paths deviate by roughly a standard deviation below the expectation.

We also employ a number of other analytical tricks. Where standard Chernoff bounds as employed in the community are usually not even tight up to a constant in the exponent, we need bounds that are *tight up to polynomial factors*. For this, we employ an exponential change of measure, which separates the large deviation part of our analysis from a mean 0 random branching process. This process is then analysed using martingales.

Lower bound techniques For the lower bounds we will assume $|U| = \omega(\log n)$ (like we reduce to in the upper bounds). This follows all previous LSH lower bounds, and it is indeed known to be necessary from [12, 16] which breaks the known lower bounds in the "medium dimension regime" when $|U| = O(\log n)$. Our lower bounds are based on the analysis of Boolean functions (e.g. [38]) and in particular certain inequalities bounding the amount of correlation possible between them. Using frameworks by [43] and [7] this enables us to show lower bounds for respectively 1-cell probe data structures and a model known as "list of points" data structures, which include all LSH algorithms, including data dependent ones.

The main difference compared to previous bounds is that we study Boolean functions on so called p-biased spaces. This turns out to be the exact right model for GapSS as the tightness with our upper bound shows. Much less is known about these spaces however, with [39] and [49] being state of the art.

We give three lower bounds, which cover different ranges of the (w_q, w_u, w_1, w_2)-GapSS parameter range. In Figure 2 two of them are plotted against our upper bound. The first bound is based on a reduction from two correlated functions to one, which is covered by [39]. This turns out to be optimal for general $w_q = w_u = w$, generalizing previous bounds which required $w = 1/2$. For $w_q \neq w_u$ it appears that a completely new hypercontractive inequality is needed, and we show that conditioned on a simple conjecture eq. (1), one can show such a result which matches our general upper bound. Finally we give a lower bound based directly on the power means inequality, which bears some resemblance to the first optimal data independent LSH lower bound by O'Donnell et al. [40]. However where those bounds were asymptotic, ours holds generally. This bound only holds in a slightly stronger model than the two first, but it is the only one that covers the case $w_2 \neq w_q w_u$.

1.3 Related Work

Work on Set Similarity Search has focused on a number of seemingly disparate problems, traditionally been studied in their exact form:

Super-/Subset queries (SQ) Pre-process a database D of n points in $\{0,1\}^d$ such that, for all query of the form $q \in \{0,1\}^d$, either report a point $x \in D$ such that $x \subseteq q$ (resp. $q \subseteq x$) or report that no such x exists.

Partial Match (PM) Pre-process a database D of n points in $\{0,1\}^d$ such that, for all query of the form $q \in \{0,1,*\}^d$, either report a point $x \in D$ matching all non-$*$ characters in q or report that no such x exists.

Similarity Search (SS) Given a similarity measure $S : \{0,1\}^d \times \{0,1\}^d \to [0,1]$, pre-process a database D of n points in $\{0,1\}^d$ such that, for all query of the form $q \in \{0,1\}^d$ return the point $x \in D$ maximizing $S(q,x)$.

Maximum Inner Product Search (MIPS) Same as Similarity Search, but $S(x,y) = \langle x, y\rangle$ — the euclidean inner product.

These problems are all part of an equivalence class of hard problems, known as Orthogonal Vectors [19]. This means that we don't expect the existence of polynomial space data structures that can solve either of these problems faster than a linear scan through the entire database. See also [3, 1, 46]. For this reason people have studied approximate versions of each problem. While the exact definition of the approximation differs in the literature, once we fix the weight of the input vectors, they all become essentially equal to GapSS as defined in this paper. This allows us to compare the best algorithms from each category against each other, as well as against our suggested Supermajorities algorithm. It should be noted that the hardness results mentioned above also holds for approximate variations, so the gap will have to be sufficiently large for any approach to work.

Partial Match The problem is equivalent to the subset query problem by the following well known reductions: (PM $\to$ SQ) Replace each $x \in D$ by the set $\{(i, p_i) : i \in [d]\}$. Then replace each query q by $\{(i, q_i) : q_i = *\}$. (SQ $\to$ PM) Keep the sets in the database as vectors and replace in each query each 0 by an $*$.

The classic approach, studied by Rivest [45], is to split up database strings like `supermajority` and file them under `s`, `u`, `p` etc. Then when given query like `set` we take the intersection of the lists `s`, `e`, `t`. Sometimes this can be done faster than brute force searching each list. He also considered the space heavy solution of storing all subsets, and showed that that when $d \le 2\log n$, the trivial space bound of 2^d can be somewhat improved. Rivest finally studied approaches based on tries and in particular the case where most of the database was random strings. The later case is in some ways similar to the LSH based methods we will describe below.

Indyk, Charikar and Panigrahi [17] also studied the exact version of the problem, and gave, for each $c \in [n]$, an algorithm with $O(n/2^c)$ time and $n2^{(O(d\log^2 d\sqrt{c/\log n})}$ space, and another with $O(dn/c)$ query time and nd^c space. Their approach was a mix between the shingling method of Rivest, building a look-up table of size $\approx 2^{\Omega(d)}$, and a brute force search. These bounds manage to be non-trivial for $d = \omega(\log n)$, however only slightly. (e.g. $n/\operatorname{poly}(\log n)$ time with polynomial space.)

There has also been a large number of practical papers written about Partial Match / Subset queries or the equivalent batch problem of subset joins [44, 35, 27, 2, 25]. Most of these use similar methods to the above, but save time and space in various places by using bloom filters and sketches such as MinHash [15] and HyperLogLog [26].

Maximum Inner Product For exact algorithms, most work has been done in the batch version (n data points, n queries). Here Alman et al. [4] gave an $n^{2-1/\tilde{O}(\sqrt{k})}$ algorithm, when $d = k \log n$.

An approximative version can be defined as: Given $c > 1$, pre-process a database D of n points in $\{0,1\}^d$ such that, for all query of the form $q \in \{0,1\}^d$ return a point $x \in D$ such that $\langle q, x\rangle \geq \frac{1}{c} \max_{x' \in D} \langle q, x'\rangle$. Here [3] gives a data structure with query time $\approx \tilde{O}(n/c^2)$, and [19] solves the batch problem in time $n^{2-1/O(\log c)}$ (both when d is $n^{o(1)}$.)

There are a large number of practical papers on this problem as well. Many are based on the Locality Sensitive Hashing framework (discussed below) and have names such as SIMPLE-LSH [37] and L2-ALSH [47]. The main problem for these algorithms is usually that no hash family of functions $h : \{0,1\}^d \times \{0,1\}^d \to [m]$ such that $\Pr[h(q) = h(x)] = \langle q, x\rangle / d$ [3] and various embeddings and asymmetries are suggested as solutions.

The state of the art is a paper from NeurIPS 2018 [50] which suggests partitioning data by the vector norm, such that the inner product can be more easily estimated by LSH-able similarities such as Jaccard. This is curiously very similar to what we suggest in this paper.

We will not discuss these approaches further since, for GapSS, they all have higher exponents than the three LSH approaches we study next.

Similarity Search The problem is usually studied as an approximate problem: Given a similarity measure $S : \{0,1\}^d \times \{0,1\}^d \to [0,1]$ and $s_1 > s_2 \in [0,1]$, pre-process a database D of n points in $\{0,1\}^d$ such that for queries $q \in \{0,1\}^d$ we return a point $x \in D$ with $S(q,x) \geq s_2$ given there is $x' \in D$ with $S(q, x') \geq s_1$.

This naturally generalizes MIPS as defined above. The formulation allows the use of Indyk and Motwani's LSH framework [29]. Here we define a family, $\mathcal{H}$, of functions $h : \{0,1\}^d \times \{0,1\}^d \to [m]$ such that

1. $\Pr_{h\sim\mathcal{H}}[h(q) = h(x)] \geq p_1$ when $S(q,x) \geq s_1$, and
2. $\Pr_{h\sim\mathcal{H}}[h(q) = h(x)] < p_2$ when $S(q,x) < s_2$.

The constructions in [29, 28] then give an algorithm for the approximate similarity search problem with space $n^{1+\rho} + dn$ and query time dominated by n^ρ evaluations of h, where $\rho = \log p_1 / \log p_2$.

If $\mathcal{H}$ exists such that $\Pr_{h\sim\mathcal{H}}[h(q) = h(x)] = S(q,x)$ is achievable (see [20] for a study of when this is the case) then such a family is an obvious choice. An example of this is Broder's MinHash algorithm, which has $\Pr_{h\sim\mathcal{H}}[h(q) = h(x)] = |q \cap x|/|q \cup x|$ where $S(q,x) = |q \cap x|/|q \cup x|$ is the Jaccard similarity.

Choosing $\mathcal{H}$ like this is however not always optimal, as Christiani and Pagh [23] shows by constructing a data structure with $\rho = \frac{\log 2s_1/(1+s_1)}{\log 2s_2/(1+s_2)} < \frac{\log s_1}{\log s_2}$ when the size of sets is equal, $|q| = |x|$. In general they get $\rho = \frac{\log b_1}{\log b_2}$ where $b_1 > b_2 \in [0,1]$ are Blanquet Similarities $B(q,x) = |q \cap x| / \max\{|q|, |x|\}$.

The most studied variant is LSH on the sphere. Here, given $\alpha > \beta \in [-1, 1]$, we pre-process a database D of n points in S^{d-1} and for a query $q \in S^{d-1}$ return $x' \in D$ with $\langle q, x'\rangle \geq \beta$ given the promise that there is $x \in D$ with $\langle q, x\rangle \geq \alpha$. In [5] they show how to get $\rho_{\text{sp}} = \frac{1-\alpha}{1+\alpha}\frac{1+\beta}{1-\beta}$.[3]

While it is clear that both MinHash and Chosen Path can solve GapSS when w_q and w_u is known in advance, using spherical LSH requires that we embed the binary vectors onto the sphere. Multiple ways come to mind, such as mapping $\{0 \mapsto -1/\sqrt{d}, 1 \mapsto 1/\sqrt{d}\}$ or $\{0 \mapsto 0,\ 1 \mapsto 1/\sqrt{w_q d}$ (for queries, resp. $1/\sqrt{w_u d}$ for data points)$\}$. We study the optimal such embedding in lemma B.1.

Two other classic methods are Bit Sampling [29] and SimHash (Hyperplane rounding) [18], which give $\rho_{\text{bs}} = \frac{\log(1-w_q-w_u+2w_1)}{\log(1-w_q-w_u+2w_2)}$ and $\rho_{\text{hp}} = \frac{\log(1-\arccos(\alpha)/\pi)}{\log(1-\arccos(\beta)/\pi)}$ respectively. (SimHash also works on the sphere, but has the same optimal embedding as spherical LSH.) These ρ-values however turn out to always be larger than ρ_{sp}, so we won't study them as much.

While Chosen Path and Spherical LSH both have proofs of optimality [23, 8, 40, 36] in the LSH model, these optimality proofs consider specific ranges, like when w_q, w_u or w_1 goes to zero. Hence they are not necessarily optimal when used in all the ranges of parameters in which GapSS is interesting. In fact they each have regions of optimality, as was observed in [23] who proposed as an open problem to find an LSF scheme that unified all of the above. This is what we do in this paper, as well as showing matching lower bounds in a wider range of parameters.

Trade-offs and Data Dependency The above algorithms, based on the LSH framework, all had space usage roughly $n^{1+\rho}$ and query time n^{ρ} for the same constant ρ. This is known as the "balanced regime" or the "LSH regime". Time/space trade-offs are important, since $n^{1+\rho}$ can sometimes be too much space, even for relatively small ρ. Early work on this was done by Panigrahy [42] and Kapralov [32] who gave smooth trade-offs ranging from space $n^{1+o(1)}$ to query time $n^{o(1)}$. A breakthrough was the use of LSF, which allowed time/space trade-offs with sublinear query time even for near linear space and small approximation [34, 22, 8].

Prior to this article, the only way to achieve trade-offs for set data was to embed it into the above spherical algorithms. In this paper we show that it is often possible to do much better, and in some cases get query time less than that of balanced spherical LSH, even with near-linear space.

Arguably the largest break-through in LSH based data structures was the introduction of data dependent LSH[6, 9, 10]. It was shown how to reduce the general case of α, β similarity search as described above, to the case $\beta = 0$ (and $\alpha \mapsto \frac{\alpha-\beta}{1-\beta}$), in which many LSH schemes work better. Using those data structures on GapSS with $w_2 > w_q w_u$ will often yield better performance than the algorithms described in this paper. However, since the data dependent methods are equivalent to Spherical LSH for $w_2 = w_u w_q$, we always dominate this case, and it is an exciting open problem to create similar reductions directly for set data, possibly using the space partitioning proposed in this algorithm as a building block.

2 The Algorithm

We now describe the full algorithm that gives Theorem 1. We state the full version of the theorem, discuss it and prove it. The section ends with an involved analysis of the survival probabilities of the branching random walk.

The Real Algorithm Let $w_q, w_u, w_1, w_2, t_q, t_u \in [0, 1]$ be given. Assume $\min\{w_q, w_u\} \geq w_1 > w_2$ and $t_q \neq w_q, t_u \neq w_u$. Also assume the existence of $k \in \mathbb{Z}_+$, $\Delta \in \mathbb{R}_+$ and the sequence $(c_\ell)_{\ell\in[k]}$,

[3]For $\beta \to 1$ this approaches $\log\alpha/\log\beta$, which would be like an LSH-able family for inner product on the sphere, but unfortunately this is not achievable with LSH. For the batch problem it was shown possible in [33].

which we will define later. For some universe, U, we are given at family $Y \subseteq \binom{U}{w_u|U|}$ of size $|Y| = n$. We assume $|U| = q$ where q is some prime number. This can always be achieved by adding at most $|U|^{0.525}$ large enough $|U|$ [11][4] Hence we only distort each of w_q, w_u, w_1, w_2 by roughly a factor $1 + 1/O(|U|^{1/2})$, which is insignificant for $|U| = \Omega(\log n)^2$, and we can always increase $|U|$ without changing the problem parameters by duplicating the sets.

Now define $h_i(r) : [q]^i \to [q]$ by $h_i(r) = \sum_j a_{i,j} r_j + b_i \mod q$ for some sequences of random numbers $a_{i,j} \in [q] \setminus \{0\}, b_i \in [q]$. Note that each h_i is a 2-independent random function such that $\Pr[h_i(r) = h_i(r')] \le 1/q$ for $r \neq r'$. Define the sequence $(\Delta_i \in \mathbb{Z}_+)_{i \in [k]}$ such that $\prod_{j=1}^{i} \Delta_j \le \Delta^i \le 2 \prod_{j=1}^{i}$. This is always possible by Lemma D.4.

We can now define the set $R_i = \{r \circ x \in R_{i-1} \times U \mid h_i(r \circ x) < \Delta_i\}$, where the notation $r \circ x$ is string concatenation, as well as the decoding function $R_i(X, t) : \mathcal{P}(U) \times [0, 1] \to R_i$

$$R_i(X,t) = \left\{ r \in R_i \;\middle|\; \forall \ell \le i : \sum_{i \in [\ell]} [r_i \in X] \ge t\ell - c_\ell \right\}$$

The decoding function is also illustrated below as pseudo-code in Algorithm 1.

Our data structure now builds a hashmap M of lists of pointers and store each set $y \in Y$ in $M[r]$ for every $r \in R_k(y, t_u)$. On a new query $q \in \binom{U}{w_q|U|}$ we look at every list $M[r]$ for $r \in R_k(q, t_q)$. For each y in such a list, we can determine in time $O(\min\{w_q, w_u\} w_2^{-1} \log n)$ whether the intersection is $> w_2|U|$ by sampling random elements from the smallest of q and y and checking if they are present in the other.

Algorithm 1: TreeSample

```
Input: Universe $U$, Set $X \subseteq U$, Threshold $t \in [0,1]$
Result: Set $P_k \subseteq U^k$ of paths
$R_0 \leftarrow \{((), 0)\}$                 // The $R_i$ values keep track of some intermediates.
for $i = 1$ to $k$ do
    $R_i \leftarrow \{\}$
    for $(r, s) \in R_{i-1}$ do
        for $x \in U$ st. $h_i(r \circ x) < \Delta_i$ do                // Sample the universe
            $s' \leftarrow s + [x \in X]$
            if $s' \ge it - c_i$ then                               // Trim to promising paths
                $R_i \leftarrow R_i \cup \{(r \circ x, s')\}$
            end
        end
    end
end
```

An optimization In the "Sample the universe" step of Algorithm 1 a naive implementation spends time $|X|$ hashing all possible elements. We now show how to make this step output sensitive, using only time equal to the number of values for which the condition is true. [5]

The requirement $s' \ge it - c_\ell$ we call the "trimming condition". This allows us to trim away most prefix paths which would be very unlikely to ever reach our requirement for the final path.

[4]And it is an open conjecture by Cramer that $(\log |U|)^2$ suffices as well.

[5]The subroutine is inspired by personal communications with Rasmus Pagh and Tobias Christiani.

To speed up finding all $x \in U$ such that $h_i(r \circ x) < \Delta_i$ we note that there are two cases relevant to the trimming condition, depending on s in the algorithm: (1) s' has to be $s+1$ or (2) $s' = s$ suffices. In the first case we are only interested in x values in X, while in the second case, all $x \in U$ values are relevant.

We have $h_i(r \circ x) = \eta + ax \mod q$ for some values η, a and b where $a > 0$. In case (2) the relevant x are simple $\{a^{-1}(v - \eta) \mod q \mid v \in [\Delta]\}$, where a^{-1} exists because q is prime. For the case (1) where x must be in X, we pre-process X by storing $ax \mod q$ for $x \in X$ in a sorted list. Using a single binary search, we can then find the relevant values with a time overhead of just $\lg |X|$. Using a more advanced predecessor data structure, this overhead can be reduced. See Algorithm 2 for a pseudocode version of this idea.

Algorithm 2: Output sensitive sample

```
Input       : $r \in [q]$, $\Delta \in [q]$
Pre-process: $s = \text{sorted}\{h(x) \mid x \in X\} \in [q]^{|X|}$ and $\kappa \in X^{|X|}$ st. $h(\kappa[i]) = s[i]$.
Result: $R = \{x \in X \mid (h(x) + r \mod q) < \Delta\}$
$i \leftarrow \min\{i \in [|X|] \mid s[i-1] < q - r \leq s[i]\}$        // We assume $s[i] = -\infty$ for $i < 0$
$R \leftarrow \{\}$
while $(s[i \mod |X|] + r \mod q) < \Delta$ do
    $R \leftarrow R \cup \{\kappa[i \mod |X|]\}$
    $i \leftarrow i + 1$
end
```

We now state the full version of Theorem 1 and a discussion of the differences between it and the idealized version in the introduction.

Theorem 4. *Let $w_q, w_u \geq w_1 \geq w_2 \geq 0$ be given with $w_1 \geq w_q w_u$ and $1 \leq t_q, t_u \leq 0$. Set k to be the smallest even integer greater than or equal to $\frac{\log n}{D(T_2|P_2) - D(t_q|w_q)}$ and assume that $t_q k/2$ and $t_u k/2$ are integers. The (w_q, w_u, w_1, w_2)-GapSS problem over a universe U can be solved with expected query time*

$$O(\varsigma_q\, k^{13}\, n^{\rho_q} + k w_q\, |U|) + \left(\tfrac{t_q(1-w_q)}{w_q(1-t_q)}\right)^{O(\sqrt{k \log k})} \quad \textit{and space usage} \quad O(\varsigma_u\, k^{13}\, n^{1+\rho_u} + n w_u\, |U|).$$

$$\textit{where} \quad \rho_q = \frac{D(T_1 \mid P_1) - D(t_q \mid w_q)}{D(T_2 \mid P_2) - D(t_q \mid w_q)} \quad \textit{and} \quad \rho_u = \frac{D(T_1 \mid P_1) - D(t_u \mid w_u)}{D(T_2 \mid P_2) - D(t_q \mid w_q)}.$$

and $\varsigma_q = \frac{w_q}{w_2} e^{2(D(T_1|P_1) - D(t_q|w_q))}$, $\varsigma_u = e^{2(D(T_1|P_1) - D(t_u|w_u))}$.

We stress that all previous Locality Sensitive algorithms with time/space trade-offs had $n^{o(1)}$ factors on n^{ρ_q} and n^{ρ_u}. These could be as large as $\exp(\sqrt{\log n})$ or even $\exp((\log n)/(\log \log n))$. In contrast, *our algorithm is the first that only loses $k \approx \log(n)$ multiplicative factors!* We also note that the constants of the size ς_q and ς_u are standard in all other similar algorithms since [29], as they come from the requirement that k is integer. Most other papers just hide them in the big-oh notation, why in the statement of Theorem 1 we have take great effort to make sure that any dependence on $w_q, w_u, w_1, w_2, t_q, t_u$ is visible and only truly universal constants, like 4, are hidden in the $O(\cdot)$.

The main thing we do lose however is the additive $(t_q/w_q)^{\tilde{O}(\sqrt{k})}$. For $w_q > e^{-\tilde{O}(\sqrt{\log n})}$ this is dominated by the main term, but for very small sets it could potentially be a important. However we note that *if w_q^{-1} is large, we can reduce it to at most $\frac{w_u}{w_2}k$ by hashing*! Define a hash function $h : U \to [m]$ where $m = O(\frac{w_q w_u}{w_2}|U|k)$ and map each set y to $\{i \in [m] \mid \exists e \in y : h(e) = i\}$, that is the OR of the hashed values. With high probability this only distorts the size of the sets and their inner products by a factor $(1 + 1/k)$ which doesn't change ρ.

Analysis Let $\mathcal{T}_q$ and $\mathcal{T}_u$ be the time it takes to compute $R_k(x, t_q)$ and $R_k(y, t_u)$ on given sets. When creating the data structure, decoding each $y \in Y$ takes time $n\mathcal{T}_u$ and uses $n\,\mathrm{E}\left[|R_k(Y, t_u)|\right]$ words of memory for space equivalent. When querying the data structure we first use time $\mathcal{T}_q$ to decode q, then $\mathrm{E}\left[|R_k(X, t_q)|\right]$ time to look in the buckets, and finally $\frac{w_u \log n}{w_2}$ time on each of $\mathrm{E}\left[|R_k(X, t_q) \cap R_k(Y, t_u)|\right] n$ expected collisions with far sets (the worst case is that we never find any y with $y \cap q > w_2|U|$ so we can't return early.)

The key to proving the theorem is thus bounding the above quantities. We do this using the following lemma, which we prove at the end of the section:

Lemma 2.1. *In Algorithm 1 let $k \in \mathbb{Z}_+$ and let $w_q, w_u, w_1, w_2 \in [0, 1]$ be the weights such that $w_1 \geq w_q w_u$. Now let $t_q, t_u \in [0, 1]$ be the thresholds such that $t_q k$ and $t_u k$ are integers, and let $\Delta > 0$ be the branching factor. Given a query set X, with $|X| = w_q\,|U|$, and data set $Y \subseteq U$, with $|Y| = w_u\,|U|$, then running Algorithm 1 with $c_\ell = \sqrt{6.5\ell \log(3k)}$ for $l < k$ and $c_k = 0$, gives that*

$$\mathrm{E}\left[|R_k(X, t_q)|\right] \leq 2\Delta^k \exp(-kD(t_q \mid w_q))\,. \tag{2}$$

$$\mathrm{E}\left[|R_k(Y, t_u)|\right] \leq 2\Delta^k \exp(-kD(t_u \mid w_u))\,. \tag{3}$$

$$\Pr\left[|R_k(X, t_q) \cap R_k(Y, t_u)| \geq 1\right] \geq 7^{-8}k^{-14}\Delta^k \exp(-kD(T_1 \mid P_1)) \qquad \text{if } |X \cap Y| \geq w_1\,|U|\,. \tag{4}$$

$$\mathrm{E}\left[|R_k(X, t_q) \cap R_k(Y, t_u)|\right] \leq 2\Delta^k \exp(-kD(T_2 \mid P_2)) \qquad \text{if } |X \cap Y| \leq w_2\,|U|\,. \tag{5}$$

where $P_j = \begin{pmatrix} w_j & w_q - w_j \\ w_u - w_j & 1 - w_q - w_u + w_j \end{pmatrix}$, $T_j = \begin{pmatrix} t_j & t_q - t_j \\ t_u - t_j & 1 - t_q - t_u + t_j \end{pmatrix}$, $t_j = \arg\inf D(T_j \mid P_j)$ for $j \in \{1, 2\}$.

Finally the expected running times, $\mathcal{T}_q$ and $\mathcal{T}_u$, it takes to compute $R_k(X, t_q)$ and $R_k(Y, t_u)$ respectively are bounded by

$$\begin{aligned} \mathrm{E}\left[\mathcal{T}_q\right] &\leq O\Big(k\,|X| + k(k + \log(n))\Delta^k \exp(-kD(t_q \mid w_q)) \left(\tfrac{t_q(1-w_q)}{w_q(1-t_q)}\right)^{\sqrt{6.5k\log(3k)}}\Big)\,. \\ \mathrm{E}\left[\mathcal{T}_u\right] &\leq O\Big(k\,|Y| + k(k + \log(n))\Delta^k \exp(-kD(t_u \mid w_u)) \left(\tfrac{t_u(1-w_u)}{w_u(1-t_u)}\right)^{\sqrt{6.5k\log(3k)}}\Big)\,. \end{aligned} \tag{6}$$

We define $\Delta = \exp(D(T_1 \mid P_1))$ and k to be the smallest integer at least $\frac{\log n}{D(T_2|P_2) - D_q(t_1|w_q)}$. We make 2 initiations of Algorithm 1, M_1, M_2, with height $k/2$, branching factor Δ, and trimming condition $c_\ell = O(\sqrt{l \log k})$ for $l < k/2$ and $c_{k/2} = 0$.

For each instance we have

$$\begin{aligned} \mathrm{E}\left[\left|R_{k/2}(X, t_q)\right|\right] &\leq 2\exp(k/2\,D(T_1 \mid P_1) - k/2\,D(t_q \mid w_q)) \\ &\leq 2\exp\left(\left(\frac{\log n}{D(T_2 \mid P_2) - D_q(t_1 \mid w_q)} + 2\right)\frac{(D(T_1 \mid P_1) - D(t_q \mid w_q))}{2}\right) \\ &= 2n^{\frac{1}{2}\frac{D(T_1|P_1) - D(t_q|w_q)}{D(T_2|P_2) - D_q(t_1|w_q)}}(D(T_1 \mid P_1) - D(t_q \mid w_q)). \end{aligned}$$

similarly we get

$$\mathrm{E}\left[\left|R_{k/2}(X, t_q)\right|\right] \leq 2n^{\frac{1}{2}\frac{D(T_1|P_1) - D(t_u|w_u)}{D(T_2|P_2) - D_q(t_1|w_q)}}(D(T_1 \mid P_1) - D(t_u \mid w_u)).$$

We combine the two data instances M_1 and M_2 by taking as representative sets returned the product of the sets returned by each of them. In particular, this means we successfully find a near set, if $\left|R_{k/2}(X, t_q) \cap R_{k/2}(Y, t_u)\right| \geq 1$ for both instances, which happens with probability at least

$$(7^{-8} k^{-14} \Delta^k \exp(-kD(T_1 \mid P_1)))^2 = (7^{-8} k^{-14})^2.$$

hence, repeating the algorithm $O(k^{28})$ times we can boost this probability to 99%.

Putting it all together now yields the full version of Theorem 1 contingent on Lemma 2.1. We follow it by a discussion of the differences between it and the idealized version in the introduction.

Theorem 1 (Full version)**.** *Let $w_q, w_u \geq w_1 \geq w_2 \geq 0$ be given with $w_1 \geq w_q w_u$ and $1 \leq t_q, t_u \leq 0$. Set k to be the smallest even integer greater than or equal to $\frac{\log n}{D(T_2|P_2) - D(t_q|w_q)}$ and assume that $t_q k/2$ and $t_u k/2$ are integers. The (w_q, w_u, w_1, w_2)-GapSS problem over a universe U can be solved with expected query time*

$$O(\varsigma_q \, k^{28} \, n^{\rho_q} + k w_q \, |U|) + O\left(\tfrac{t_q(1-w_q)}{w_q(1-t_q)}\right)^{\sqrt{6.5k \log(3k))}} \quad \textit{and space usage} \quad O(\varsigma_u \, k^{28} \, n^{1+\rho_u} + n w_u \, |U|).$$

$$\textit{where} \quad \rho_q = \frac{D(T_1 \mid P_1) - D(t_q \mid w_q)}{D(T_2 \mid P_2) - D(t_q \mid w_q)} \quad \textit{and} \quad \rho_u = \frac{D(T_1 \mid P_1) - D(t_u \mid w_u)}{D(T_2 \mid P_2) - D(t_q \mid w_q)}.$$

and $\varsigma_q = \frac{w_q}{w_2} e^{2(D(T_1|P_1) - D(t_q|w_q))}$, $\varsigma_u = e^{2(D(T_1|P_1) - D(t_u|w_u))}$.

We stress that all previous Locality Sensitive algorithms with time/space trade-offs had $n^{o(1)}$ factors on n^{ρ_q} and n^{ρ_u}. These could be as large as $\exp(\sqrt{\log n})$ or even $\exp((\log n)/(\log\log n))$. In contrast, *our algorithm is the first that only loses $k \approx \log(n)$ multiplicative factors!* We also note that the constants of the size ς_q and ς_u are standard in all other similar algorithms since [29], as they come from the requirement that k is integer. Most other papers just hide them in the big-oh notation, why in the statement of Theorem 1 we have take great effort to make sure that any dependence on $w_q, w_u, w_1, w_2, t_q, t_u$ is visible and only truly universal constants, like 4, are hidden in the $O(\cdot)$.

The main thing we do lose however is the additive $(t_q/w_q)^{\tilde{O}(\sqrt{k})}$. For $w_q > e^{-\tilde{O}(\sqrt{\log n})}$ this is dominated by the main term, but for very small sets it could potentially be a important. However we note that *if w_q^{-1} is large, we can reduce it to at most $\frac{w_u}{w_2} k$ by hashing*! And, as above, the ratio between w_q, w_u and w_1, w_2 is usually considered a small constant. Hashing trick: Define a hash function $h : U \to [m]$ where $m = O(\frac{w_q w_u}{w_2}|U|k)$ and map each set y to $\{i \in [m] \mid \exists e \in y : h(e) = i\}$, that is the OR of the hashed values. With high probability this only distorts the size of the sets and their inner products by a factor $(1 + 1/k)$ which doesn't change ρ.

Contingent on Lemma 2.1 this proves the theorem.

2.1 Bounds on branching

The remainder the section is dedicated to proving Lemma 2.1.

We need the following lemmas:

Lemma 2.2. *Let $k \in \mathbb{Z}_+$ and $p_1, p_2 \in [0,1]$, such that, both $p_1 k$ and $p_2 k$ are integers. Choose $p \in [0,1]$, such that, $p \geq p_1 p_2$. Let $X^{(i)} \in \mathbb{R}^2$ be independent identically distributed 2-dimensional Bernoulli variables, where their probability matrix is $P = \begin{pmatrix} p & p_1 - p \\ p_2 - p & 1 - p_1 - p_2 + p \end{pmatrix}$. We then get that*

$$\Pr\left[\sum_{i \in [k]} X^{(i)} = \left(\begin{smallmatrix} p_1 \\ p_2 \end{smallmatrix}\right) k \wedge \sum_{i \in [k]} X_1^{(i)} X_2^{(i)} = \lceil pk \rceil\right] \geq \frac{1}{400} k^{-3.5}.$$

Lemma 2.3. *Let $k \in \mathbb{Z}_+$ and $p, p_1, p_2 \in [0,1]$, such that, pk, p_1k, and p_2k are integers and $p \geq p_1p_2$. Let $X^{(i)} \in \{0,1\}^2$ be independent identically distributed variables. We then get that*

$$\Pr\left[\forall \ell \leq k : \sum_{i\in[k]} X^{(i)} \geq \binom{p_1}{p_2} l \,\middle|\, \sum_{i\in[k]} X^{(i)} = \binom{p_1}{p_2} k \wedge \sum_{i\in[k]} X_1^{(i)} X_2^{(i)} = pk\right] \geq k^{-3} .$$

The proofs of these are deferred to Appendix D.

We prove Lemma 2.1.

Proof of (2) *and* (3). We only provide the proof for (2) since the proof (3) is completely analogous.

Let $r \in R_k$ be a representative string and define the random variables $\mathcal{X}^{(i)} = [r_i \in X]$ for $i \in [k]$, by definition of Algorithm 1 $(\mathcal{X}^{(i)})_{i\in[k]}$ are independent. We then want to upper bound $\Pr\left[\forall \ell \leq k : \sum_{i\in[\ell]} \mathcal{X}^{(i)} \geq t_q\ell - c_\ell\right] \leq \Pr\left[\sum_{i\in[k]} \mathcal{X}^{(i)} \geq t_qk\right]$. Now by the standard Entropy-Chernoff bound we get that $\Pr\left[\sum_{i\in[k]} \mathcal{X}^{(i)} \geq t_qk\right] \leq \exp(-kD(t_q \mid w_u))$. Using linearity of expectation we get that

$$\mathrm{E}\left[|R_k(X, t_q)|\right] \leq |R_k| \exp(-kD(t_q \mid w_q)) \leq 2\Delta^k \exp(-kD(t_q \mid w_q)) .$$

□

Proof of (5). Like in the proof of (2) and (3) we consider a single representative string $r \in R_k$ and define the random variables $\mathcal{X}^{(i)} = \begin{pmatrix}[r_i \in X]\\ [r_i \in Y]\end{pmatrix}$ for $i \in [k]$, by definition of Algorithm 1 $(\mathcal{X}^{(i)})_{i\in[k]}$ are independent. We want to upper bound $\Pr\left[\forall \ell \leq k : \sum_{i\in[\ell]} \mathcal{X}^{(i)} \geq \binom{t_q}{t_u} \ell - c_\ell\right] \leq$ $\Pr\left[\sum_{i\in[k]} \mathcal{X}^{(i)} \geq \binom{t_q}{t_u} k\right]$. We make a 2-dimensional Entropy-Chernoff bound to get $\Pr\left[\sum_{i\in[k]} \mathcal{X}^{(i)} \geq \binom{t_q}{t_u} k\right] \leq \exp(-D(T_2 \mid P_2))$. We now finish the proof by using linearity of expectation

$$\mathrm{E}\left[|R_k(X, t_q) \cap R_k(Y, t_u)|\right] \leq |R_k| \exp(-D(T_2 \mid P_2)) \leq 2\Delta^k \exp(-D(T_2 \mid P_2)) .$$

□

Proof of (6). As a preprocessing stage we make k sorted lists of $(a_ix)_{x\in X}$ where a_i is the coefficient in $h_i(p \circ x) = h'_i(p) + a_ix \mod q$, this takes $O(k\,|X|)$ time.

We will argue that at each level of tree that we only use $O(k + \log|X|) = O(k + \log n)$ amortized time per active path. More precisely, at level l we use $O((k + \log n)\,|R_l(X, t_q)|)$ amortized time.

Let $l \in [k]$ be fixed and consider an active path $r \in R_l(X, t_q)$. If $\sum_{i\in[\ell]}[r_i \in X] \geq t_q(l+1) - c_{l-1}$ then every one of its children will be active. So we need to find $\{x \in U \mid h_l(p \circ x) < \Delta_l\} = h_l^{-1}([\Delta_l])$. Now $h_l(p \circ x) = h'_l(p) + ax \mod q$ where $a \neq 0 \mod q$ and $s = h'_l(p)$ can be computed in $O(k)$ time. We then get that $h_l^{-1}([\Delta_l]) = \left\{a^{-1}(i - s) \,\middle|\, i \in [\Delta_l]\right\}$, this we can find in time proportional with the number of active children, so charging the cost to them gives the result.

If $\sum_{i\in[\ell]}[r_i \in X] < t_q(l+1) - c_{l-1}$ then only the children $r \circ x \in R_{l+1}$ where $x \in X$ will be active. So we need to find $\{x \in X \mid h_l(p \circ x) < \Delta_l\}$. Again using that $h_l(p \circ x) = h'_l(p) + ax \mod q$ where $a \neq 0 \mod q$ and $s = h'_l(p)$ can be computed in $O(k)$ time, we have reduced the problem to finding $h_l^{-1}([\Delta_l]) = \{x \in X \mid s + ax \mod q < \Delta_l\}$. This we note we can rewrite as

$h_l^{-1}([\Delta_l]) = \{x \in X \mid s \le ax \vee ax < \Delta + s - q\}$, so using our sorted list this can be done in $O(\log n)$ time plus time proportional with the number of active children, so charging this cost to them gives the result.

We bound the expected number of active paths on a level $l \in [k]$. Let $r \in R_l$ be a representative string and define the random variables $\mathcal{X}^{(i)} = [r_i \in X]$ for $i \in [k]$, by definition of Algorithm 1 $(\mathcal{X}^{(i)})_{i\in[k]}$ are independent. We then want to upper bound $\Pr\left[\forall j \le \ell : \sum_{i\in[j]} \mathcal{X}^{(i)} \ge t_q j - c_j\right] \le \Pr\left[\sum_{i\in[\ell]} \mathcal{X}^{(i)} \ge t_q\ell - c_\ell\right]$. By an Entropy-Chernoff bound we get that $\Pr\left[\sum_{i\in[\ell]} \mathcal{X}^{(i)} \ge t_q\ell - c_\ell\right] \le \exp(-lD((t_q - c_\ell/l) \mid w_q))$. By convexity of the Kullback-Leibler divergence we get that $lD((t_q - c_\ell/l) \mid w_q) \ge \ell D(t_q \mid w_q) - c_l \frac{\partial}{\partial t_q} D(t_q \mid w_q)$. It is easy to check that $\frac{\partial}{\partial t_q} D(t_q \mid w_q) = \log \frac{t_q(1-w_q)}{w_q(1-t_q)}$, hence we get that

$$\Pr\left[\sum_{i\in[l]} \mathcal{X}^{(i)} \ge t_q l - c_l\right] \le \exp(-lD(t_q \mid w_q)) \left(\frac{t_q(1-w_q)}{w_q(1-t_q)}\right)^{\sqrt{6.5l\log(3k)}} .$$

So using linearity of expectation we get that

$$\begin{aligned}\mathrm{E}\left[|R_l(X,t_q)|\right] &\le |R_l| \exp(-\ell D(t_q \mid w_q)) \left(\frac{t_q(1-w_q)}{w_q(1-t_q)}\right)^{\sqrt{6.5\ell\log(3k)}} \\ &\le 2\Delta^l \exp(-\ell D(t_q \mid w_q)) \left(\frac{t_q(1-w_q)}{w_q(1-t_q)}\right)^{\sqrt{6.5\ell\log(3k)}} .\end{aligned}$$

Now the expected cost of the tree becomes

$$\begin{aligned}\mathrm{E}\left[\sum_{l\in[k]} O((k+\log(n))\,|R_l(X,t_q)|)\right] &= O((k+\log(n)) \sum_{l\in[k]} \mathrm{E}\left[|R_l(X,t_q)|\right]) \\ &\le O(k(k+\log(n))\Delta^k \exp(-kD(t_q \mid w_q)) \left(\frac{t_q(1-w_q)}{w_q(1-t_q)}\right)^{\sqrt{6.5k\log(3k)}}) .\end{aligned}$$

□

Proof of (4). We will restrict out representative strings further and consider

$$S = \left\{ r \in R_k \;\middle|\; \forall \ell \le k : \begin{pmatrix}[r_i \in X] \\ [r_i \in Y]\end{pmatrix} \ell - c_\ell \le \sum_{i\in[\ell]} \begin{pmatrix}[r_i \in X] \\ [r_i \in Y]\end{pmatrix} \le \begin{pmatrix} t_q \\ t_u \end{pmatrix} \ell \right\} ,$$

It is easy to check that $S \subseteq R_k(X,t_q) \cap R_k(Y,t_u)$, thus we have that

$$\Pr\left[|R_k(X,t_q) \cap R_k(Y,t_u)| \ge 1\right] \ge \Pr\left[|S| \ge 1\right] ,$$

and using Paley-Zygmund's inequality we get that

$$\Pr\left[|S| \ge 1\right] \ge \frac{\mathrm{E}\left[|S|\right]^2}{\mathrm{E}\left[|S|^2\right]} .$$

We then have two goals: 1) Lower bound $\mathrm{E}\left[|S|\right]$, and 2) Upper bound $\mathrm{E}\left[|S|^2\right]$.

Lower bounding $\mathrm{E}[|S|]$

Let $r \in R_k$ be a representative string and define the random variables $\mathcal{X}^{(i)} = \begin{pmatrix} [r_i \in X] \\ [r_i \in Y] \end{pmatrix}$ for $i \in [k]$, and call their law μ. Define the logarithmic moment generating function

$$\Lambda(\lambda) = \log \mathrm{E}\left[\exp(\langle \lambda, \mathcal{X} \rangle)\right] \ ,$$

as well as the Fenchel-Legendre transform of $\Lambda(\lambda)$

$$\Lambda^*(x) = \sup_{\lambda \in \mathbb{R}^2_+} \left(\langle \lambda, x \rangle - \Lambda(\lambda) \right) .$$

We choose z such that $\nabla \Lambda(z) = \left(\begin{smallmatrix} t_q \\ t_u \end{smallmatrix}\right)$, which by a property of the Fenchel-Legendre transform implies that z is well-defined, $z = \nabla \Lambda^*(t_q, t_u)$, and $\Lambda^*(t_q, t_u) = \langle z, t \rangle - \Lambda(z)$.

We will make an exponential change of measure. We define the random variables $\tilde{\mathcal{X}}^{(i)}$ with law $\tilde{\mu}$ given by

$$\mathrm{d}\tilde{\mu}(x) = \exp(\langle z, x \rangle - \Lambda(z))\mathrm{d}\mu(x) \Leftrightarrow \mathrm{d}\mu(x) = \exp(\Lambda(z) - \langle z, x \rangle)\mathrm{d}\tilde{\mu}(x)$$

Since $\int \mathrm{d}\tilde{\mu}(x) = \int \exp(\langle z, x \rangle - \Lambda(z))\mathrm{d}\mu(x) = \exp(-\Lambda(z)) \int \exp(\langle z, x \rangle \mathrm{d}\mu(x) = 1$, then $\tilde{\mu}$ is indeed a probability law. We also have that $\mathrm{E}\left[\tilde{X}\right] = \int x \mathrm{d}\tilde{\mu} = \int x \exp(\langle z, x \rangle - \Lambda(z))\mathrm{d}\mu = \frac{\int x \exp(\langle z, x \rangle)\mathrm{d}\mu}{\mathrm{E}[\exp(\langle z, X \rangle)]} = \nabla \Lambda(z) = \left(\begin{smallmatrix} t_q \\ t_u \end{smallmatrix}\right)$.

We will argue that $\Pr\left[\tilde{\mathcal{X}} = \begin{pmatrix} 1 \\ 1 \end{pmatrix}\right] = t_1$. First we note that

$$\begin{aligned} \Lambda^*(t) &= D(T_1 \mid P_1) \\ &= t_1 \log \frac{t_1}{w_1} + (t_q - t_1) \log \frac{t_q - t_1}{w_q - w_1} + (t_u - t_1) \log \frac{t_u - t_1}{w_u - w_1} + (1 - t_q - t_u + t_1) \log \frac{1 - t_q - t_u + t_1}{1 - w_q - w_u + w_1} \ . \end{aligned}$$

By definition of t_1 we then have that

$$\frac{\partial}{\partial t_1} D(T_1 \mid P_1) = 0 \Leftrightarrow \log \frac{t_1}{w_1} - \log \frac{t_q - t_1}{w_q - w_1} - \log \frac{t_u - t_1}{w_u - w_1} + \log \frac{1 - t_q - t_u + t_1}{1 - w_q - w_u + w_1} = 0 \ .$$

From this we get that

$$\left(\begin{smallmatrix} z_q \\ z_u \end{smallmatrix}\right) = z = \nabla_{\left(\begin{smallmatrix} t_q \\ t_u \end{smallmatrix}\right)} \Lambda^*(t) = \begin{pmatrix} \log \frac{t_q - t_1}{w_q - w_1} - \log \frac{1 - t_q - t_u + t_1}{1 - w_q - w_u + w_1} \\ \log \frac{t_u - t_1}{w_u - w_1} - \log \frac{1 - t_q - t_u + t_1}{1 - w_q - w_u + w_1} \end{pmatrix} = \begin{pmatrix} \log \frac{t_1}{w_1} - \log \frac{t_u - t_1}{w_u - w_1} \\ \log \frac{t_u - t_1}{w_u - w_1} - \log \frac{1 - t_q - t_u + t_1}{1 - w_q - w_u + w_1} \end{pmatrix}$$

and $\Lambda(z) = -\log \frac{1 - t_q - t_u + t_1}{1 - w_q - w_u + w_1}$. So we have that $z_q + z_u - \Lambda(z) = \log \frac{t_1}{z_1}$, which gives us that $\Pr\left[\tilde{\mathcal{X}} = \begin{pmatrix} 1 \\ 1 \end{pmatrix}\right] = \exp(z_1 + z_2 - \Lambda(z)) \Pr\left[\mathcal{X} = \begin{pmatrix} 1 \\ 1 \end{pmatrix}\right] = t_1$. This also give us that $\Pr\left[\tilde{\mathcal{X}} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}\right] = 1 - t_q - t_u + t_1$.

We define the set

$$\begin{aligned} A = &\left\{ (x^{(i)})_{i \in [k]} \in \left(\mathbb{R}^2\right)^k \,\middle|\, \forall \ell \le k : \left(\begin{smallmatrix} t_q \\ t_u \end{smallmatrix}\right) \ell - c_\ell \le \sum_{i \in [\ell]} x^{(i)} \le \left(\begin{smallmatrix} t_q \\ t_u \end{smallmatrix}\right) \ell \right\} \\ &\cap \left\{ (x^{(i)})_{i \in [k]} \in \left(\mathbb{R}^2\right)^k \,\middle|\, \sum_{i \in [\ell]} x^{(i)} = \left(\begin{smallmatrix} t_q \\ t_u \end{smallmatrix}\right) k \wedge \sum_{i \in [\ell]} (1 - x_1^{(i)})(1 - x_2^{(i)}) = \lceil (1 - t_q - t_u + t_1) k \rceil \right\} \ , \end{aligned}$$

and note that

$$\Pr\left[\forall \ell \le k : \left(\begin{smallmatrix} t_q \\ t_u \end{smallmatrix}\right) \ell - c_\ell \le \sum_{i\in[\ell]} \mathcal{X}^{(i)} \le \left(\begin{smallmatrix} t_q \\ t_u \end{smallmatrix}\right) \ell\right] \ge \Pr\left[(\mathcal{X}^{(i)})_{i\in[k]} \in A\right] .$$

Now using the exponential change of measure, we get that

$$\begin{aligned}
\Pr\left[(\mathcal{X}^{(i)})_{i\in[k]} \in A\right] &= \int_{(x^{(i)})_{i\in[k]}\in A} \mathrm{d}\mu^{\otimes k} \\
&= \int_{(\tilde{x}^{(i)})_{i\in[k]}\in A} \exp(k\Lambda(z) - \langle z, \sum_{i\in[k]} \tilde{x}^{(i)}\rangle)\, \mathrm{d}\tilde{\mu}^{\otimes k} \\
&= \exp(-kD(T_1 \mid P_1)) \int_{(\tilde{x}^{(i)})_{i\in[k]}\in A} \exp\left(-\langle z, \sum_{i\in[k]} \left(\tilde{x}^{(i)} - \left(\begin{smallmatrix} t_q \\ t_u \end{smallmatrix}\right)\right)\rangle\right) \mathrm{d}\tilde{\mu}^{\otimes k} \\
&= \exp(-kD(T_1 \mid P_1)) \Pr\left[(\tilde{\mathcal{X}}^{(i)})_{i\in[k]} \in A\right]
\end{aligned}$$

where the last inequality follows from the fact that if $(\tilde{x}^{(i)})_{i\in[k]} \in A$ then $\sum_{i\in[k]} \tilde{x}^{(i)} = \left(\begin{smallmatrix} t_q \\ t_u \end{smallmatrix}\right) k$.

We can bound $\Pr\left[(\tilde{\mathcal{X}}^{(i)})_{i\in[k]} \in A\right]$ as

$$\begin{aligned}
&\Pr\left[(\tilde{\mathcal{X}}^{(i)})_{i\in[k]} \in A\right] \\
&\ge \Pr\left[\forall \ell \le k : \sum_{i\in[k]} \tilde{\mathcal{X}}^{(i)} \le \left(\begin{smallmatrix} t_q \\ t_u \end{smallmatrix}\right) \ell \wedge \sum_{i\in[k]} \tilde{\mathcal{X}}^{(i)} = \left(\begin{smallmatrix} t_q \\ t_u \end{smallmatrix}\right) k \wedge \sum_{i\in[k]} (1 - \tilde{\mathcal{X}}_1^{(i)})(1 - \tilde{\mathcal{X}}_2^{(i)}) = \lceil (1 - t_q - t_u + t_1)k \rceil\right] \\
&\qquad - \Pr\left[\exists \ell \le k : \sum_{i=1}^{l} \tilde{\mathcal{X}}_1^{(i)} \le t_q \ell - c_\ell\right] - \Pr\left[\exists \ell \le k : \sum_{i=1}^{l} \mathcal{X}_2^{(i)} \le t_u \ell - c_\ell\right]
\end{aligned}$$

We will first lower bound the first term in the expression. Here we want to use Lemma 2.2 and Lemma 2.3 and to ease the notation we introduce the negated random variables $\mathcal{Y}^{(i)} = 1 - \tilde{\mathcal{X}}^{(i)}$. We then have that $\mathrm{E}\left[\mathcal{Y}^{(i)}\right] = \left(\begin{smallmatrix} 1-t_q \\ 1-t_u \end{smallmatrix}\right)$ and $\Pr\left[\mathcal{Y}^{(i)} = \left(\begin{smallmatrix} 1 \\ 1 \end{smallmatrix}\right)\right] = 1 - t_q - t_u + t_1 \ge (1 - t_q)(1 - t_u)$. We can then rewrite using $\mathcal{Y}^{(i)}$:

$$\begin{aligned}
&\Pr\left[\forall \ell \le k : \sum_{i\in[k]} \tilde{\mathcal{X}}^{(i)} \le \left(\begin{smallmatrix} t_q \\ t_u \end{smallmatrix}\right) \ell \wedge \sum_{i\in[k]} \tilde{\mathcal{X}}^{(i)} = \left(\begin{smallmatrix} t_q \\ t_u \end{smallmatrix}\right) k \wedge \sum_{i\in[k]} (1 - \tilde{\mathcal{X}}_1^{(i)})(1 - \tilde{\mathcal{X}}_2^{(i)}) = \lceil (1 - t_q - t_u + t_1)k \rceil\right] \\
&= \Pr\left[\forall \ell \le k : \sum_{i\in[k]} \mathcal{Y}^{(i)} \ge \left(\begin{smallmatrix} 1-t_q \\ 1-t_u \end{smallmatrix}\right) \ell \wedge \sum_{i\in[k]} \mathcal{Y}^{(i)} = \left(\begin{smallmatrix} 1-t_q \\ 1-t_u \end{smallmatrix}\right) k \wedge \sum_{i\in[k]} \mathcal{Y}_1^{(i)} \mathcal{Y}_2^{(i)} = \lceil (1 - t_q - t_u + t_1)k \rceil\right]
\end{aligned}$$

Now using Lemma 2.2 we have that

$$\Pr\left[\sum_{i\in[k]} \mathcal{Y}^{(i)} = \left(\begin{smallmatrix} 1-t_q \\ 1-t_u \end{smallmatrix}\right) k \wedge \sum_{i\in[k]} \mathcal{Y}_1^{(i)} \mathcal{Y}_2^{(i)} = \lceil (1 - t_q - t_u + t_1)k \rceil\right] \ge \frac{1}{400} k^{-3.5} .$$

Combining this with Lemma 2.3 we get that

$$\Pr\left[\forall \ell \le k : \sum_{i\in[k]} \mathcal{Y}^{(i)} \ge \binom{1-t_q}{1-t_u}\ell \wedge \sum_{i\in[k]} \mathcal{Y}^{(i)} = \binom{1-t_q}{1-t_u}k \wedge \sum_{i\in[k]} \mathcal{Y}_1^{(i)}\mathcal{Y}_2^{(i)} = \lceil (1-t_q-t_u+t_1)k\rceil\right]$$
$$= \Pr\left[\sum_{i\in[k]} \mathcal{Y}^{(i)} = \binom{1-t_q}{1-t_u}k \wedge \sum_{i\in[k]} \mathcal{Y}_1^{(i)}\mathcal{Y}_2^{(i)} = \lceil (1-t_q-t_u+t_1)k\rceil\right]$$
$$\cdot \Pr\left[\forall \ell \le k : \sum_{i\in[k]} \mathcal{Y}^{(i)} \ge \binom{1-t_q}{1-t_u}\ell \,\middle|\, \sum_{i\in[k]} \mathcal{Y}^{(i)} = \binom{1-t_q}{1-t_u}k \wedge \sum_{i\in[k]} \mathcal{Y}_1^{(i)}\mathcal{Y}_2^{(i)} = \lceil (1-t_q-t_u+t_1)k\rceil\right]$$
$$\ge \frac{1}{400}k^{-6.5} .$$

We now upper bound $\Pr\left[\exists \ell \le k : \sum_{i=1}^{l} \tilde{\mathcal{X}}_1^{(i)} \le t_q\ell - c_\ell\right]$. This will be done by a union bound and Hoeffding's inequality.

$$\Pr\left[\exists \ell \le k : \sum_{i=1}^{l} \tilde{\mathcal{X}}_1^{(i)} \le t_q\ell - c_\ell\right] = \sum_{l\le k}\Pr\left[\sum_{i=1}^{l} \tilde{\mathcal{X}}_1^{(i)} \le t_q\ell - c_\ell\right] \le \sum_{l\le k}\exp\left(-\frac{c_\ell^2}{l}\right) \le \frac{1}{1200}k^{-6.5}$$

where we have used that $c_\ell = O(\sqrt{l\log(k)})$. Similarly, we upper bound

$$\Pr\left[\exists \ell \le k : \sum_{i=1}^{l} \tilde{\mathcal{X}}_2^{(i)} \le t_u\ell - c_\ell\right] \le \frac{1}{1200}k^{-6.5}$$

We now get that

$$\Pr\left[(\mathcal{X}^{(i)})_{i\in[k]} \in A\right] \ge \frac{1}{1200}\exp(-kD(T_1 \mid P_1))k^{-6.5} ,$$

so by linearity of expectation we get that

$$\mathrm{E}[S] \ge |R_k|\frac{1}{1200}k^{-6.5}\exp(-kD(T_1 \mid P_1)) \ge \frac{1}{1200}k^{-6.5}\Delta^k\exp(-kD(T_1 \mid P_1)) .$$

Upper bounding $\mathrm{E}\left[|S|^2\right]$

Consider two representative strings $r, r' \in R_k$ and let $q \in R_l$ be their common prefix, hence l is the length of their common prefix. Define the random variables $\mathcal{X}^{(i)} = \begin{pmatrix}[r_i \in X]\\ [r_i \in Y]\end{pmatrix}$, $\mathcal{Y}^{(j)} = \begin{pmatrix}[r'_j \in X]\\ [r'_j \in Y]\end{pmatrix}$, and $\mathcal{Z}^{(h)} = \begin{pmatrix}[q_h \in X]\\ [q_h \in Y]\end{pmatrix}$ for $i, j \in [k]\setminus[\ell]$ and $h \in [l]$. We then get that

$$\Pr\left[p, p' \in S\right] \le \Pr\left[\sum_{h\in[\ell]}\mathcal{Z}^{(h)} + \sum_{i\in[k]\setminus[l]}\mathcal{X}^{(i)} \ge tk \wedge \sum_{h\in[\ell]}\mathcal{Z}^{(h)} + \sum_{j\in[k]\setminus[l]}\mathcal{Y}^{(j)} \ge tk \wedge \sum_{h\in[\ell]}\mathcal{Z}^{(h)} \le tl\right]$$
$$\le \Pr\left[\sum_{h\in[\ell]}\mathcal{Z}^{(h)} + \sum_{i\in[k]\setminus[l]}\mathcal{X}^{(i)} + \sum_{j\in[k]\setminus[l]}\mathcal{Y}^{(j)} \ge (2k-\ell)t - \delta\right] .$$

Now $\sum_{h\in[\ell]} \mathcal{Z}^{(h)} + \sum_{i\in[k]\setminus[l]} \mathcal{X}^{(i)} + \sum_{j\in[k]\setminus[l]} \mathcal{Y}^{(j)}$ is almost a sum of independent random variable. We have that $\mathcal{X}^{(k-\ell+1)}$ and $\mathcal{Y}^{(k-\ell+1)}$ are correlated since they are chosen by sampling without replacement, but this implies that

$$\mathrm{E}\left[\exp(\langle \lambda, \mathcal{X}^{(k-\ell+1)} + \mathcal{Y}^{(k-\ell+1)}\rangle)\right] \le \mathrm{E}\left[\exp(\langle \lambda, \mathcal{X}^{(k-\ell+1)}\rangle)\right] \mathrm{E}\left[\exp(\langle \lambda, \mathcal{Y}^{(k-\ell+1)}\rangle)\right]$$

We can then use a 2-dimensional Entropy-Chernoff bound and get that

$$\Pr\left[\sum_{h\in[\ell]} \mathcal{Z}^{(h)} + \sum_{i\in[k]\setminus[l]} \mathcal{X}^{(i)} + \sum_{j\in[k]\setminus[l]} \mathcal{Y}^{(j)} \ge (2k-\ell)t - \delta\right] \le \exp(-(2k-\ell)D(T_1 \mid P_1)) \ ,$$

Using this we can upper bound $\mathrm{E}\left[|S|^2\right] = \mathrm{E}\left[\sum_{r,r'\in R_k}[r, r' \in S]\right]$ by splitting the sum by the length of their common prefix.

$$\begin{aligned}
\mathrm{E}\left[|S|^2\right] &= \mathrm{E}\left[\sum_{r,r'\in S_k}[r, r' \in S]\right] \\
&\le \sum_{i=1}^{k}\left(\prod_{j=1}^{i}\Delta_j\right)\binom{\Delta_{i+1}}{2}\left(\prod_{j=i+2}^{k}\Delta_j\right)\exp(-(2k-i)D(T_1 \mid P_1)) \\
&\le \left(\prod_{j=1}^{k}\Delta_j\right)^2 \exp(-2kD(T_1 \mid P_1))\sum_{i=1}^{k}\exp(iD(T_1 \mid P_1))\left(\prod_{j=1}\Delta_j\right)^{-1} \\
&\le \left(\prod_{j=1}^{k}\Delta_j\right)^2 \cdot \exp(-2kD(T_1 \mid P_1)) \cdot k \cdot \exp(kD(T_1 \mid P_1)) \cdot \Delta^{-k} \\
&\le 4k\Delta^k \exp(-kD(T_1 \mid P_1))
\end{aligned}$$

Finishing the proof

Having lower bounded $\mathrm{E}\left[|S|\right]$ and upper bounded $\mathrm{E}\left[|S|^2\right]$ we can finish the proof.

$$\begin{aligned}
\Pr\left[|R_k(X, t_q) \cap R_k(Y, t_u)| \ge 1\right] &\ge \Pr\left[|S| \ge 1\right] \\
&\ge \frac{\mathrm{E}\left[|S|\right]^2}{\mathrm{E}\left[|S|^2\right]} \\
&\ge \frac{\frac{1}{1200^2}k^{-13}\Delta^{2k}\exp(-2kD(T_1 \mid P_1))}{4k\Delta^k\exp(-kD(T_1 \mid P_1))} \\
&= \frac{1}{2400^2}k^{-14}\Delta^l\exp(-kD(T_1 \mid P_1)) \ .
\end{aligned}$$

□

3 Lower bounds

As we discussed in the introduction, it is necessary for our lower bounds to assume $d = \omega(\log n)$. We will also assume w_q, w_u, w_1, w_2 are constants, like we do four our upper bounds, though we don't believe this to be necessary.

We proceed to define the hard distributions for all further lower bounds.

1. A query $x \in \{0,1\}^d$ is created by sampling d random independent bits with Bernoulli(w_q) distribution.

2. A dataset $P \subseteq \{0,1\}^d$ is constructed by sampling $n-1$ vectors with random independent bits from such that $y_i \sim$ Bernoulli(w_2/w_q) if $x_i = 1$ and $y_i \sim$ Bernoulli$((w_u - w_2)/(1-w_q))$ otherwise, for all $y \in P$.

3. A 'close point', y', is created by $y'_i \sim$ Bernoulli(w_1/w_q) if $x_i = 1$ and $y'_i \sim$ Bernoulli$((w_u - w_1)/(1-w_q))$ otherwise. This point is also added to P.

The values are chosen such that $E|x \cap y|/d = w_1$, $E|y|/d = w_u$ for all $y \in P$, and $E|x \cap y'|/d = w_1$ and $E|x \cap y|/d = w_u$ for all $y \in P \setminus \{y'\}$. By a union bound over P, the actual values are within factors $1+o(1)$ of their expectations with high probability. Changing at most $o(\log n)$ coordinates we ensure the weights of queries/database points is exactly their expected value, while only changing the inner products by factors $1+o(1)$. Since the changes don't contain any new information, we can assume for lower bounds that entries are independent. Thus any $(w_q, w_u, w_1(1-o(1)), w_2(1+o(1)))$-GapSS data structure on P must thus be able to return y' with at least constant probability when given the query x.

Model For the first bound follow O'Donnel et al. [40] and Christiani [22] and directly lower bound the quantity $\frac{\log(p_1/\min\{p_u,p_q\})}{\log(p_2/\min\{p_u,p_q\})}$ which lower bounds ρ_u and ρ_q in Definition 3.

For the second and third lower bound we follow Andoni et al. [8] and lower bound a general so called "list-of-points" data structures (see Definition 2). This is a slightly more general model, though no it is believed that all bounds for the first model can be shown in the list-of-points model as well.

The second and third bound are shown using so called Hypercontractive inequalities and can be extended to show cell probe lower bounds by the arguments in [43].

3.1 p-biased analysis

In the analysis of Boolean functions is is common to use $\{-1,1\}^d$ as the function domain. We'll map 1 to -1 (true) and 0 to 1 (false).

Given functions $f, g : \{-1,1\}^n \to \{0,1\}$, we write

$$f(x) = \sum_{S \subseteq [n]} \hat{f}(S)\phi_S(x), \qquad g(y) = \sum_{S \subseteq [n]} \hat{g}(S)\gamma_S(y) \tag{7}$$

where $\hat{f}, \hat{g} : \mathcal{P}([n]) \to \mathbb{R}$ and $\phi(x_i) = \frac{x_i - \mu_q}{\sigma_q}$, $\gamma(y_i) = \frac{y_i - \mu_u}{\sigma_u}$ for $\mu_q = 1 - 2w_q, \sigma_q = 2\sqrt{w_q(1-w_q)}$ and $\mu_u = 1 - 2w_u, \sigma_u = 2\sqrt{w_u(1-w_u)}$. Finally $\phi_S, \gamma_S : \{-1,1\}^n \to \mathbb{R}$ are defined $\phi_S(x) = \prod_{i \in S} \phi(x_i)$ and respectively for y.

Any Boolean function can be expanded as (7), but it is particularly useful in our case. To see why, let π be the probability distribution, with the following probability mass function: $\pi(-1) = w_q, \pi(1) = 1 - w_q$, and let $\pi^n : \{-1,1\}^n \to [0,1]$ be the product distribution on $\{-1,1\}^n$. We then have the useful properties:

$$p_q = \Pr_{x \sim \pi^n}[f(x) = 1] = E_{x \sim \pi^n}[f(x)] = E_{x \sim \pi^n}[\sum_{S \subseteq [n]} \hat{f}(S)\phi_S(x)] = \hat{f}(\emptyset)$$

$$= E_{x \sim \pi^n_{w_q}}[f(x)^2] = E_{x \sim \pi^n_{w_q}}[\sum_{S,T \subseteq [n]} \hat{f}(S)\hat{f}(T)\phi_S(x)\phi_T(x)] = \sum_{S \subseteq [n]} \hat{f}(S)^2.$$

If we think of f as an LSF-filter, $\Pr_{x\sim\pi^n}[f(x)=1]$ is the probability that the filter accepts a random point with expected weight w_q (nw_q of coordinates being -1).

Next, we let ψ be the probability distribution, with the following probability mass function:

$$\psi(-1,-1)=w \qquad \psi(-1,1)=w_q-w$$
$$\psi(1,-1)=w_u-w \qquad \psi(1,1)=1-w_q-w_u+w,$$

then $p=\Pr_{x,y\sim\psi^n}[f(x)=1,g(y)=1]$ is the probability that a random query point, x, and a random data point, y, with $E\langle x,y\rangle/d=w$ both get caught by their respective filter. ($\Pr[x\in Q, y\in U]$ in the language of Definition 3.) This has the following nice form:

$$\begin{aligned}
p &= E_{x,y\sim\psi^n}[f(x)g(y)]\\
&= E_{x,y\sim\psi^n}[\sum_{S,T\subseteq[n]}\hat f(S)\hat g(T)\phi_S(x)\gamma_T(x)]\\
&= \sum_{S\subseteq[n]}\hat f(S)\hat g(S)E_{x,y\sim\psi^n}[\phi_S(x)\gamma_T(x)]\\
&= \sum_{S\subseteq[n]}\hat f(S)\hat g(S)E_{x,y\sim\psi^n}\left[\prod_{i\in S}\frac{x_i-\mu_q}{\sigma_q}\frac{y_i-\mu_u}{\sigma_u}\right]\\
&= \sum_{S\subseteq[n]}\hat f(S)\hat g(S)\left(\frac{E_{x,y\sim\psi^n}[x_iy_i]-\mu_q\mu_u}{\sigma_q\sigma_u}\right)^{|S|}\\
&= \sum_{S\subseteq[n]}\hat f(S)\hat g(S)\left(\frac{w-w_qw_u}{\sqrt{w_q(1-w_q)w_u(1-w_u)}}\right)^{|S|}.
\end{aligned}$$

Note that in the case $w=w_qw_u$ all terms except ($S=\emptyset$) are 0, so we have $\Pr_{x,y\sim\psi^n}[f(x)=1,g(y)=1]=\Pr[f(x)=1]\Pr[g(y)=1]$ as we would expect for x and y independent.

We will define the norm $\|f\|_q=(E_{x\sim\pi^n}f(x)^q)^{1/q}$ and equivalently for g with its respective distribution. Note that since f and g are Boolean, we have $\|f\|_q=(E_{x\sim\pi^n}f(x))^{1/q}=\Pr_q[f(x)=1]^{1/q}$. This will turn out to be very useful.

3.2 Symmetric Lower bound

The simplest approach is to use the expansion directly. This is what O'Donnel used [40] to prove the first optimal LSH lower bounds of $\rho\geq 1/c$ for data-independent hashing. Besides handling the case of set similarity with filters rather than hash functions, we slightly generalize the approach a big by using the power-means inequality rather than log-concavity. [6]

We will show an inequality on the form

$$\left(\frac{\Pr_{x,y',f,g}[f(x)=1,g(y')=1]}{\min\{\Pr_{x,f}[f(x)=1],\Pr_{y,g}[g(y)=1]\}}\right)^{1/\log\alpha}\leq\left(\log\frac{\Pr_{x,y,f,g}[f(x)=1,g(y)=1]}{\min\{\Pr_{x,f}[f(x)=1],\Pr_{y,g}[g(y)=1]\}}\right)^{1/\log\beta}$$

where $\alpha=\frac{w_1-w_qw_u}{\sqrt{w_q(1-w_q)w_u(1-w_u)}}$ and $\beta=\frac{w_2-w_qw_u}{\sqrt{w_q(1-w_q)w_u(1-w_u)}}$, and y' and y are sampled as respectively a close and a far point (see the top of the section).

[6]This widens the range in which the bound is applicable – the O'Donnel bound is only asymptotic for $r\to 0$. However the values we obtain outside this range, when applied to Hamming space LSH, aren't sharp against the upper bounds.

If we knew the filter family $\mathcal{F}$ was regular, that is $(Q, U) \sim \mathcal{F}$ have fixed $|Q|$ and $|U|$, we wouldn't have to take the expectation over f and g. However doing so is only a minor syntactic annoyance in the proof below, and ensures that one cannot circumvent the proof by letting $|Q|$ and $|U|$ be stochastic.

We will prove something slightly stronger than Theorem 2:

Lemma 3.1. *Given an LSF-family $\mathcal{F}$, let $f, g : \{-1,1\}^n \to \{0,1\}$ be random functions such that $f^{-1}(1) = Q$ and $g^{-1}(1) = U$ for $Q, U \sim \mathcal{F}$. Assume further that $E_{f,g}[\sum_{|S|=k} \hat{f}(S)\hat{g}(S)] \geq 0$ for all $k \in [n]$, then any LSF data structure with $\rho_q = \rho_u = \rho$ must have*

$$\rho \geq \log\left(\frac{w_1 - w_q w_u}{\sqrt{w_q(1-w_q)w_u(1-w_u)}}\right) \Bigg/ \log\left(\frac{w_2 - w_q w_u}{\sqrt{w_q(1-w_q)w_u(1-w_u)}}\right).$$

In particular this bound holds when $\hat{f} = \hat{g}$ almost surely, since $\sum_{|S|=k} \hat{f}(S)^2$ is clearly non-negative. In the context of Theorem 2 we have $w_q = w_u$ and $Q = U$ as $Q, U \sim \mathcal{F}$, so Theorem 2 follows from Lemma 3.1.

Proof. Let $\alpha = \frac{w_1 - w_q w_u}{\sqrt{w_q(1-w_q)w_u(1-w_u)}}$ and $\beta = \frac{w_2 - w_q w_u}{\sqrt{w_q(1-w_q)w_u(1-w_u)}}$, such that $p_1 = \sum_{S \subseteq [n]} \hat{f}(S)\hat{g}(S)\alpha^{|S|}$ and $p_2 = \sum_{S \subseteq [n]} \hat{f}(S)\hat{g}(S)\beta^{|S|}$.

By Hölder's inequality $\sum_S \hat{f}(S)\hat{g}(S) \leq \min\{\|f\|_1\|g\|_\infty, \|f\|_\infty\|g\|_1\} = \min\{p_q, p_u\}$. Let $p = E_{f,g}[\min\{p_u, p_q\}]$. By assumption $E_{f,g}[\sum_{|S|=k} \hat{f}(S)\hat{g}(S)] \geq 0$ for all k, so we have that $\sum_k \alpha^k E_{f,g}[\sum_{|S|=k} \hat{f}(S)\hat{g}(S)]/p$ is a weighted average over the α^k terms. As such we can use the power-means inequality:

$$\begin{aligned}(E_{f,g}[p_1]/p)^{1/\log\alpha} &= \Big(\sum_{k\in[n]} (e^k)^{\log\alpha} E_{f,g}[\sum_{|S|=k} \hat{f}(S)\hat{g}(S)]/p\Big)^{1/\log\alpha} \\ &\leq \Big(\sum_{k\in[n]} (e^k)^{\log\beta} E_{f,g}[\sum_{|S|=k} \hat{f}(S)\hat{g}(S)]/p\Big)^{1/\log\beta} = (E_{f,g}[p_2]/p)^{1/\log\beta}.\end{aligned}$$

which implies by rearrangement

$$\rho = \frac{\log(E_{f,g}[p_1]/p)}{\log(E_{f,g}[p_2]/p)} \geq \frac{\log\alpha}{\log\beta}.$$

For ρ_q the inequality above follows from $\log(p/p_1)/\log(p/p_2)$ being increasing in p. For ρ_u it is simply increasing the denominator or decreasing the numerator. $\square$

As noted the bound is sharp against our upper bound when w_u, w_q, w_1, w_2 are all small. Also notice that $\log\alpha/\log\beta \leq \frac{1-\alpha}{1+\alpha}\frac{1-\beta}{1+\beta}$ is a rather good approximation for α and β close to 1. Here the right hand side is the ρ value of Spherical LSH with the batch-normalization embedding discussed in Appendix B.1.

It would be interesting to try an extend this bound to get rid of the $\sum_{|S|=k} \hat{f}(S)\hat{g}(S) \geq 0$ assumption. We conjecture that this is the case, such that the bound holds even when $f \neq g$ and $w_q \neq w_u$.

Note that the lower bound becomes 0 when $w_2 \to w_q w_u$. In the next sections we will find a bound for exactly this case.

3.3 Hypercontractive Lower Bounds

For $x \in \{-1,1\}^d$ let $y \sim N_\sigma(x)$ be sampled by choosing $y_i \in \{-1,1\}$ for each coordinate $i \in [d]$ independently, such that with probability σ we set $y_i = x_i$ and otherwise we set y_i at random. The classic Bonami-Beckner (or Hypercontractive) inequality [40] says that for any $1 \le p \le q$ and $0 \le \sigma \le \sqrt{(p-1)/(q-1)}$, any Boolean function $f : \{-1,1\}^d \to \mathbb{R}$ satisfies $\|T_\sigma f\|_q \le \|f\|_p$ where $T_\sigma f(x) = E_{y \sim N_\sigma(x)}[f(y)]$.

This inequality has been used to show many lower bounds for locality sensitive data structures, see e.g. [36, 43, 22, 8]. Usually in a model like the following defined in [8] as a "list-of-points" data structure.

Definition 2 (List-of-points)**.** *Given some universes, Q, U, a similarity measure $S : Q \times U \to [0,1]$ and two thresholds $1 \ge s_1 > s_2 \ge 0$,*

1. *We fix (possibly random) sets $A_i \subseteq \{-1,1\}^d$, for $1 \le i \le m$; and with each possible query point $q \in \{-1,1\}^d$, we associate a (random) set of indices $I(q) \subseteq [m]$;*
2. *For a given dataset P, we maintain m lists of points $L_1, L_2, \dots, L_m$, where $L_i = P \cap A_i$.*
3. *On query q, we scan through each list L_i for $i \in I(q)$ and check whether there exists some $p \in L_i$ with $S(q,p) \ge s_2$. If it exists, return p.*

The data structure succeeds, for a given $q \in Q, p \in P$ with $S(q,p) \ge s_1$, if there exists $i \in I(q)$ such that $p \in L_i$.

We use the same hard distribution as before. By Yao's principle we can assume the data structure is deterministic. For $i \in [m]$ we define $p_q^{(i)} = \Pr[i \in I(q)]$, $p_u^{(i)} = \Pr[y \in A_i]$, and $p_1^{(i)} \ge \Pr[i \in I(q), y \in A_i]$ for all q, y with $S(q,y) \ge s_1$.

Let r and s be constants, if

$$p_1^{(i)} \le (p_q^{(i)})^{1/r} (p_u^{(i)})^{1/s} \quad \text{for all } i, \tag{8}$$

it was shown in [8] (lemma 7.2 and 7.3) that if "far points" are independent of the query, that is $\Pr[i \in I(q), y \in A_i] = p_q^{(i)} p_u^{(i)}$ for all q, i and $y \in P$, then any data structure that succeeds with constant probability must have

$$\rho_q \ge (1+\rho_u)(1-r) + r/s \tag{9}$$

where n^{ρ_q} is the expected query time and $n^{1+\rho_u}$ is the expected space consumption. In our case the "random" requirement corresponds to $w_2 = w_q w_u$.

Note that eq. (9) is equivalent to the requirement to the parametric inequalities

$$\rho_q \ge (1/r - 1)\alpha + 1/s \quad \text{or} \quad \rho_u \ge \alpha/r + 1/s - 1,$$

where $\alpha \ge 0$ which is the form we will use below. [7]

It was also shown in [8] that eq. (8) gives a cell-probe lower bound of $\Omega((n/w)^{\frac{r}{(r-1)s}})$ memory cells of size w, when the data structure is only allowed 1 probe. We won't go into details about this, just note that this matches eq. (9) at $\rho_q = 0$ for memory cells up to size $n^{o(1)}$.

[7]This is indeed what we would have expected from the LSF-model, Definition 3, since $\rho_q = \frac{\log p_1/p_q}{\log p_2/p_q} = \frac{\log p_q^{1/r} p_u^{1/s}/p_q}{\log p_q p_u/p_q} = (1/r-1)\alpha + 1/s$ and $\rho_u = \frac{\log p_1/p_u}{\log p_2/p_q} = \frac{\log p_q^{1/r} p_u^{1/s}/p_u}{\log p_q p_u/p_q} = \alpha/r + 1/s - 1$ for $\alpha = \frac{\log p_q}{\log p_u} \ge 0$.

3.3.1 Lower Bound 1

Recall Theorem 3: Given $\alpha \geq 0$ and $r, s \geq 2$, let $u_q = \log \frac{1-w_q}{w_q}$, $u_u = \log \frac{1-w_u}{w_u}$, $\sigma = \frac{\sinh(u_q(1-1/r))}{\sinh(u_q/r)}$, and $\upsilon = \frac{\sinh(u_u(1-1/s))}{\sinh(u_u/s)}$. If r and s are such that $\sqrt{\sigma\upsilon} = \frac{w_1 - w_q w_u}{\sqrt{w_q(1-w_q)w_u(1-w_u)}}$, then any LSF data structure must have

$$\rho_q \geq (1/r-1)\alpha + 1/s \quad \text{or} \quad \rho_u \geq \alpha/r + 1/s - 1,$$

Proof. We will prove this theorem using the p-biased version of the hypercontractive inequality, which says:

Theorem 5 ([39] also [38] Theorem 10.18 and [49])**.** *Let (Ω, π) be a finite probability space, $|\Omega| \geq 2$, in which every outcome has probability at least $\lambda < 1/2$. Let $f \in L^2(\Omega, \pi)$. Then for any $q > 2$ and $\upsilon = \sqrt{\frac{\sinh(u/q)}{\sinh(u/q')}}$,*

$$\sum_{S \subseteq [n]} \upsilon^2 \hat{f}(S)^2 \leq \|f\|_{q'}$$

where $1/q + 1/q' = 1$ and $u = \log \frac{1-\lambda}{\lambda}$.

We can generalize this to two general functions, using Cauchy Schwartz:

$$p_1 = \sum_{S \subseteq [n]} \sqrt{\upsilon\sigma}^{|S|} \hat{f}(S)\hat{g}(S) \leq \sqrt{\sum_{S \subseteq [n]} \upsilon^{|S|} \hat{f}^2(S) \sum_{S \subseteq [n]} \sigma^{|S|} \hat{g}^2(S)} \leq \|f\|_r \|g\|_s = p_q^{1/r} p_u^{1/s}.$$

where $\upsilon = \frac{\sinh(u_q(1-1/r))}{\sinh(u_q/r)}$, $\sigma = \frac{\sinh(u_u(1-1/s))}{\sinh(u_u/s)}$, $u_q = \log \frac{1-w_q}{w_q}$, $u_u = \log \frac{1-w_u}{w_u}$ and $r, s \geq 2$.

Combined with the discussion around eq. (9) this gives the theorem. ☐

We continue to prove a lower bound for the case $w_q = w_u$, $w_2 = w_q w_u$. This is discussed in the introduction, but here we check all the necessary details. In the theorem, we set $\alpha = 1$, $r = s$ and $\sigma = \upsilon$, then the theorem asks us to set

$$r = \frac{2u_q}{\log \frac{e^{u_q}(e^{u_q}+\upsilon)}{1+e^{u_q}\upsilon}} = \frac{2\log \frac{1-w_q}{w_q}}{\log \frac{1-2w_q+w_1}{w_1}}$$

where we note that $r \geq 2$ since $w_1 < 1 - 2w_q + w_1$. Now $\rho_q, \rho_u \geq 1/r + 1/s - 1$ is exactly $\log\left(\frac{w_1}{1-2w_q+w_1}\frac{1-w_q}{w_q}\right) / \log\left(\frac{1-w_q}{w_q}\right)$, matching Example 1 in the upper bounds as we wanted.

Optimal choice of r and s The goal is to maximize $\alpha/r + 1/s$ conditioned on $\sqrt{\sigma\upsilon} = \frac{w_1 - w_q w_u}{\sqrt{w_q(1-w_1)w_u(1-w_u)}}$ thus maximizing both ρ_q and ρ_u. To make things easier, we substitute $r \mapsto 1/(1-r')$ and $s \mapsto 1/(1-s')$. Then we have $\frac{d\upsilon}{dr'} = \frac{u_q \sinh(u_q)}{\sinh(u_q(1-r'))^2}$ and likewise for σ and s'. By Lagrange multipliers we get the two equations, for some $\lambda \in \mathbb{R}$: $\frac{\lambda\sigma u_q \sinh(u_q)}{\sinh(u_q(1-r'))^2} = \alpha$, $\quad \frac{\lambda\upsilon u_u \sinh(u_u)}{\sinh(u_u(1-s'))^2} = 1$. Dividing through gives the condition:

$$\frac{\sigma u_q \sinh(u_q)\sinh(u_u s')\sinh(u_u(1-s'))}{\upsilon u_u \sinh(u_u)\sinh(u_q r')\sinh(u_q(1-r'))} = \alpha.$$

We can insert this in the theorem to get the optimal r, s for any α on the trade-off. Because of the $r, s \geq 2$ condition, this is not always possible to achieve, but when it is Figure 2 suggests that the lower bound is tight when this condition can be met and further $w_q = w_u$.

Wolff [49] has shown how to extend the p-biased hypercontractive inequality beyond $r, s \geq 2$. However his work is only asymptotic. From the plots it is also clear that for $w_q \neq w_u$ Theorem 3 is not sharp. It thus seems evident that we need new methods. In the next section we will investigate a new two-function hypercontractive inequality for this purpose.

3.3.2 Lower Bound 2

We conjecture a new hypercontractive inequality:

Conjecture 2 (Two-Function p-Biased Hypercontractivity Inequality). *For $0 < w_q w_u \leq w \leq w_q, w_u < 1$, Let $\psi : \{-1,1\}^2 \to [0,1]$ be the joint probability density function $\sim \begin{bmatrix} w & w_u - w \\ w_q - w & 1 - w_q - w_u + w \end{bmatrix}$.*

For any pair of Boolean functions $f, g : \{-1,1\}^n \to \{0,1\}$ then

$$E_{x,y\sim\psi}[f(x)g(y)] \leq \|f\|_r \|g\|_s \quad (\textit{if } w \geq w_q w_u)$$
$$E_{x,y\sim\psi}[f(x)g(y)] \geq \|f\|_r \|g\|_s \quad (\textit{if } w \leq w_q w_u)$$

where $r = s = \log \frac{(1-w_q)(1-w_u)}{w_q w_u} \Big/ \log \frac{1-w_q-w_u+w}{w}$.

We reduce it to a simple two-variable inequality, from which Conjecture 2 and Conjecture 1 would follow. For this we will use the following inductive result by O'Donnell, which we have slightly generalized to support (x, y) from arbitrary shared distributions, rather than just ρ correlated. The proof in O'Donnell [38] goes through without changes.

Theorem 6 (Two-Function Hypercontractivity Induction Theorem [38]). *Assume that*

$$\mathop{\mathrm{E}}_{(x,y)\sim\pi}[f(x)g(y)] \leq \|f\|_r \|g\|_q$$

holds for every $f, g \in L^2(\Omega, \pi)$. Then the inequality also holds for every $f, g \in L^2(\Omega^n, \pi^n)$.

This means we just have to show a certain 'two point' inequality. That is, we would like the following to be true:

Lemma 3.2. *For $0 < w_q w_u \leq w \leq w_q, w_u < 1$, and any $f_{-1}, f_1, g_{-1}, g_1 \in \mathbb{R}$, then*

$$\begin{aligned} &f_{-1}g_{-1}w + f_{-1}g_1(w_q - w) + f_1 g_{-1}(w_u - w) + f_1 g_1(1 - w_q - w_u + w) \\ &\qquad \leq (w_q f_{-1}^r + (1-w_q) f_1^r)^{1/r} (w_u g_{-1}^s + (1-w_u) g_1^s)^{1/s} \end{aligned}$$

for $r = s = \log \frac{(1-w_q)(1-w_u)}{w_q w_u} \Big/ \log \frac{1-w_q-w_u+w}{w}$.

If $w \leq w_q w_u$ then the inequality goes the other direction.

Unfortunately we don't have a proof of this. However computer optimization suggests that it is true at least up to an error of 10^{-14}. Equality is achieved when $f_{-1}/f_1 = g_{-1}/g_1$ are either 1 or $(1 - w_q - w_u + w)/w$, and in these points the gradient match, which suggests the choice of r, s is sharp.

Dividing through, we may assume that $f(1)$ and $g(1)$ are both either 0 or 1. (If the values are negative, we can bound LSH by the positive versions. Rhs doesn't care.) If $g(1) = 0$ we just have to show

$$f(-1)g(-1)w + g(-1)(w_u - w) \leq w_q^{1/r} f(-1) w_u^{1/s} g(-1)$$

which follows from the one-function Hypercontractive Inequality.

Otherwise we can focus on proving

$$\begin{aligned}xyw + x(w_q - w) + y(w_u - w) + (1 - w_q - w_u + w)\\ \le (w_q x^r + 1 - w_q)^{1/r}(w_u y^r + 1 - w_u)^{1/r}\end{aligned}$$

where we defined $x = f(-1)/f(1), y = g(-1)/g(1)$. This is exactly eq. (1).

Assuming now Conjecture 2, lower bound 2 (Conjecture 1) follows from the discussion around eq. (9).

4 Conclusion

We show new matching upper and lower bounds for Set Similarity in the symmetric setting, $w_q = w_u$. We also show strong evidence that our upper bound is optimal in the asymmetric setting, $w_q \neq w_u$, as well as in the time-space trade-offs. If the lower bounds can be extended, this would unify the approaches between sparse and vectors on the sphere, and finally allow grand old MinHash to retire (from data structures, we don't make any claims about sketching).

4.1 Open problems

More lower bounds Besides proving Conjecture 1 it would be useful to extend it to the entire space/time trade-off. This would entail expanding the work of [49] to give a complete hypercontractive inequality for p-biased spaces without unknown universal constants.

Data dependent Data dependent LSH is able to reduce approximate similarity search problems to the case where far points are as far away as had they been random. For (w_q, w_u, w_1, w_2)-GapSS this corresponds to the case $w_2 = w_q w_u$. This would finally give the "optimal" algorithm for GapSS without any "non-data dependent" disclaimers.

Sparse, non-binary data We now know that threshold functions do well on sparse binary data and dense data on the sphere. In practice the most common type of data is perhaps sparse data on the sphere. Finding a way to give interesting results about this case is an interesting direction for research.

Sketching We have shown that Supermajorities can shave large polynomial factors of space and query time in LSH. Can they be used to give similar gains in the field of sketching sets under various similarity measures? Can one expand the work of [41] and show optimality of some intersection sketching scheme?

4.2 Acknowledgements

We would like to thank Rasmus Pagh and Tobias Christiani on suggesting the problem discussed in this paper, as well as for reviews of some previous manuscripts. Much of the initial work in the paper was done while the first author was visiting Eric Price at University of Texas, and we would like to thank people there for encouragement and discussions on Boolean functions. Many people have helped reading versions of the manuscript, and wee would like to thank Morgan Mingle, Morten Stöckel, John Kallaugher, Ninh Pham, Evangelos Kipouridis and everyone else who has given feedback. Finally we really appreciate the comprehensive comments from the anonymous reviewers!

References

[1] Amir Abboud, Aviad Rubinstein, and Ryan Williams. Distributed pcp theorems for hardness of approximation in p. In *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 25–36. IEEE, 2017.

[2] Parag Agrawal, Arvind Arasu, and Raghav Kaushik. On indexing error-tolerant set containment. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of data*, pages 927–938. ACM, 2010.

[3] Thomas Dybdahl Ahle, Rasmus Pagh, Ilya Razenshteyn, and Francesco Silvestri. On the complexity of inner product similarity join. In *Proceedings of the 35th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*, pages 151–164. ACM, 2016.

[4] Josh Alman, Timothy M Chan, and Ryan Williams. Polynomial representations of threshold functions and algorithmic applications. In *2016 IEEE 57th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 467–476. IEEE, 2016.

[5] Alexandr Andoni, Piotr Indyk, Thijs Laarhoven, Ilya Razenshteyn, and Ludwig Schmidt. Practical and optimal lsh for angular distance. In *Advances in Neural Information Processing Systems*, pages 1225–1233, 2015.

[6] Alexandr Andoni, Piotr Indyk, Huy L Nguyen, and Ilya Razenshteyn. Beyond locality-sensitive hashing. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1018–1028. Society for Industrial and Applied Mathematics, 2014.

[7] Alexandr Andoni, Thijs Laarhoven, Ilya Razenshteyn, and Erik Waingarten. Optimal hashing-based time-space trade-offs for approximate near neighbors. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 47–66. Society for Industrial and Applied Mathematics, 2017.

[8] Alexandr Andoni, Thijs Laarhoven, Ilya Razenshteyn, and Erik Waingarten. Optimal hashing-based time-space trade-offs for approximate near neighbors. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 47–66. SIAM, 2017.

[9] Alexandr Andoni and Ilya Razenshteyn. Optimal data-dependent hashing for approximate near neighbors. In *Proceedings of the Forty-Seventh Annual ACM Symposium on Theory of Computing*, pages 793–801. ACM, 2015.

[10] Alexandr Andoni, Ilya Razenshteyn, and Negev Shekel Nosatzki. Lsh forest: Practical algorithms made theoretical. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 67–78. SIAM, 2017.

[11] Roger C Baker, Glyn Harman, and János Pintz. The difference between consecutive primes, ii. *Proceedings of the London Mathematical Society*, 83(3):532–562, 2001.

[12] Anja Becker, Léo Ducas, Nicolas Gama, and Thijs Laarhoven. New directions in nearest neighbor searching with applications to lattice sieving. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 10–24. SIAM, 2016.

[13] John D Biggins. Martingale convergence in the branching random walk. *Journal of Applied Probability*, 14(1):25–37, 1977.

[14] Andrei Z Broder. On the resemblance and containment of documents. In *Compression and Complexity of Sequences 1997. Proceedings*, pages 21–29. IEEE, 1997.

[15] Andrei Z Broder, Steven C Glassman, Mark S Manasse, and Geoffrey Zweig. Syntactic clustering of the web. *Computer Networks and ISDN Systems*, 29(8-13):1157–1166, 1997.

[16] Timothy M Chan. Orthogonal range searching in moderate dimensions: kd trees and range trees strike back. In *33rd International Symposium on Computational Geometry (SoCG 2017)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2017.

[17] Moses Charikar, Piotr Indyk, and Rina Panigrahy. New algorithms for subset query, partial match, orthogonal range searching, and related problems. In *International Colloquium on Automata, Languages, and Programming*, pages 451–462. Springer, 2002.

[18] Moses S Charikar. Similarity estimation techniques from rounding algorithms. In *Proceedings of the thiry-fourth annual ACM Symposium on Theory of Computing*, pages 380–388. ACM, 2002.

[19] Lijie Chen and Ryan Williams. An equivalence class for orthogonal vectors. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 21–40. SIAM, 2019.

[20] Flavio Chierichetti and Ravi Kumar. Lsh-preserving functions and their applications. *Journal of the ACM (JACM)*, 62(5):33, 2015.

[21] Seung-Seok Choi, Sung-Hyuk Cha, and Charles C Tappert. A survey of binary similarity and distance measures. *Journal of Systemics, Cybernetics and Informatics*, 8(1):43–48, 2010.

[22] Tobias Christiani. A framework for similarity search with space-time tradeoffs using locality-sensitive filtering. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 31–46. SIAM, 2017.

[23] Tobias Christiani and Rasmus Pagh. Set similarity search beyond minhash. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017, Montreal, QC, Canada, June 19-23, 2017*, pages 1094–1107, 2017. URL: `https://doi.org/10.1145/3055399.3055443`, `doi:10.1145/3055399.3055443`.

[24] Richard Cole, Lee-Ad Gottlieb, and Moshe Lewenstein. Dictionary matching and indexing with errors and don't cares. In *Proceedings of the thirty-sixth annual ACM symposium on Theory of computing*, pages 91–100. ACM, 2004.

[25] Raul Castro Fernandez, Jisoo Min, Demitri Nava, and Samuel Madden. Lazo: A cardinality-based method for coupled estimation of jaccard similarity and containment. In *ICDE*, 2019.

[26] Philippe Flajolet, Éric Fusy, Olivier Gandouet, and Frédéric Meunier. Hyperloglog: the analysis of a near-optimal cardinality estimation algorithm. In *Discrete Mathematics and Theoretical Computer Science*, pages 137–156. Discrete Mathematics and Theoretical Computer Science, 2007.

[27] Ashish Goel and Pankaj Gupta. Small subset queries and bloom filters using ternary associative memories, with applications. *ACM SIGMETRICS Performance Evaluation Review*, 38(1):143–154, 2010.

[28] Sariel Har-Peled, Piotr Indyk, and Rajeev Motwani. Approximate nearest neighbor: Towards removing the curse of dimensionality. *Theory of computing*, 8(1):321–350, 2012.

[29] Piotr Indyk and Rajeev Motwani. Approximate nearest neighbors: towards removing the curse of dimensionality. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 604–613. ACM, 1998.

[30] TS Jayram, Subhash Khot, Ravi Kumar, and Yuval Rabani. Cell-probe lower bounds for the partial match problem. *Journal of Computer and System Sciences*, 69(3):435–447, 2004.

[31] Lianyin Jia, Lulu Zhang, Guoxian Yu, Jinguo You, Jiaman Ding, and Mengjuan Li. A survey on set similarity search and join. *International Journal of Performability Engineering*, 14(2), 2018.

[32] Michael Kapralov. Smooth tradeoffs between insert and query complexity in nearest neighbor search. In *Proceedings of the 34th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*, pages 329–342. ACM, 2015.

[33] Matti Karppa, Petteri Kaski, and Jukka Kohonen. A faster subquadratic algorithm for finding outlier correlations. *ACM Transactions on Algorithms (TALG)*, 14(3):31, 2018.

[34] Thijs Laarhoven. Tradeoffs for nearest neighbors on the sphere. *arXiv preprint arXiv:1511.07527*, 2015.

[35] Sergey Melnik and Hector Garcia-Molina. Adaptive algorithms for set containment joins. *ACM Transactions on Database Systems (TODS)*, 28(1):56–99, 2003.

[36] Rajeev Motwani, Assaf Naor, and Rina Panigrahi. Lower bounds on locality sensitive hashing. In *Proceedings of the twenty-second annual symposium on Computational geometry*, pages 154–157. ACM, 2006.

[37] Behnam Neyshabur and Nathan Srebro. On symmetric and asymmetric lshs for inner product search. In *International Conference on Machine Learning*, pages 1926–1934, 2015.

[38] Ryan O'Donnell. *Analysis of boolean functions*. Cambridge University Press, 2014.

[39] Krzysztof Oleszkiewicz. On a nonsymmetric version of the khinchine-kahane inequality. In *Stochastic inequalities and applications*, pages 157–168. Springer, 2003.

[40] Ryan O'Donnell, Yi Wu, and Yuan Zhou. Optimal lower bounds for locality-sensitive hashing (except when q is tiny). *ACM Transactions on Computation Theory (TOCT)*, 6(1):5, 2014.

[41] Rasmus Pagh, Morten Stöckel, and David P Woodruff. Is min-wise hashing optimal for summarizing set intersection? In *Proceedings of the 33rd ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 109–120. ACM, 2014.

[42] Rina Panigrahy. Entropy based nearest neighbor search in high dimensions. In *Proceedings of the seventeenth annual ACM-SIAM symposium on Discrete algorithm*, pages 1186–1195. Society for Industrial and Applied Mathematics, 2006.

[43] Rina Panigrahy, Kunal Talwar, and Udi Wieder. A geometric approach to lower bounds for approximate near-neighbor search and partial match. In *2008 49th Annual IEEE Symposium on Foundations of Computer Science*, pages 414–423. IEEE, 2008.

[44] Karthikeyan Ramasamy, Jignesh M. Patel, Jeffrey F. Naughton, and Raghav Kaushik. Set containment joins: The good, the bad and the ugly. In *VLDB*, 2000.

[45] Ronald L Rivest. Partial-match retrieval algorithms. *SIAM Journal on Computing*, 5(1):19–50, 1976.

[46] Aviad Rubinstein. Hardness of approximate nearest neighbor search. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*, pages 1260–1268. ACM, 2018.

[47] Anshumali Shrivastava and Ping Li. Asymmetric lsh (alsh) for sublinear time maximum inner product search (mips). In *Advances in Neural Information Processing Systems*, pages 2321–2329, 2014.

[48] Kengo Terasawa and Yuzuru Tanaka. Spherical lsh for approximate nearest neighbor search on unit hypersphere. In *Workshop on Algorithms and Data Structures*, pages 27–38. Springer, 2007.

[49] Paweł Wolff. Hypercontractivity of simple random variables. *Studia Mathematica*, 3(180):219–236, 2007.

[50] Xiao Yan, Jinfeng Li, Xinyan Dai, Hongzhi Chen, and James Cheng. Norm-ranging lsh for maximum inner product search. In *Advances in Neural Information Processing Systems*, pages 2956–2965, 2018.

A Locality Sensitive Filters

The Locality Sensitive Filter approach to similarity search is an extension by Becker et al. [12] to the Locality Sensitive Hashing framework by Indyk and Motwani [29]. We will use the following definition by Christiani [22], which we have slightly extended to support separate universes for query and data points:

Definition 3 (LSF). *Let X and Y be some universes, let $S : X \times Y \to \mathbb{R}$ be a similarity function, and let $\mathcal{F}$ be a probability distribution over $\{(Q, U) \mid Q \subseteq X, U \subseteq Y\}$. We say that F is $(s_1, s_2, p_1, p_2, p_q, p_u)$-sensitive if for all points $x \in X, y \in Y$ and (Q, U) sampled randomly from $\mathcal{F}$ the following holds:*

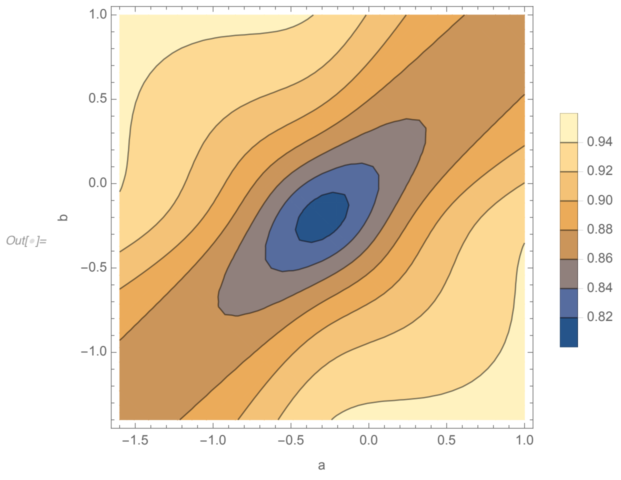
1. *If $S(x, y) \geq s_1$ then $\Pr[x \in Q, y \in U] \geq p_1$.*
2. *If $S(x, y) \leq s_2$ then $\Pr[x \in Q, y \in U] \leq p_2$.*
3. *$\Pr[x \in Q] \leq p_q$ and $\Pr[x \in U] \leq p_u$.*

We refer to (Q, U) as a filter and to Q as the query filter and U as the update filter.

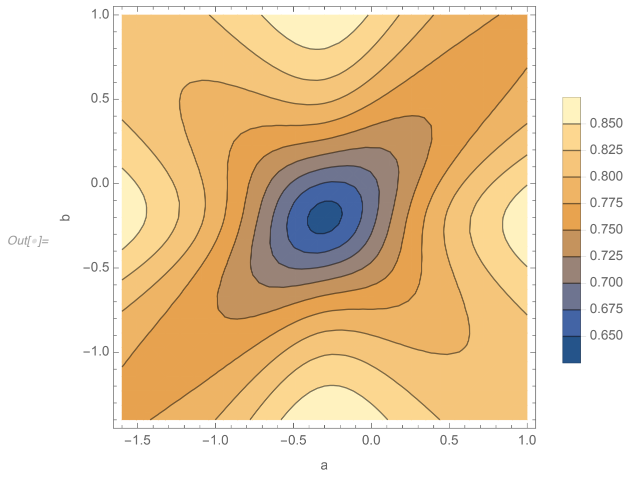
The main theorem from the same article gives a standard algorithm with the following guarantees (paraphrasing):

Theorem 7 (LSF theorem). *Suppose we have access to a family of filters that is $(s_1, s_2, p_1, p_2, p_q, p_u)$-sensitive. Then we can construct a fully dynamic data structure that solves the (s_1, s_2)-similarity search problem with query time $dn^{\rho_q + o(1)}$, update time $dn^{\rho_u + o(1)}$, and space usage $dn + n^{1+\rho_u+o(1)}$ where $\rho_q = \log(p_q/p_1) / \log(p_q/p_2)$ and $\rho_u = \log(p_u/p_1) / \log(p_q/p_2)$.*

We must be able to sample, store, and evaluate filters from $\mathcal{F}$ in time $dn^{o(1)}$.



(a) ρ-hyperplane



(b) ρ-spherical

Figure 3: ρ-values for hyperplane and spherical LSH under different shifts.

We will use Definition 3 with $S(x, y) = |x \cap y|$. For given values of w_q and w_u, the (s_1, s_2)-similarity search problem then corresponds to the (w_u, w_q, w_1, w_2)-GapSS problem.

B Comparison to other methods

B.1 Embedding onto the Sphere

Lemma B.1 (Embedding Lemma). *Let $g, h : \{0, 1\}^d \to \mathbb{R}$ be function on the form $g(1) = a_1 x + b_1$ and $h(y) = a_2 y + b_2$. Let $\rho(x, y, y') = f(\alpha(x, y))/f(\alpha(x, y'))$ where $\alpha(x, y) = \langle x, y\rangle/\|x\|\|y\|$ be such that*

$$f(z) \geq 0, \quad \tfrac{d}{dz}\left((\pm 1 - z)\tfrac{d}{dz}\log f(z)\right) \geq 0 \quad \text{and} \quad \tfrac{d^3}{dz^3}\log f(z) \leq 0$$

for all $z \in [-1, 1]$. Assume we know that $\|x\|_2^2 = w_q d$, $\|y\|_2^2 = w_u d$, $\langle x, y'\rangle = w_1 d$ and $\langle x, y\rangle = w_2 d$, then $\arg\min_{a_1,b_1,a_2,b_2} \rho(g(x), h(y), h(y')) = (1, -w_q, 1, -w_u)$.

In this section we will show that Hyperplane [18] and Spherical [8] LSH both satisfy the requirements of the lemma. Hence we get two algorithms with ρ-values:

$$\rho_{\text{hp}} = \frac{\log(1 - \arccos(\alpha)/\pi)}{\log(1 - \arccos(\beta)/\pi)}, \quad \rho_{\text{sp}} = \frac{1-\alpha}{1+\alpha}\frac{1+\beta}{1-\beta}.$$

where $\alpha = \frac{w_1 - w_q w_u}{\sqrt{w_q(1-w_q)w_u(1-w_u)}}$ and $\beta = \frac{w_2 - w_q w_u}{\sqrt{w_q(1-w_q)w_u(1-w_u)}}$, and space/time trade-offs using the ρ_q, ρ_u values in [22]. [8] Figure 3 shows how ρ varies with different translations a, b.

Taking $t_q = w_q(1+o(1))$ and $t_u = w_u(1+o(1))$ in theorem 1 recovers ρ_{sp} by standard arguments. This implies that theorem 1 dominates Spherical LSH (for binary data).

Lemma B.2. *The functions $f(z) = (1 - z)/(1 + z)$ for Spherical LSH and $f(z) = -\log(1 - \arccos(z)/\pi)$ for Hyperplane LSH satisfy lemma B.1.*

[8]Unfortunately the space/time aren't on a form applicable to lemma B.1. From numerical experiments we however still conjecture that the embedding is optimal for those as well.

Proof. For Spherical LSH we have $f(z) = (1-z)/(1+z)$ and get

$$\begin{aligned}\tfrac{d}{dz}\left((\pm 1 - z)\tfrac{d}{dz}\log f(z)\right) &= 2(1 \mp 2z + z^2)/(1-z^2)^2 \geq 0,\\ \tfrac{d^3}{dz^3}\log f(z) &= -4(1+3z^2)/(1-z^2)^3 \leq 0.\end{aligned}$$

For Hyperplane LSH we have $f(z) = -\log(1 - \arccos(z)/\pi)$ and get

$$\begin{aligned}\tfrac{d}{dz}\left((1 - z)\tfrac{d}{dz}\log f(z)\right) &= \frac{(\arccos(z) - \sqrt{1-z^2} - \pi)\log(1-\arccos(z)/\pi) - \sqrt{1-z^2}}{(1+z)\sqrt{1-z^2}(\pi - \arccos(z))^2\log(1-\arccos(z)/\pi)^2},\\ \tfrac{d}{dz}\left((-1 - z)\tfrac{d}{dz}\log f(z)\right) &= \frac{(\arccos(z) + \sqrt{1-z^2} - \pi)\log(1-\arccos(z)/\pi) + \sqrt{1-z^2}}{(1-z)\sqrt{1-z^2}(\pi - \arccos(z))^2\log(1-\arccos(z)/\pi)^2}.\end{aligned}$$

In both cases the denominator is positive, and the numerator can be shown to be likewise by applying the inequalities $\sqrt{1-z^2} \leq \arccos(z)$, $\sqrt{1-z^2} + \arccos(z) \leq \pi$ and $x \leq \log(1+x)$.

The $\frac{d^3}{dz^3}\log f(z) \leq 0$ requirement is a bit trickier, but a numerical optimization shows that it's in fact less than -1.53. □

Finally we prove the embedding lemma:

Proof of lemma B.1. We have

$$\alpha = \frac{\langle x+a, y+b\rangle}{\|x+a\|\|y+b\|} = \frac{w_1 + w_q b + w_u a + ab}{\sqrt{(w_q(1+a)^2 + (1-w_q)a^2)(w_u(1+b)^2 + (1-w_u)b^2)}}$$

and equivalent with w_2 for β. We'd like to show that $a = -w_q$, $b = -w_u$ is a minimum for $\rho = f(\alpha)/f(\beta)$.

Unfortunately the f's we are interested in are usually not convex, so it is not even clear that there is just one minimum. To proceed, we make the following substitution $a \to (c+d)\sqrt{w_q(1-w_q)} - w_q$, $b \to (c-d)\sqrt{w_u(1-w_u)} - w_u$ to get

$$\alpha(c,d) = \frac{cd + \frac{w_1 - w_q w_u}{\sqrt{w_q(1-w_q)w_u(1-w_u)}}}{\sqrt{(1+c^2)(1+d^2)}}.$$

We can further substitute $cd \mapsto rs$ and $\sqrt{(1+c^2)(1+d^2)} \mapsto r+1$ or $r \geq 0$, $-1 \leq s \leq 1$, since $1 + cd \leq \sqrt{(1+c^2)(1+d^2)}$ by Cauchy Schwartz, and $(cd, \sqrt{(1+c^2)(1+d^2)})$ can take all values in this region.

The goal is now to show that $h = f\left(\frac{rs+x}{r+1}\right)/f\left(\frac{rs+y}{r+1}\right)$, where $1 \geq x \geq y \geq -1$, is increasing in r. This will imply that the optimal value for c and d is 0, which further implies that $a = -w_q$, $b = -w_u$ for the lemma.

We first show that h is quasi-concave in s, so we may limit ourselves to $s = \pm 1$. Note that $\log h = \log f\left(\frac{rs+x}{r+1}\right) - \log f\left(\frac{rs+y}{r+1}\right)$, and that $\frac{d^2}{ds^2}\log f\left(\frac{rs+x}{r+1}\right) = \left(\frac{r}{1+r}\right)^2 \frac{d^2}{dz^2}\log f(z)$ by the chain rule. Hence it follows from the assumptions that h is log-concave, which implies quasi-concavity as needed.

We now consider $s = \pm 1$ to be a constant. We need to show that $\frac{d}{dr}h \geq 0$. Calculating,

$$\frac{d}{dr} f\left(\frac{rs+x}{r+1}\right)/f\left(\frac{rs+y}{r+1}\right) = \frac{(s-x)f\left(\frac{rs+y}{r+1}\right)f'\left(\frac{rs+x}{r+1}\right) - (s-y)f\left(\frac{rs+x}{r+1}\right)f'\left(\frac{rs+y}{r+1}\right)}{(1+r)^2 f\left(\frac{rs+y}{r+1}\right)^2}.$$

Since $f \geq 0$ it suffices to show $\frac{d}{dx}(s-x)f'\left(\frac{rs+x}{r+1}\right)/f\left(\frac{rs+x}{r+1}\right) \geq 0$. If we substitute $z = \frac{rs+x}{r+1}$, $z \in [-1,1]$, we can write the requirement as $\frac{d}{dz}(s-z)f'(z)/f(z) \geq 0$ or $\frac{d}{dz}\left((\pm 1 - z)\frac{d}{dz}\log f(z)\right) \geq 0$. □

B.2 A MinHash dominating family

We assume some familiarity with LSF vocabulary. See appendix A for a short introduction.

We first state the LSF-Symmetrization lemma implicit in [23]:

Lemma B.3 (LSF-Symmetrization)**.** *Given a (p_1, p_2, p_q, p_u)-sensitive LSF-family, we can create a new family that is $(p_1 q/p, p_2 q/p, q, q)$-sensitive, where $p = \max\{p_q, p_u\}$ and $q = \min\{p_q, p_u\}$.*

For some values of p_1, p_2, p_q, p_u this will be better than simply taking $\max(\rho_u, \rho_q)$. In particular when symmetrization may reduce ρ_u by a lot by reducing its denominator.

Proof. W.l.o.g. assume $p_q \geq p_u$. When sampling a query filter, $Q \subseteq U$, pick a random number $\varrho \in [0,1]$. If $\varrho > p_u/p_q$ use $\emptyset$ instead of Q. The new family then has $p'_q = p_q \cdot p_u/p_q$ and so on giving the lemma. □

Using this lemma it is easy to make a version of supermajority LSF that always beats Chosen Path: Simply take $t_q = t_u = 1$ and apply lemma B.3. Then we have exactly the same ρ value as Chosen Path. We do however conjecture that symmetrization is not necessary for supermajorities, since we have another (presumably more efficient) form of symmetrization via asymmetric $t_u \neq t_q$.

Now recall the filter family from the introduction:

$$F_0(x) = [s_0 \in x], F_1(x) \quad = [s_1 \in x \wedge s_0 \notin x], \dots, F_i(x) = [s_i \in x \wedge s_0 \notin x \wedge \dots \wedge s_{i-1} \notin x], \dots$$

where $(s_i \in U)_{i \in \mathbb{N}}$ is a random sequence by sampling elements of U with replacement.

Using just one of these functions, combined with symmetrization, gives the ρ value:

$$\rho_i = \log \frac{(1-w_q-w_u+w_1)^i w_1}{\max\{(1-w_q)^i w_q,\, (1-w_u)^i w_u\}} \Big/ \log \frac{(1-w_q-w_u+w_2)^i w_2}{\max\{(1-w_q)^i w_q,\, (1-w_u)^i w_u\}}.$$

We want to show $\rho_{\text{mh}} = \log \frac{w_1}{w_q+w_u-w_1} \big/ \log \frac{w_2}{w_q+w_u-w_2} \geq \min_{i \geq 0} \rho_i$. For this we show the following lemma, which intuitively says that it is never advantageous to combine multiple filter families:

Lemma B.4. *The function $f(x,y,z,t) = \log(\max\{x,y\}/z)/\log(\max\{x,y\}/t)$, defined for $\min\{x,y\} \geq z \geq t > 0$, is quasi-concave.*

This means in particular that

$$\frac{\log(\max\{x+x', y+y'\}/(z+z'))}{\log(\max\{x+x', y+y'\}/(t+t'))} \geq \min\left\{\frac{\log(\max\{x,y\}/z)}{\log(\max\{x,y\}/t)}, \frac{\log(\max\{x',y'\}/z')}{\log(\max\{x',y'\}/t')}\right\},$$

when the variables are in the range of the lemma.

Proof. We need to show that the set

$$\{(x,y,z,t) : \log(\max\{x,y\}/z)/\log(\max\{x,y\}/t) \geq \alpha\} = \{(x,y,z,t) : \max\{x,y\}^{1-\alpha} t^{\alpha} \geq z\}$$

is convex for all $\alpha \in [0,1]$ (since $z \geq t$ so $f(x,y,z,t) \in [0,1]$). This would follow if $g(x,y,t) = \max\{x,y\}^{1-\alpha} t^{\alpha}$ would be quasi-concave itself, and the eigenvalues of the Hessian of g are exactly 0, 0 and $-(1-\alpha)\alpha t^{\alpha-2} \max\{x,y\}^{-\alpha-1}\left(\max\{x,y\}^2 + t^2\right)$ so g is even concave! □

We can then show that MinHash is always dominated by one of the filters described, as

$$\rho_{\mathrm{mh}} = \frac{\log \frac{w_1}{w_q+w_u-w_1}}{\log \frac{w_2}{w_q+w_u-w_2}} = \frac{\log \frac{\sum_{i\ge 0}(1-w_q-w_u+w_1)^i w_1}{\max\{\sum_{i\ge 0}(1-w_q)^i w_q, \sum_{i\ge 0}(1-w_u)^i w_u\}}}{\log \frac{\sum_{i\ge 0}(1-w_q-w_u+w_2)^i w_2}{\max\{\sum_{i\ge 0}(1-w_q)^i w_q, \sum_{i\ge 0}(1-w_u)^i w_u\}}} \ge \min_{i\ge 0} \frac{\log \frac{(1-w_q-w_u+w_1)^i w_1}{\max\{(1-w_q)^i w_q, (1-w_u)^i w_u\}}}{\log \frac{(1-w_q-w_u+w_2)^i w_2}{\max\{(1-w_q)^i w_q, (1-w_u)^i w_u\}}},$$

where the right hand side is exactly the symmetrization of the filters $F^{(i)}$. By monotonicity of $(1-w_q)^i w_q$ and $(1-w_u)^i w_u$ we can further argue that it is even possible to limit ourselves to one of $i \in \{0, \infty, \log(w_q/w_u)/\log((1-w_q)/(1-w_u))\}$, where the first gives Chosen Path, the second gives Chosen Path on the complemented sets, and the last gives a balanced trade-off where $(1-w_q)^i w_q = (1-w_u)^i w_u$.

C Plots

We provide more plots comparing different approaches to GapSS in Figure 4.

D Technical lemmas

Lemma D.1. *Let $k \in \mathbb{Z}_+$ and $p_1, p_2 \in [0,1]$, such that, both $p_1 k$ and $p_2 k$ are integers. Choose $p \in [0,1]$, such that, $p \ge p_1 p_2$. Let $X^{(i)} \in \mathbb{R}^2$ be independent identically distributed 2-dimensional Bernoulli variables, where their probability matrix is $P = \begin{pmatrix} p & p_1 - p \\ p_2 - p & 1 - p_1 - p_2 + p \end{pmatrix}$. We then get that*

$$\Pr\left[\sum_{i\in[k]} X^{(i)} = \left(\begin{smallmatrix} p_1 \\ p_2 \end{smallmatrix}\right) k \wedge \sum_{i\in[k]} X_1^{(i)} X_2^{(i)} = \lceil pk \rceil\right] \ge \frac{1}{400} k^{-3.5} .$$

In the proof we will be using the Stirling's approximation

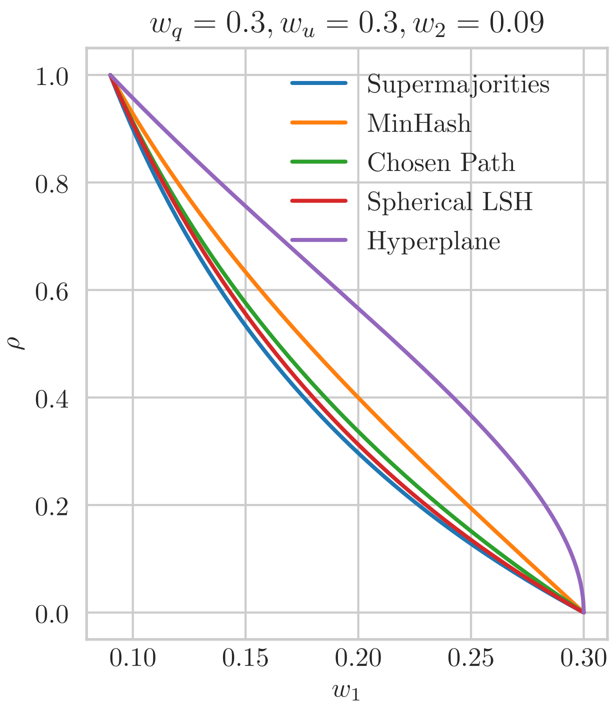
$$\sqrt{2\pi n} n^n e^{-n} \le n! \le e\sqrt{n} n^n e^{-n} .$$

This implies the following useful bounds on the binomial and multinomial coefficients.

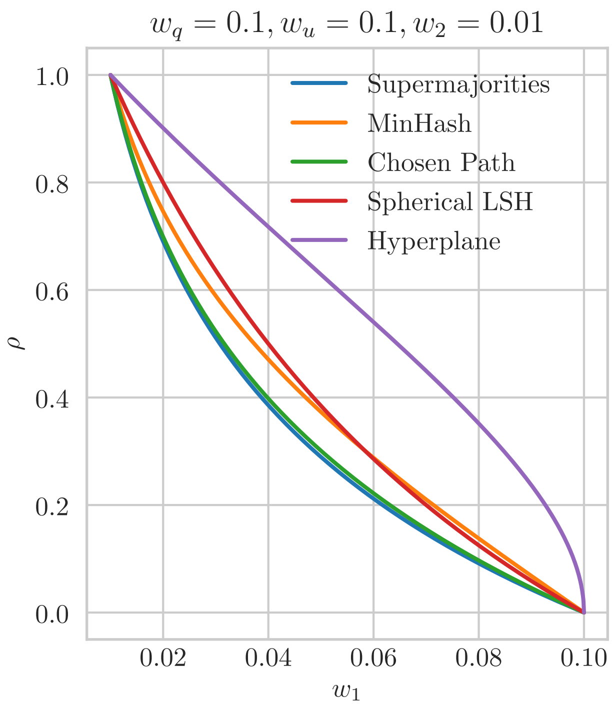
$$\begin{aligned} \binom{n}{an} &\ge \frac{\sqrt{2\pi}}{e^2} \cdot \frac{1}{\sqrt{a(1-a)n}} (an)^{an} ((1-a)n)^{(1-a)n} && (10) \\ &\ge \frac{\sqrt{2\pi}}{e^2} n^{-0.5} ((1-a)n)^{(1-a)n} . \\ \binom{n}{an, bn, cn} &\ge \frac{\sqrt{2\pi}}{e^4} \cdot \frac{1}{\sqrt{abc(1-a-b-c)n^3}} (an)^{an} (bn)^{bn} (cn)^{cn} ((1-a-b-c)n)^{(1-a-b-c)n} && (11) \\ &\ge \frac{\sqrt{2\pi}}{e^4} n^{-1.5} (an)^{an} (bn)^{bn} (cn)^{cn} ((1-a-b-c)n)^{(1-a-b-c)n} . \end{aligned}$$

Proof. If $p_1 = 1$ then $p = p_2$ and we get that

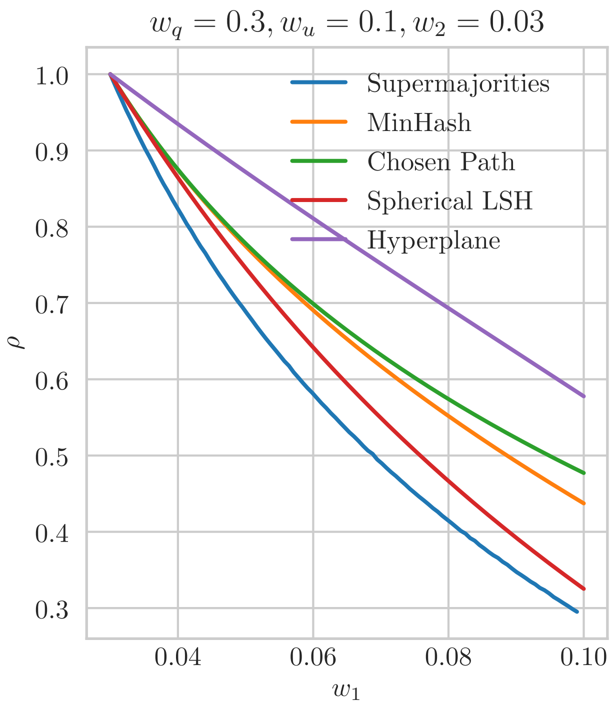
$$\Pr\left[\sum_{i\in[k]} X_2^{(i)} = p_2 k\right] = \binom{k}{p_2 k} p_2^{p_2 k} (1-p_2)^{(1-p_2)k} \ge \frac{\sqrt{2\pi}}{e^2} k^{-\frac{1}{2}} ,$$



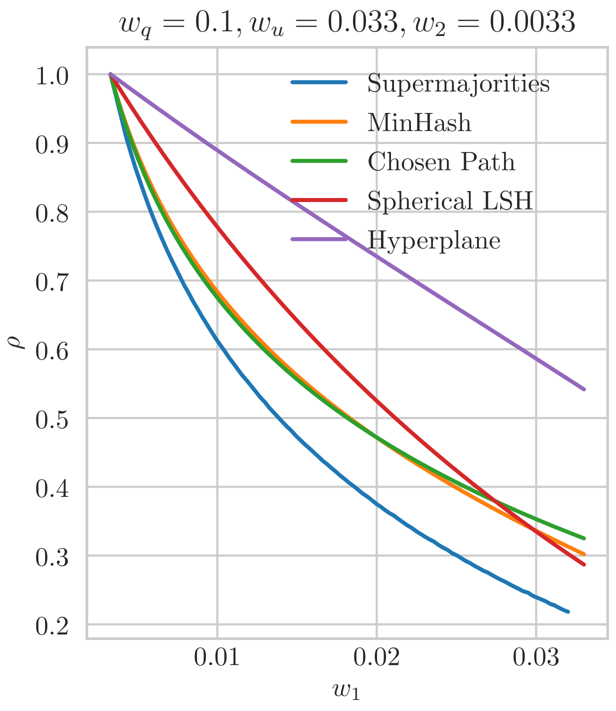
(a) Regular set sizes, relatively large. Note that Spherical LSH is proven optimal in this case, as the sets get large enough.



(b) Regular set sizes, relatively small. Note that Chosen Path is proven optimal in this case, as the sets get small enough.

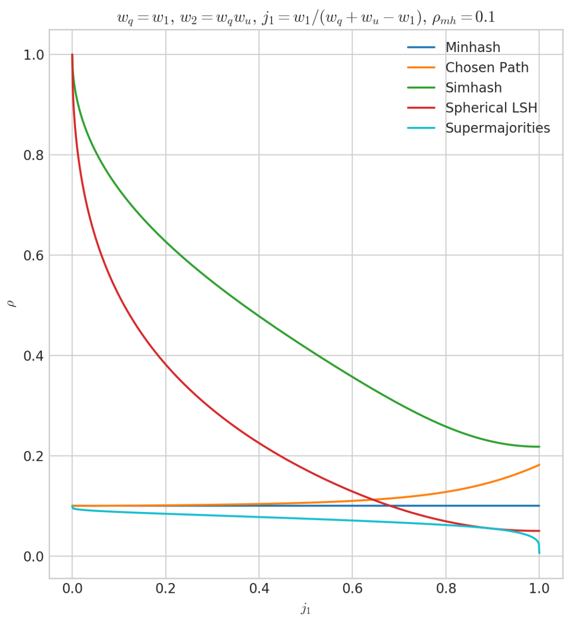


(c) Irregular set sizes, relatively large.

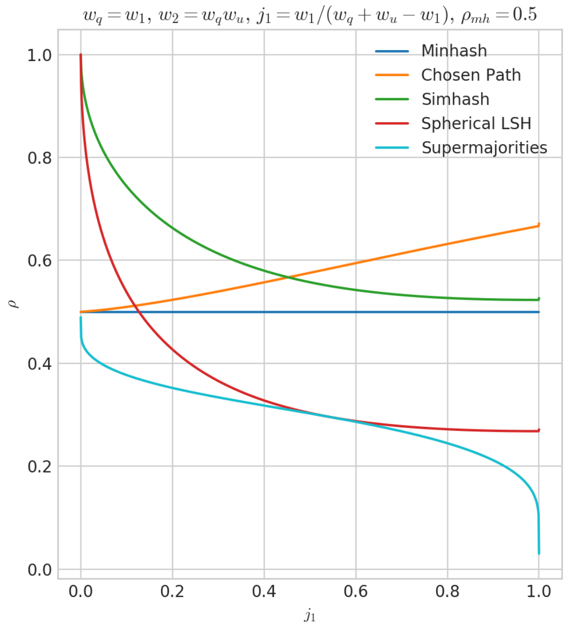


(d) Irregular set sizes, relatively small.

Figure 4: Examples of ρ-values obtained from Theorem 1 in the "balanced" case where query time equals update time, $n^{\rho+o(1)}$. The quantity ρ is plotted in various settings of (w_q, w_u, w_1, w_2)-GapSS, compared to that of other algorithms.



(a) Fixing $\rho_{mh} = 1/10$, $w_2 = w_q w_u$, $w_1 = w_q$ and varying $j_1 = w_1/(w_q + w_u - w_1)$ on the x-axis.



(b) Fixing $\rho_{mh} = 1/2$, $w_2 = w_q w_u$, $w_1 = w_q$ and varying $j_1 = w_1/(w_q + w_u - w_1)$ on the x-axis.

Figure 5: These plots show the ρ values of various LSH schemes for superset search ($w_q = w_1$) in the balanced case $\rho_q = \rho_u$. Fixing the ρ value of MinHash we see how other methods compare. In particular in the case where j_1 (the Jaccard similarity between subset and superset) is small, which happens when the sets themselves are small, we see the Spherical LSH approaches are nearly useless, while Supermajority quickly gains an edge.

where we have used eq. (10). We get the same bound when $p_1 = 0$, $p_2 = 1$, or $p_2 = 0$.

Now assume that $p_1, p_2 \notin \{0, 1\}$, we then have that $\frac{1}{k} \le p_1 \le 1 - \frac{1}{k}$ and $\frac{1}{k} \le p_2 \le 1 - \frac{1}{k}$. We first note that

$$\begin{aligned}
&\Pr\left[\sum_{i\in[k]} X^{(i)} = (p_1k, p_2k) \wedge \sum_{i\in[k]} X_1^{(i)} X_2^{(i)} = \lceil pk \rceil\right] \\
&\quad = \binom{k}{\lceil pk\rceil, p_1k - \lceil pk\rceil, p_2k - \lceil pk\rceil} p^{\lceil pk\rceil}(p_1-p)^{p_1k-\lceil pk\rceil}(p_2-p)^{p_2k-\lceil pk\rceil}(1-p_1-p_2+p)^{k-p_1k-p_2k+\lceil pk\rceil} \\
&\quad \ge \frac{\sqrt{2\pi}}{e^4}\cdot\frac{1}{k^{3/2}}\exp\Big(-\lceil pk\rceil\log\frac{\lceil pk\rceil}{pk} - (p_1k-\lceil pk\rceil)\log\frac{p_1k-\lceil pk\rceil}{p_1k-pk} \\
&\qquad - (p_2k-\lceil pk\rceil)\log\frac{p_2k-\lceil pk\rceil}{p_2k-pk} - (k-p_1k-p_2k+\lceil pk\rceil)\log\frac{k-p_1k-p_2k+\lceil pk\rceil}{k-p_1k-p_2k+pk}\Big)
\end{aligned}$$

where we have used eq. (11). We will bound each of the terms $\lceil pk\rceil\log\frac{\lceil pk\rceil}{pk}$, $(p_1k-\lceil pk\rceil)\log\frac{p_1k-\lceil pk\rceil}{p_1k-pk}$, $(p_2k-\lceil pk\rceil)\log\frac{p_2k-\lceil pk\rceil}{p_2k-pk}$, and $(k-p_1k-p_1k+\lceil pk\rceil)\log\frac{k-p_1k-p_2k+\lceil pk\rceil}{k-p_1k-p_2k+pk}$ individually.

Using that $p \ge p_1p_2 \ge \frac{1}{k^2}$ we get that

$$\lceil pk\rceil\log\frac{\lceil pk\rceil}{pk} = (1+pk)\log\left(1+\frac{1}{pk}\right) \le 1+\log(1+k)\ .$$

Now using that $1-p_1-p_2+p \ge (1-p_1)(1-p_2) \ge \frac{1}{k^2}$ we get that

$$\begin{aligned}
(k-p_1k-p_1k+\lceil pk\rceil)\log\frac{k-p_1k-p_2k+\lceil pk\rceil}{k-p_1k-p_2k+pk} &\le (k(1-p_1-p_2+p)+1)\log\left(1+\frac{1}{k(1-p_1-p_2+p)}\right) \\
&\le 1+\log(1+k)\ .
\end{aligned}$$

We easily get that

$$(p_1k-\lceil pk\rceil)\log\frac{p_1k-\lceil pk\rceil}{p_1k-pk} \le (p_1k-\lceil pk\rceil)\log\frac{p_1k-pk}{p_1k-pk} = 0\ .$$

Similarly, we get that $(p_2k-\lceil pk\rceil)\log\frac{p_2k-\lceil pk\rceil}{p_2k-pk} = 0$.

Combining all this we get that

$$\begin{aligned}
\Pr\left[\sum_{i\in[k]} X^{(i)} = (p_1k, p_2k) \wedge \sum_{i\in[k]} X_1^{(i)}X_2^{(i)} = \lceil pk\rceil\right] &\ge \frac{\sqrt{2\pi}}{e^4}k^{-1.5}\exp(-(1+\log(1+k)) - (1+\log(1+k))) \\
&\ge \frac{1}{400}k^{-3.5}\ .
\end{aligned}$$

□

Lemma D.2. *Let $k \in \mathbb{Z}_+$ and $p, p_1, p_2 \in [0,1]$, such that, pk, p_1k, and p_2k are integers and $p \ge p_1p_2$. Let $X^{(i)} \in \{0,1\}^2$ be independent identically distributed variables. We then get that*

$$\Pr\left[\forall l \le k : \sum_{i\in[k]} X^{(i)} \ge \begin{pmatrix}p_1\\p_2\end{pmatrix} l \,\middle|\, \sum_{i\in[k]} X^{(i)} = \begin{pmatrix}p_1\\p_2\end{pmatrix} k \wedge \sum_{i\in[k]} X_1^{(i)}X_2^{(i)} = pk\right] \ge k^{-3}\ .$$

In the proof we will use the folklore result.

Lemma D.3. *Let $k \in Z_+$ and $(a_i)_{i\in[k]}$ numbers such that $\sum_{i\in[k]} a_i \geq 0$ then there exists a $s \in [k]$ such that $\sum_{i\in[l]} a_{(s+i) \mod k} \geq 0$ for every $l \leq k$.*

Proof. Using Lemma D.3 we get that $\sum_{i\in[l]} X_1^{(i)} \geq p_1 l$ for every $l \leq k$ with probability at least k^{-1} since every variable identically distributed. Fixing $(X_1^{(i)})_{i\in[k]}$ and using Lemma D.3 2 times we get that $\sum_{i\in[l]} X_1^{(i)} X_2^{(i)} \geq \frac{p}{p_1} \sum_{i\in[l]} X_1^{(i)}$ and $\sum_{i\in[l]}(1 - X_1^{(i)})X^{(i)} \geq \frac{p_2-p}{1-p_1} \sum_{i\in[l]} X_1^{(i)}$ for every $l \leq k$ with probability at least k^{-2}. If all these three events happens then for every $l \leq k$ we get that

$$\begin{aligned}
\sum_{i\in[l]} X_2^{(i)} &= \sum_{i\in[l]} X_1^{(i)} X_2^{(i)} + \sum_{i\in[l]} (1 - X_1^{(i)}) X_2^{(i)} \\
&\geq \frac{p}{p_1} \sum_{i\in[l]} X_1^{(i)} + \frac{p_2 - p}{1 - p_1} \sum_{i\in[l]} X_1^{(i)} \\
&= \frac{p - p_1 p_2}{p_1(1 - p_1)} \sum_{i\in[l]} X_1^{(i)} + \frac{p_2 - p}{1 - p_1} l \\
&\geq \frac{p - p_1 p_2}{p_1(1 - p_1)} p_1 l + \frac{p_2 - p}{1 - p_1} l \\
&= p_2 l \ .
\end{aligned}$$

So we conclude that with probability at least k^{-3} then $\sum_{i\in[l]} X^{(i)} \geq \begin{pmatrix} p_1 \\ p_2 \end{pmatrix} l$ for every $l \leq k$ which finishes the proof. ☐

Lemma D.4. *The sequence $(\Delta_i)_{i\in\mathbb{N}}$ is defined to be the smallest positive integer such that $\prod_{j=1}^{i} \Delta_j \geq \Delta^i$ for every $i \in \mathbb{N}$. One can show that if $\Delta \geq 1$ then $\prod_{j=1}^{i} \Delta_j < 2\Delta^i$.*

Proof. The proof is by induction on i. The result is trivially true for $i = 1$. Now assume that the result is true for i. We then have to cases. Either $\prod_{j=1}^{i} \Delta_j \geq \Delta^{i+1}$ but then $\Delta_{i+1} = 1$ and $\prod_{j=1}^{i+1} \Delta_j = \prod_{j=1}^{i} \Delta_j < 2\Delta^i \leq 2\Delta^{i+1}$. Otherwise we have that $\prod_{j=1}^{i} \Delta_j < \Delta^{i+1}$ but then we have that $\prod_{j=1}^{i+1} \Delta_j \leq \Delta^{i+1} + \prod_{j=1}^{i} \Delta_j < 2\Delta^{i+1}$ since Δ_{i+1} is chosen to be the smallest integer satisfying the relation. ☐