Building Self-Clustering RDF Databases Using Adaptive-LSH

ABSTRACT

The Resource Description Framework (RDF) is a W3C standard for representing graph-structured data, and SPARQL is the standard query language for RDF. Recent advances in Information Extraction, Linked Data Management and the Semantic Web have led to a rapid increase in both the volume and the variety of RDF data that are publicly available. As businesses start to capitalize on RDF data, RDF data management systems are being exposed to workloads that are far more diverse and dynamic than what they were designed to handle. Consequently, there is a growing need for developing workload-adaptive and self-tuning RDF data management systems. To realize this objective, we introduce a fast and efficient method for dynamically clustering records in an RDF data management system. Specifically, we assume nothing about the workload upfront, but as SPARQL queries are executed, we keep track of records that are co-accessed by the queries in the workload and physically cluster them. To decide dynamically (and, in constant-time) where a record needs to be placed in the storage system, we develop a new locality-sensitive hashing (LSH) scheme, ADAPTIVELSH. Using ADAPTIVELSH, records that are co-accessed across similar sets of queries can be hashed to the same or nearby physical pages in the storage system. What sets ADAPTIVELSH apart from existing LSH schemes is that it can auto-tune to achieve the aforementioned clustering objective with high accuracy even when the workloads change. Experimental evaluation of ADAPTIVELSH in our prototype RDF data management system, *System-XYZ* (name withheld due to double-blind), as well as in a standalone hashtable shows significant end-to-end improvements over existing solutions.

1. INTRODUCTION

Physical data organization plays an important role in the performance tuning of database management systems. A particularly important problem is clustering (in the storage system) records that are frequently co-accessed by queries in a workload. Suboptimal clustering has negative performance implications due to random I/O and cache stalls [8]. This problem has received attention in the context of SQL databases and has led to the introduction of tuning advisors that work either in an *offline* [7, 67] or *online* fashion (i.e., self-tuning databases) [28].

In this paper, we address the problem in the context of RDF data management systems. Due to the diversity of applications on the World Wide Web, SPARQL workloads that RDF data management systems service are far more dynamic than conventional SQL workloads [15,49]. First of all, on the Web, queries are influenced by real-life events, which can be highly unpredictable [15, 49, 62]. Second, hotspots in RDF, which denote the RDF resources that are frequently queried, can have fluctuating phases of popularity. For example, an analysis over real SPARQL query logs reveal that in one week intervals before, during and after a conference, the popularity of the RDF resources related to that conference can peak and then drop significantly [49]. Third, our attempts at modeling these fluctuations over a collection of real SPARQL workloads [18] using a measure called *volatility* [34]) have revealed that the volatility of RDF resources follow normal-like distributions [61]. That is, while some RDF resources are queried frequently and this frequency does not fluctuate much over time (i.e., low volatility), for others this fluctuation can be very high (i.e., high volatility).

For the second class of queries with high volatility, tuning techniques for RDF data management systems are in their infancy, and relational solutions are not directly applicable. More specifically, depending on the workload, it might be necessary to completely change the underlying physical representation in an RDF data management system, such as by dynamically switching from a row-oriented representation to a columnar representation [12]. On the other hand, existing solutions are either *offline* requiring the workload to be known upfront [37, 41], or *online*, in which, the tuning techniques work well only when the schema changes are minor [24]. Consequently, with the increasing demand to support highly dynamic workloads in RDF [15, 49], there is a growing need to develop more adaptive tuning solutions, in which records in an RDF database can be dynamically and continuously clustered based on the current workload.

Whenever a SPARQL query is executed, there is an opportunity to observe how records in an RDF database are being utilized. This information about query access patterns can be used to dynamically cluster records in the storage system. The ability to do this dynamically is important in RDF systems because of the high variability and dynamism

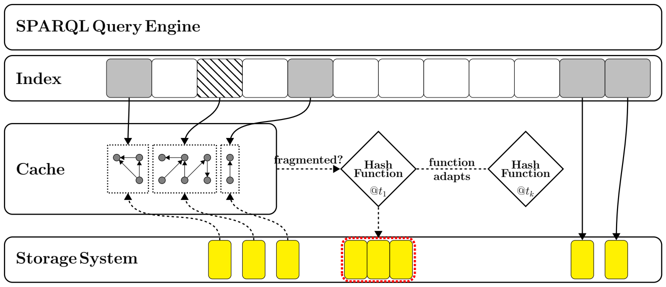


Figure 1: Adaptive record placement using a combination of adaptive hashing and caching.

present in SPARQL workloads [15, 49]. While this problem has been studied as physical clustering [53] and distribution design [26], the highly dynamic nature of the queries over RDF data introduces new challenges. First, traditional algorithms are offline, and since clustering is an NP-hard problem where most approximations have quadratic complexity [48], they are not suitable for online database clustering. Instead, techniques are needed with similar clustering objectives, but that have linear running time. Second, systems are typically expected to execute most queries in subseconds [56], leaving only fractions of a second to update their physical data structures (e.g., dynamically moving records across the storage system).

We address the aforementioned issues by making two contributions. First, as shown in Fig. 1, instead of clustering the whole database, we cluster only the "warm" portions of the database by relying on the admission policy of the existing database cache. Second, we develop a self-tuning locality-sensitive hash (LSH) function, namely, AdaptiveLSH to decide in constant-time where in the storage system to place a record. AdaptiveLSH has two important properties: First, it tries to ensure that (i) records with *similar* utilization patterns (i.e., those records that are co-accessed across similar sets of queries) are mapped as much as possible to the same pages in the storage system, while (ii) minimizing the number of records with *dissimilar* utilization patterns that are falsely mapped to the same page. Second, unlike conventional LSH [36,46], AdaptiveLSH can auto-tune so as to achieve the aforementioned clustering objectives with high accuracy even when the workload changes.

These ideas are illustrated in Fig. 1. Let us assume that initially, the records in a database are *not* clustered according to any particular workload. Therefore, the performance of the system is suboptimal. However, every time records are fetched from the storage system, there is an opportunity to bring together into a single page those records that are co-accessed but are fragmented across the storage system. AdaptiveLSH achieves these with minimal overhead. Furthermore, AdaptiveLSH is continuously updated to reflect any changes in the workload characteristics. Consequently, as more queries are executed, records in the database become more clustered, thereby, improving performance.

The paper is organized as follows: Section 2 discusses related work. Section 3 gives a conceptual description of the problem. Section 4 describes the overview of our approach while Section 5 provides the details. In Section 6, we describe how physical clustering takes place in the database, in particular, how AdaptiveLSH can be used in an RDF data management system, and we experimentally evaluate our techniques. Finally, we discuss conclusions and future work in Section 7.

2. RELATED WORK

Locality-sensitive hashing (LSH) [36, 46] has been used in various contexts such as nearest neighbour search [14, 16, 32, 42, 46, 64], Web document clustering [22, 23] and query plan caching [10]. In this paper, we use LSH in the physical design of RDF databases. While multiple families of LSH functions have been developed [23, 27, 29, 36, 46], these functions assume that the input distribution is either uniform or static. In contrast, AdaptiveLSH can continuously adapt to changes in the input distribution to achieve higher accuracy, which translates to adapting to changes in the query access patterns in the workloads in the context of RDF databases.

Physical design has been a topic of ongoing discussion in the world of RDF and SPARQL [4, 12, 63, 66]. One option is to represent data in a single large table [25] and build clustered indexes, where each index implements a different sort order [40, 58, 65]. It has also been argued that grouping data can help improve performance [4, 63]. For this reason, multiple physical representations have been developed: in the *group-by-predicate* representation, the database is vertically partitioned and the tables are stored in a column-store [4]; in the *group-by-entity* representation, implicit relationships within the database are discovered (either manually [66] or automatically [21]), and the RDF data are mapped to a relational database; and in the *group-by-vertex* representation, RDF's inherent graph-structure is preserved, whereby data can be grouped by the vertices in the graph [68]. These workload-oblivious representations have issues for different types of queries, due to reasons such as fragmented data, unnecessarily large intermediate result tuples generated during query evaluation and/or suboptimal index pruning [12].

To address some of these issues, workload-aware techniques have been proposed [37, 41]. For example, view materialization techniques have been implemented for RDF over relational engines [37]. However, these materialized views are difficult to adapt to changing workloads for reasons discussed in Section 1. Workload-aware distribution techniques have also been developed for RDF [41] and implemented in systems such as WARP [41] and Partout [35], but these systems are not runtime-adaptive. With AdaptiveLSH, we aim to address the problem adaptively, by clustering fragmented records in the database based on the workload.

While there are self-tuning SQL databases [28, 44, 45] and techniques for automatic schema design in SQL [7, 17, 53, 67], these techniques are not directly applicable to RDF. In RDF, the advised changes to the underlying physical schema can be drastic, for example, requiring the system to switch from a row-oriented representation to a columnar one, all at runtime, which are hard to achieve using existing techniques. Consequently, there have been efforts in designing workload-adaptive and self-tuning RDF data management systems [9, 12, 13, 59, 60]. In H2RDF [59], the choice between centralized versus distributed execution is made adaptively. In PHDStore [9], data are adaptively replicated and distributed across the compute nodes; however, the underlying physical layout is fixed within each node. A mechanism for adaptively caching partial results is introduced in [60]. With

$$
\begin{array}{c}
\\ r_0 \\ r_1 \\ r_2 \\ r_3 \\ r_4 \\ r_5 \\ r_6 \\ r_7
\end{array}
\begin{array}{c}
\begin{array}{cccccccc} q_0 & q_1 & q_2 & q_3 & q_4 & q_5 & q_6 & q_7 \end{array} \\
\left(\begin{array}{cccc:cccc}
0 & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} \\
\mathbf{1} & 0 & \mathbf{1} & 0 & 0 & 0 & 0 & 0 \\ \hdashline
0 & 0 & 0 & \mathbf{1} & 0 & \mathbf{1} & 0 & \mathbf{1} \\
\mathbf{1} & 0 & \mathbf{1} & 0 & \mathbf{1} & 0 & \mathbf{1} & 0 \\ \hdashline
\mathbf{1} & 0 & \mathbf{1} & 0 & \mathbf{1} & 0 & \mathbf{1} & 0 \\
0 & \mathbf{1} & 0 & \mathbf{1} & 0 & \mathbf{1} & 0 & \mathbf{1} \\ \hdashline
\mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{array}\right)
\end{array}
$$

(a) Representation at $t = 8$.

$$
\begin{array}{c}
\\ r_7 \\ r_1 \\ r_4 \\ r_3 \\ r_6 \\ r_0 \\ r_5 \\ r_2
\end{array}
\begin{array}{c}
\begin{array}{cccccccc} q_0 & q_1 & q_2 & q_3 & q_4 & q_5 & q_6 & q_7 \end{array} \\
\left(\begin{array}{cccc:cccc}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\mathbf{1} & 0 & \mathbf{1} & 0 & 0 & 0 & 0 & 0 \\ \hdashline
\mathbf{1} & 0 & \mathbf{1} & 0 & \mathbf{1} & 0 & \mathbf{1} & 0 \\
\mathbf{1} & 0 & \mathbf{1} & 0 & \mathbf{1} & 0 & \mathbf{1} & 0 \\ \hdashline
\mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} \\
0 & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} \\ \hdashline
0 & \mathbf{1} & 0 & \mathbf{1} & 0 & \mathbf{1} & 0 & \mathbf{1} \\
0 & 0 & 0 & \mathbf{1} & 0 & \mathbf{1} & 0 & \mathbf{1}
\end{array}\right)
\end{array}
$$

(b) Clustered on rows

$$
\begin{array}{c}
\\ r_7 \\ r_1 \\ r_4 \\ r_3 \\ r_6 \\ r_0 \\ r_5 \\ r_2
\end{array}
\begin{array}{c}
\begin{array}{cccccccc} q_0 & q_2 & q_6 & q_4 & q_3 & q_5 & q_7 & q_1 \end{array} \\
\left(\begin{array}{cccc:cccc}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\mathbf{1} & \mathbf{1} & 0 & 0 & 0 & 0 & 0 & 0 \\ \hdashline
\mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} & 0 & 0 & 0 & 0 \\
\mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} & 0 & 0 & 0 & 0 \\ \hdashline
\mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} \\
0 & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} \\ \hdashline
0 & 0 & 0 & 0 & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} \\
0 & 0 & 0 & 0 & \mathbf{1} & \mathbf{1} & \mathbf{1} & 0
\end{array}\right)
\end{array}
$$

(c) Clustered on rows and columns

$$
\begin{array}{c}
\\ c_7 \\ c_1 \\ c_4 \\ c_3 \\ c_6 \\ c_0 \\ c_5 \\ c_2
\end{array}
\begin{array}{c}
\begin{array}{cc} G_0 & G_1 \end{array} \\
\left(\begin{array}{cc}
0 & 0 \\
2 & 0 \\ \hdashline
2 & 2 \\
2 & 2 \\ \hdashline
4 & 4 \\
3 & 4 \\ \hdashline
2 & 2 \\
1 & 2
\end{array}\right)
\end{array}
$$

(d) Grouping of bits

$$
\begin{array}{c}
\\ c_7 \\ c_1 \\ c_4 \\ c_3 \\ c_6 \\ c_0 \\ c_5 \\ c_2
\end{array}
\begin{array}{c}
\begin{array}{cc} G_0 & G_1 \end{array} \\
\left(\begin{array}{cc}
0 & 0 \\
2 & 0 \\ \hdashline
4 & 0 \\
4 & 0 \\ \hdashline
4 & 4 \\
3 & 4 \\ \hdashline
0 & 4 \\
0 & 3
\end{array}\right)
\end{array}
$$

(e) Alternative grouping

Figure 2: Matrix representation of query access patterns.

AdaptiveLSH, we are addressing the adaptive record layout problem, therefore, we believe that AdaptiveLSH will complement existing techniques and facilitate the development of runtime adaptive RDF systems.

3. PRELIMINARIES

Given a sequence of database records that represent the records' serialization order in the storage system, the access patterns of a query can conceptually be represented as a bit vector, where a bit is set to 1 if the corresponding record in the sequence is accessed by that query. We call this bit vector a *query access vector* ($\vec{q}$).

Depending on the system, a record may denote a single RDF triple (i.e., the atomic unit of information in RDF) as in systems like RDF-3x [57], or a collection of RDF triples. Our conceptual model is applicable either way.

As more queries are executed, their query access vectors can be accumulated column-by-column in a matrix, as shown in Fig. 2a. We call this matrix a *query access matrix*. For presentation, let us assume that queries are numbered according to their order of execution by the RDF data management system.

Each row of the query access matrix constitutes what we call a *record utilization vector* ($\vec{r}$), which represents the set of queries that access record r. As a convention, to distinguish between a query and its access vector (likewise, a record and its utilization vector), we use the symbols q and $\vec{q}$ (likewise, r and $\vec{r}$), respectively. The complete list of symbols are given in Table 1.

To model the memory hierarchy, we use an additional notation in the matrix representation: records that are physically stored together on the same disk/memory page should be grouped together in the query access matrix. For example, Fig. 2a and Fig. 2b represent two alternative ways

	Symbol	*Description*
Constants	ω	database size (i.e., number of records)
	ϵ	number of pages in the storage system
	k	maximum no. of query access vectors that can be stored
	b	number of entries in each record utilization counter
	t	current time
Data structures	$\vec{q}$	query access vector (contains ω bits)
	$\vec{r}$	record utilization vector (contains k bits)
	$\vec{c}$	record utilization counter (contains b entries)
	$\vec{P}$	depending on the context, a point in a k-dimensional or b-dimensional (Taxicab) space
	$M_{\omega \times k}$	query access matrix; contains the last k most representative query access vectors (in columns), or equivalently, ω record utilization vectors (in rows)
	$C_{\omega \times b}$	frequency matrix; represents record utilization frequency over b groups of query access vectors
Accessors	$q[i]$	value of the i^{th} bit in query access vector $\vec{q}$
	$r[i]$	value of the i^{th} bit in record utilization vector $\vec{r}$
	$c[i]$	value of the i^{th} entry in record utilization counter $\vec{c}$
	$P[i]$	value of the i^{th} coordinate in point $\vec{P}$
	$M[i][j]$	value of the i^{th} row and j^{th} column in matrix
	$C[i][j]$	value of the i^{th} row and j^{th} column in matrix
Distances	$\delta(\vec{r_x}, \vec{r_y})$	Hamming distance between two record utilization vectors
	$\delta^H(\vec{q_x}, \vec{q_y})$	MIN-HASH distance between two query access vectors
	$\delta^M(\vec{P_x}, \vec{P_y})$	Manhattan distance between two points

Table 1: Symbols used throughout the manuscript

in which the records in an RDF database can be clustered (groups are separated by horizontal dashed lines). Even though both figures depict essentially the same query access patterns, the physical organization in Fig. 2b is superior, because in Fig. 2a, most queries require access to 4 pages each, whereas in Fig. 2b, the number of accesses is reduced by almost half.

Given a sequence of queries and the number of pages in the storage system, our objective is to *store records with similar utilization vectors together so as to minimize the total number of page accesses*. To determine the similarity between record utilization vectors, we exploit the following property: two records are co-accessed by a query if both of the corresponding bits in that query's access vector are set to 1. Extending this concept to a set of queries, we say that two records are co-accessed across multiple queries if the corresponding bits in the record utilization vectors are set to 1 for all the queries in the set. For example, according to Fig. 2a, records r_1 and r_3 are co-accessed by queries q_0 and q_2, and records r_0 and r_6 are co-accessed across the queries q_1–q_7.

Given a sequence of queries, it is possible that a pair of records are not co-accessed in *all* of the queries. Therefore, to measure the extent to which a pair of records are co-accessed, we rely on their Hamming distance [38]. Specifically, given two record utilization vectors for the same sequence of queries, their Hamming distance—denoted as $\delta(\vec{q_x}, \vec{q_y})$—is defined as the minimum number of substitutions necessary to make the two bit vectors the same [38].[1] Hence, the smaller the Hamming distance between a pair of records, the greater the extent to which they are co-accessed.

Consider the record utilization vectors $\vec{r_0}$, $\vec{r_2}$, $\vec{r_5}$ and $\vec{r_6}$ across the query sequence q_0–q_7 in Fig. 2a. The pairwise Hamming distances are as follows: $\delta(r_0, r_6) = 1$, $\delta(r_2, r_5) = 1$, $\delta(r_0, r_5) = 3$, $\delta(r_0, r_2) = 4$, $\delta(r_5, r_6) = 4$ and $\delta(r_2, r_6) =$

[1] The Hamming distance between two record utilization vectors is equal to their edit distance [52], as well as the Manhattan distance [50] between these two vectors in l_1 norm.

5. Consequently, to achieve better physical clustering, we should try to store r_0 and r_6 together and r_2 and r_5 together, while keeping r_0 and r_6 apart from r_2 and r_5.

4. OVERVIEW OF ADAPTIVE-LSH

The dynamic nature of queries over RDF data necessitate a solution different from existing clustering algorithms [12]. That is, while conventional clustering algorithms [48] might be perfectly applicable for the *offline* tuning of a database, in an *online* scenario, what is needed is an algorithm that clusters records on-the-fly and within microseconds. Clustering is an NP-complete problem [48], and most approximations take at least quadratic time. It is not very well-understood which clustering algorithm is more suitable for which types of input distributions [5], let alone the fact that incremental versions of these algorithms are largely domain-specific [6]. In contrast AdaptiveLSH is a self-tuning locality-sensitive hash (LSH) function, which is used as follows:

As records are fetched from the storage system, we keep track of records that are accessed by the same query but are fragmented across the pages in the storage system. Then, we use AdaptiveLSH to decide, in constant-time, how a fragmented record needs to be clustered in the storage system (cf., Fig. 1). Furthermore, we develop methods to continuously auto-tune this LSH function to adapt to changing query access patterns that we encounter while executing a workload. This way, AdaptiveLSH can achieve much higher clustering accuracy than conventional LSH schemes, which are static.

Let $\mathbb{Z}_{\alpha \cdots \beta}$ denote the set of integers in the interval $[\alpha, \beta]$, and let $\mathbb{Z}^n_{\alpha \cdots \beta}$ denote the n-fold Cartesian product:

$$\underbrace{\mathbb{Z}_{\alpha \cdots \beta} \times \cdots \times \mathbb{Z}_{\alpha \cdots \beta}}_{n}.$$

Furthermore, let us assume that we are given a non-injective, surjective function $f : \mathbb{Z}_{0 \cdots (k-1)} \to \mathbb{Z}_{0 \cdots (b-1)}$, where $b \ll k$, and for all $y \in \mathbb{Z}_{0 \cdots (b-1)}$, it holds that

$$\Big|\{x : f(x) = y\}\Big| \leq \Big\lceil \frac{k}{b} \Big\rceil.$$

In other words, f is a hash function with the property that, given k input values and b possible outcomes, no more than $\lceil \frac{k}{b} \rceil$ values in the domain of the function will be hashed to the same value (Section 5.2 discusses in more detail how f can be constructed).[2] Then, we define AdaptiveLSH as $h : \mathbb{Z}^k_{0 \cdots 1} \to \mathbb{Z}_{0 \cdots (\epsilon-1)}$, where ϵ represents the number of pages in the storage system. More specifically, h is defined as a composition of two functions h_1 and h_2.

Definition 1 (AdaptiveLSH)**.**

Let

$$\vec{r} = (r[0], \ldots, r[k-1]) \in \mathbb{Z}^k_{0 \cdots 1}, \textit{ and}$$
$$\vec{c} = (c[0], \ldots, c[b-1]) \in \mathbb{Z}^b_{0 \cdots \lceil \frac{k}{b} \rceil}.$$

Then, an adaptive LSH function h is defined as

$$h = \mathbf{h_2} \circ \mathbf{h_1}$$

where

$$\mathbf{h_1} : \mathbb{Z}^k_{0 \cdots 1} \to \mathbb{Z}^b_{0 \cdots \lceil \frac{k}{b} \rceil}, \textit{ where } h_1(\vec{r}) = \vec{c} \textit{ iff}$$

$$\forall y \; c[y] = \sum_{x=0}^{k-1} \begin{cases} r[x] & : f(x) = y \\ 0 & : f(x) \neq y \end{cases}$$

$$\mathbf{h_2} : \mathbb{Z}^b_{0 \cdots \lceil \frac{k}{b} \rceil} \to \mathbb{Z}_{0 \cdots (\epsilon-1)}, \textit{ where } h_2(\vec{c}) = v \textit{ iff}$$

$$v = \begin{cases} \textit{coordinate of } \vec{c} \textit{ (rounded to the} \\ \textit{nearest integer) on a space-filling} \\ \textit{curve [55] of length } \epsilon \textit{ that covers} \\ \mathbb{Z}^b_{0 \cdots \lceil \frac{k}{b} \rceil} \end{cases}$$

According to Def. 1, h is constructed as follows:

1. Using a hash function f (which can be treated as a black box for the moment), a record utilization vector $\vec{r}$ with k bits is divided into b disjoint segments $\vec{r}_0, \ldots, \vec{r}_{b-1}$ such that $\vec{r}_0, \ldots, \vec{r}_{b-1}$ contain all the bits in $\vec{r}$, and each $\vec{r}_i \in \{\vec{r}_0, \ldots, \vec{r}_{b-1}\}$ has at most $\lceil \frac{k}{b} \rceil$ bits. Then, a record utilization counter $\vec{c}$ with b entries is computed such that the i^{th} entry of $\vec{c}$ (i.e., $c[i]$) contains the number of 1-bits in $\vec{r}_i$. Without loss of generality, a record utilization counter $\vec{c}$ can be represented as a b-dimensional point in the coordinate system $\mathbb{Z}^b_{0 \cdots \lceil \frac{k}{b} \rceil}$.

2. The final hash value is computed by computing the z-value [33,55] of the points in $\mathbb{Z}^b_{0 \cdots \lceil \frac{k}{b} \rceil}$, and dropping off the last m bits from the produced z-values, where $m = k - \lceil \log_2 \epsilon \rceil$.

In Section 5.1, we show that AdaptiveLSH that maps k-dimensional record utilization vectors to natural numbers in the interval $[0, \ldots, \epsilon - 1]$ is locality-sensitive, with two important implications: (i) records with similar record utilization vectors (i.e., small Hamming distances) are likely to be hashed to the same value, while (ii) records with dissimilar record utilization vectors are likely to be separated. Therefore, the problem of clustering records in the storage system can be approximated using AdaptiveLSH, such that clustering n records takes $\mathrm{O}(n)$ time.

The quality of AdaptiveLSH, that is, how well it approximates the original Hamming distances, depends on two factors: (i) the characteristics of the workload so far, which is reflected by the bit distribution in the record utilization vectors, and (ii) the choice of f. In Section 5.2, we demonstrate that f can be tuned to adapt to the changing patterns in record utilization vectors to maintain the approximation quality of AdaptiveLSH at a steady and high level.

Algorithms 1–3 present our approach for computing the outcome of AdaptiveLSH and for incrementally tuning the LSH function every time a query is executed. Note that we have two design considerations: (i) tuning should take constant-time, otherwise, there is no point in using a function, (ii) the memory footprint should be low because it would be desirable to maximize the allocation of memory

[2]This uniformity condition simplifies the sensitivity analysis of AdaptiveLSH, but it is not a requirement from an algorithmic point of view. Relaxing this condition is left as future work.

Algorithm 1 Initialize

Ensure:
Record utilization counters are allocated and initialized
procedure INITIALIZE()
construct int $C[\omega][2b]$ ▷ For simplicity, C is allocated statically; however, in practice, it can be allocated dynamically to reduce memory footprint.
for all $i \in (0, \ldots, \omega - 1)$ **do**
for all $j \in (0, \ldots, 2b - 1)$ **do**
$C[i][j] \leftarrow 0$
end for
end for
end procedure

Algorithm 2 Tune

Require:
$\vec{q_t}$: query access vector produced at time t
Ensure:
Underlying data structures are updated and f is tuned such that the LSH function maintains a steady approximation quality
1: **procedure** TUNE($\vec{q_t}$)
2: RECONFIGURE-F($\vec{q_t}$)
3: **for all** $i \in$ POSITIONAL($\vec{q_t}$) **do**
4: loc $\leftarrow f(t)$
5: **if** loc $<$ (shift % b) **then**
6: loc += b
7: **end if**
8: $C[i]$[loc]++ ▷ Increment record utilization counters based on the new query access pattern
9: **if** $t\%\lceil \frac{k}{b} \rceil = 0$ **then** ▷ Reset "old" counters
10: shift++
11: $C[i]$[(shift+b)%$2b$] $\leftarrow 0$
12: **end if**
13: **end for**
14: **end procedure**

to core database functionality. Consequently, instead of relying on record utilization vectors, the algorithm computes and incrementally maintains record utilization counters (cf., Algorithm 1) that are much easier to maintain and that have a much smaller memory footprint due to the fact that $b \ll k$. Then, whenever there is a need to compute the outcome of the LSH function for a given record, the HASH procedure is called with the *id* of the record, which in turn relies on h_2 to compute the hash value (cf., Algorithm 3).

The TUNE procedure looks at the next query access vector, and updates f (line 2), which will be discussed in more detail in Section 5.2. Then it computes positions of records that have been accessed by that query (line 3), and increments the corresponding entries in the utilization counters of those records that have been accessed (line 8). To determine which entry to increment, the algorithm relies on h_1, hence, $f(t)$ (cf., Def. 1) and a shifting scheme. In line 11, old entries in record utilization counters are reset based on an approach that we discuss in Section 5.3. In that section we also discuss the shifting scheme.

5. DETAILS OF ADAPTIVE-LSH AND OPTIMIZATIONS

This section is structured as follows: Section 5.1 shows that ADAPTIVELSH has the properties of a locality-sensitive hashing scheme. Section 5.2 describes our approach for tuning f based on the most recent query access patterns, and Section 5.3 explains how old bits are removed from record

Algorithm 3 Hash

Require:
id: id of record whose hash is being computed
Ensure:
Hash value is returned
procedure HASH(id)
return Z-VALUE(C[id]) ▷ Apply h_2
end procedure

utilization counters.

5.1 Properties of Adaptive-LSH

This section discusses the locality-sensitive properties of $h = h_2 \circ h_1$ and demonstrates that h can be used for clustering the records. First, the relationship between record utilization vectors and the record utilization counters that are obtained by applying h_1 is shown.

Theorem 1 (Distance Bounds)**.** *Given a pair of record utilization vectors $\vec{r}_1$ and $\vec{r}_2$ with size k, let $\vec{c}_1$ and $\vec{c}_2$ denote two record utilization counters with size b such that $\vec{c}_1 = h_1(\vec{r}_1)$ and $\vec{c}_2 = h_1(\vec{r}_2)$ (cf., Definition 1). Furthermore, let $c_1[i]$ and $c_2[i]$ denote the i^{th} entry in $\vec{c}_1$ and $\vec{c}_2$, respectively. Then,*

$$\delta(\vec{r_1}, \vec{r_2}) \geq \sum_{i=0}^{b-1} \left| c_1[i] - c_2[i] \right| \tag{1}$$

where $\delta(\vec{r_1}, \vec{r_2})$ represents the Hamming distance between $\vec{r}_1$ and $\vec{r}_2$.

Proof 1. *It is possible to prove Theorem 1 by induction on b.*
Base case*: Theorem 1 holds when $b = 1$. According to Definition 1, when $b = 1$, $c_1[0]$ and $c_2[0]$ correspond to the total number of 1-bits in $\vec{r}_1$ and $\vec{r}_2$, respectively. Note that the Hamming distance between $\vec{r}_1$ and $\vec{r}_2$ will be smallest if and only if these two record utilization vectors are aligned on as many 1-bits as possible. In that case, they will differ in only $\left|c_1[0] - c_2[0]\right|$ bits, which corresponds to their Hamming distance. Consequently, Equation 1 holds for $b = 1$.*
Inductive step*: It needs to be shown that if Equation 1 holds for $b \leq \alpha$, where α is a natural number greater than or equal to 1, then it must also hold for $b = \alpha + 1$.*

Let $\Pi_f(\vec{r}, g)$ denote a record utilization vector $r' = (r'[0], \ldots, r'[k-1])$ such that for all $i \in \{0, \ldots, k-1\}$, $r'[i] = r[i]$ holds if $f(i) = g$, and $r'[i] = 0$ otherwise. Then,

$$\delta(\vec{r}_1, \vec{r}_2) = \sum_{g=0}^{b-1} \delta(\Pi_f(\vec{r}_1, g), \Pi_f(\vec{r}_2, g)). \tag{2}$$

That is, the Hamming distance between any two record utilization vectors is the summation of their individual Hamming distances within each group of bits that share the same hash value with respect to f. This property holds because f is a (total) function, and Π_f masks all the irrelevant bits. As an abbreviation, let $\delta_g = \delta(\Pi_f(\vec{r}_1, g), \Pi_f(\vec{r}_2, g))$. Then, due to the same reasoning as in the base case, for $g = \alpha$, the following equation holds:

$$\delta_\alpha(\vec{r_1}, \vec{r_2}) \geq \left| c_1[\alpha] - c_2[\alpha] \right| \tag{3}$$

Consequently, due to the additive property in Equation 2, Equation 1 holds also for $b = \alpha + 1$. Thus, by induction, Theorem 1 holds. □

Theorem 1 suggests that the Hamming distance between any two record utilization vectors $\vec{r_1}$ and $\vec{r_2}$ can be approximated using record utilization counters $\vec{c_1} = h_1(\vec{r_1})$ and $\vec{c_2} = h_1(\vec{r_2})$ because Equation 1 provides a lower bound on $\delta(\vec{r_1}, \vec{r_2})$. In fact, the right-hand side of Equation 1 is equal to the Manhattan distance [50] between $\vec{c_1}$ and $\vec{c_2}$ in $\mathbb{Z}^b_{0\cdots\lceil\frac{k}{b}\rceil}$, and since $\delta(\vec{r_1}, \vec{r_2})$ is equal to the Manhattan distance between $\vec{r_1}$ and $\vec{r_2}$ in $\mathbb{Z}^k_{0\cdots 1}$, it is easy to see that h_1 is a transformation that approximates Manhattan distances. The following corollary captures this property.

Corollary 2 (Distance Approximation). *Given a pair of record utilization vectors $\vec{r_1}$ and $\vec{r_2}$ with size k, let $\vec{c_1}$ and $\vec{c_2}$ denote two points in the coordinate system $\mathbb{Z}^b_{0\cdots\lceil\frac{k}{b}\rceil}$ such that $\vec{c_1} = h_1(\vec{r_1})$ and $\vec{c_2} = h_1(\vec{r_2})$ (cf., Definition 1). Let $\delta^M(\vec{r_1}, \vec{r_2})$ denote the Manhattan distance between $\vec{r_1}$ and $\vec{r_2}$, and let $\delta^M(\vec{c_1}, \vec{c_2})$ denote the Manhattan distance between $\vec{c_1}$ and $\vec{c_2}$. Then, the following holds:*

$$\delta(\vec{r_1}, \vec{r_2}) = \delta^M(\vec{r_1}, \vec{r_2}) \geq \delta^M(\vec{c_1}, \vec{c_2}) \tag{4}$$

Proof 2. *Hamming distance in $\mathbb{Z}^k_{0\cdots 1}$ is a special case of Manhattan distance. Furthermore, by definition [50], the right hand side of Equation 1 equals the Manhattan distance $\delta^M(\vec{c_1}, \vec{c_2})$; therefore, Equation 4 holds.* □

Next, it is demonstrated that $h = h_2 \circ h_1$ is a locality-sensitive transformation [36, 46]. In particular, the definition of locality-sensitiveness by Tao et al. [64] is used, and it is shown that the probability that two record utilization vectors $\vec{r_1}$ and $\vec{r_2}$ are hashed to the same page increases as the (Manhattan) distance between r_1 and r_2 decreases.

Theorem 3 (Collision Probabilities). *Given a pair of record utilization vectors $\vec{r_1}$ and $\vec{r_2}$ with size k, let $\delta^M(\vec{r_1}, \vec{r_2})$ denote the Manhattan distance between $\vec{r_1}$ and $\vec{r_2}$. Furthermore, let m denote the number of rightmost bits that are dropped by h_2. When $b = 1$ (i.e., the size of the record utilization counters produced by h_1), the probability that* the pair of record utilization vectors will be hashed to the same value by $h_2 \circ h_1$ provided that their initial Manhattan distance is *x is given by the following formula:*

$$\mathrm{P}\scriptstyle{\mathrm{R}}\displaystyle\Big(h_2 \circ h_1(\vec{r_1}) = h_2 \circ h_1(\vec{r_2}) \;\Big|\; \delta^M(\vec{r_1}, \vec{r_2}) = x\Big) = \frac{\sum_{a=0}^{x}\sum_{\Delta=0}^{k-x}\binom{x}{a}\binom{k}{k-x}\binom{k-x}{\Delta}\rho(\Delta+a, x-2a, m)}{2^{2k}} \tag{5}$$

where $\rho(x, y, m) : (\mathbb{Z}_{0\cdots\infty}, \mathbb{Z}_{-\infty\cdots\infty}, \mathbb{Z}_{0\cdots\infty}) \rightarrow \{0, 1\}$ is a function such that

$$\rho(x, y, m) = \begin{cases} 1 & \textit{if } 0 \leq (x \bmod 2^m) + y < 2^m \\ 0 & \textit{else} \end{cases}.$$

Proof 3. *If the Hamming/Manhattan distance between $\vec{r_1}$ and $\vec{r_2}$ is x, then it means that these two vectors will differ in exactly x bits, as shown below.*

$$\begin{aligned}\mathbf{r_1}: &\ \square\square\square\,\overbrace{\mathbf{111\ldots1}}^{a}\,\mathbf{0\cdots000}\,\square\square\square\\ \mathbf{r_2}: &\ \square\square\square\,\mathbf{000\cdots0}\,\underbrace{\mathbf{1\ldots111}}_{x-a}\,\square\square\square\end{aligned}$$

Furthermore, if $\vec{r_1}$ has $\Delta + a$ bits set to 1, then $\vec{r_2}$ must have $\Delta + (x - a)$ bits set to 1, where Δ denotes the number of matching 1-bits between $\vec{r_1}$ and $\vec{r_2}$. Note that when $b = 1$, $\vec{c_1} = h_1(\vec{r_1}) = (\Delta + a)$ and $\vec{c_2} = h_1(\vec{r_2}) = (\Delta + x - a)$.

It is easy to see that $a \in \mathbb{Z}_{0\cdots x}$ and $\Delta \in \mathbb{Z}_{0\cdots k-x}$. For each value of a, the non-matching bits in $\vec{r_1}$ and $\vec{r_2}$ can be combined in $\binom{x}{a}$ possible ways, and these non-matching bits can be positioned across the k bits in $\binom{k}{k-x}$ possible ways. Likewise, for each value of Δ, those matching 1-bits that are counted by Δ can be combined in $\binom{k-x}{\Delta}$ possible ways, hence, the first three components of the multiplication in the numerator of Equation 5.

Among the aforementioned combinations, $h_2(\vec{c_1}) = h_2(\vec{c_2})$ will be true if and only if the binary representations of $\vec{c_1}$ and $\vec{c_2}$ share the same sequence of bits except for their last m bits. This condition will be satisfied if and only if $\Delta + a$ and $\Delta + x - a$ have the same quotient when divided by 2^m. In other words, $(\Delta + a) \bmod 2^m + (\Delta + x - a) - (\Delta + a)$ must be greater than or equal to 0 or less than 2^m, hence, the need to multiply by ρ in the numerator of Equation 5.

Since $\vec{r_1}$ and $\vec{r_2}$ consist of k bits, there can be $2^k \times 2^k = 2^{2k}$ possible combinations in total, which corresponds to the denominator of Equation 5. □

Next, Theorem 3 is extended to cases in which $b \geq 2$. However, first, some auxiliary statements need to be made.

Lemma 4. *Let $\rho(x, y, m) : (\mathbb{Z}_{0\cdots\infty}, \mathbb{Z}_{-\infty\cdots\infty}, \mathbb{Z}_{0\cdots\infty}) \rightarrow \{0, 1\}$ denote a function such that*

$$\rho(x, y, m) = \begin{cases} 1 & \textit{if } 0 \leq (x \bmod 2^m) + y < 2^m \\ 0 & \textit{else} \end{cases}.$$

Then, for any $m \in \mathbb{Z}_{0\cdots\infty}$, the following properties hold:

Property A. *For any $x \in \mathbb{Z}_{0\cdots\infty}$, if $0 \leq y_1 \leq y_2$, then $\rho(x, y_1, m) \geq \rho(x, y_2, m)$;*

Property B. *For any $x \in \mathbb{Z}_{0\cdots\infty}$, if $y_2 \leq y_1 \leq 0$, then $\rho(x, y_1, m) \geq \rho(x, y_2, m)$;*

Property C. *For any x_1, x_2, y_1, y_2 such that $x_1, x_2 \in \mathbb{Z}_{0\cdots\infty}$ and $x_1 + y_1 = x_2 + y_2$, if $0 \leq y_1 \leq y_2$, then $\rho(x_1, y_1, m) \geq \rho(x_2, y_2, m)$; and*

Property D. *For any x_1, x_2, y_1, y_2 such that $x_1, x_2 \in \mathbb{Z}_{0\cdots\infty}$ and $x_1 + y_1 = x_2 + y_2$, if $y_2 \leq y_1 \leq 0$, then $\rho(x_1, y_1, m) \geq \rho(x_2, y_2, m)$; and*

Proof 4. *All of the four properties are proven by contradiction.*

Property A can be proven as follows:

A1. *For any $x \in \mathbb{Z}_{0\cdots\infty}$ and $m \in \mathbb{Z}_{0\cdots\infty}$,*

$$0 \leq x \bmod 2^m < 2^m \tag{6}$$

by the definition of the modulo operation.

A2. *Assume $0 \leq y_1 \leq y_2$.*

- *Assume $\rho(x, y_1, m) < \rho(x, y_2, m)$.*
- *Therefore, since $0 \leq y_1 \leq y_2$,*

$$0 \leq x \bmod 2^m + y_1 \leq x \bmod 2^m + y_2. \tag{7}$$

- *Since $\rho(x, y_1, m) < \rho(x, y_2, m)$, $\rho(x, y_1, m) = 0$ and $\rho(x, y_2, m) = 1$. (Note that the codomain of ρ is the set $\{0, 1\}$.)*

- *If $\rho(x, y_1, m) = 0$, according to the definition of ρ (cf., Lemma 4) and Equation 6, and based on the fact that $y_1 \geq 0$, the following statement must be true:*

$$x \ mod \ 2^m + y_1 \geq 2^m. \tag{8}$$

- *Likewise, if $\rho(x, y_2, m) = 1$, according to the definition of ρ (cf., Lemma 4), the following statement must be true:*

$$0 \leq x \ mod \ 2^m + y_2 < 2^m. \tag{9}$$

- *Therefore, according to Equations 8 and 9,*

$$x \ mod \ 2^m + y_2 < x \ mod \ 2^m + y_1. \tag{10}$$

- *However, Equation 10 contradicts Equation 7, therefore,*

$$\rho(x, y_1, m) \geq \rho(x, y_2, m). \tag{11}$$

A3. *Since Equation 11 holds under the assumption that $0 \leq y_1 \leq y_2$, Property A in Lemma 4 holds.*

Property B can be proven as follows:

B1. *For any $x \in \mathbb{Z}_{0 \cdots \infty}$ and $m \in \mathbb{Z}_{0 \cdots \infty}$,*

$$0 \leq x \ mod \ 2^m < 2^m \tag{12}$$

by the definition of the modulo operation.

B2. *Assume $y_2 \leq y_1 \leq 0$.*

- *Assume $\rho(x, y_1, m) < \rho(x, y_2, m)$.*
- *Therefore, since $y_2 \leq y_1 \leq 0$,*

$$x \ mod \ 2^m + y_2 \leq x \ mod \ 2^m + y_1 < 2^m. \tag{13}$$

- *Since $\rho(x, y_1, m) < \rho(x, y_2, m)$, $\rho(x, y_1, m) = 0$ and $\rho(x, y_2, m) = 1$. (Note that the codomain of ρ is the set $\{0, 1\}$.)*
- *If $\rho(x, y_1, m) = 0$, according to the definition of ρ (cf., Lemma 4) and Equation 12, and based on the fact that $y_1 \leq 0$, the following statement must be true:*

$$x \ mod \ 2^m + y_1 < 0. \tag{14}$$

- *Likewise, if $\rho(x, y_2, m) = 1$, according to the definition of ρ (cf., Lemma 4), the following statement must be true:*

$$0 \leq x \ mod \ 2^m + y_2 < 2^m. \tag{15}$$

- *Therefore, according to Equations 14 and 15,*

$$x \ mod \ 2^m + y_1 < x \ mod \ 2^m + y_2. \tag{16}$$

- *However, Equation 16 contradicts Equation 13, therefore,*

$$\rho(x, y_1, m) \geq \rho(x, y_2, m). \tag{17}$$

B3. *Since Equation 17 holds under the assumption that $y_2 \leq y_1 \leq 0$, Property B in Lemma 4 holds.*

Property C can be proven as follows:

C1. *For any $x_1, x_2 \in \mathbb{Z}_{0 \cdots \infty}$ and $m \in \mathbb{Z}_{0 \cdots \infty}$,*

$$\begin{aligned} 0 \leq x_1 \ mod \ 2^m < 2^m \\ 0 \leq x_2 \ mod \ 2^m < 2^m \end{aligned} \tag{18}$$

by the definition of the modulo operation.

C2. *Assume*

$$\begin{aligned} &0 \leq y_1 \leq y_2 \text{ and} \\ &x_1 + y_1 = x_2 + y_2. \end{aligned} \tag{19}$$

- *Assume $\rho(x_1, y_1, m) < \rho(x_2, y_2, m)$.*
- *Since $\rho(x_1, y_1, m) < \rho(x_2, y_2, m)$, $\rho(x_1, y_1, m) = 0$ and $\rho(x_2, y_2, m) = 1$. (Note that the codomain of ρ is the set $\{0, 1\}$.)*
- *If $\rho(x_1, y_1, m) = 0$, according to the definition of ρ (cf., Lemma 4) and Equation 18, and based on the fact that $y_1 \geq 0$, the following statement must be true:*

$$x_1 \ mod \ 2^m + y_1 \geq 2^m. \tag{20}$$

- *Likewise, if $\rho(x_2, y_2, m) = 1$, according to the definition of ρ (cf., Lemma 4), the following statement must be true:*

$$0 \leq x_2 \ mod \ 2^m + y_2 < 2^m. \tag{21}$$

- *Since $x_1 = x_2 + y_2 - y_1$ (cf., Equation 19), the following statements are true:*

$$\begin{aligned} x_1 \ mod \ 2^m &= (x_2 + y_2 - y_1) \ mod \ 2^m \\ &= \big(x_2 \ mod \ 2^m + \\ &\quad (y_2 - y_1) \ mod \ 2^m\big) \ mod \ 2^m. \end{aligned} \tag{22}$$

- *Note that*

$$\begin{aligned} &x_2 \ mod \ 2^m + (y_2 - y_1) \ mod \ 2^m \geq \\ &\quad (x_2 \ mod \ 2^m + (y_2 - y_1) \ mod \ 2^m) \ mod \ 2^m \end{aligned}$$

(i.e., the mod 2^m of any positive number is always less than or equal to the number itself).

- *Therefore, Equation 22 can be restated as*

$$x_1 \ mod \ 2^m \leq x_2 \ mod \ 2^m + (y_2 - y_1) \ mod \ 2^m.$$

- *Likewise, the following statement must be true:*

$$x_1 \ mod \ 2^m \leq x_2 \ mod \ 2^m + (y_2 - y_1) \tag{23}$$

because $(y_2 - y_1) \geq 0$, therefore, $y_2 - y_1$ is always greater than or equal to its modulo in 2^m.

- *Therefore,*

$$x_1 \ mod \ 2^m + y_1 \leq x_2 \ mod \ 2^m + y_2.$$

- *It is also known from Equation 21 that $x_2 mod 2^m < 2^m$, therefore $x_1 \ mod \ 2^m < 2^m$ must also be true. This, however, contradicts Equation 20, therefore,*

$$\rho(x_1, y_1, m) \geq \rho(x_2, y_2, m). \tag{24}$$

C3. *Since Equation 24 holds under the assumption that $0 \leq y_1 \leq y_2$ and $x_1 + x_2 = y_1 + y_2$, Property C in Lemma 4 holds.*

Property D can be proven as follows:

D1. For any $x_1, x_2 \in \mathbb{Z}_{0\cdots\infty}$ and $m \in \mathbb{Z}_{0\cdots\infty}$,

$$\begin{aligned} 0 \le x_1 \ \ mod\ 2^m < 2^m \\ 0 \le x_2 \ \ mod\ 2^m < 2^m \end{aligned} \tag{25}$$

by the definition of the modulo operation.

D2. Assume

$$\begin{aligned} y_2 \le y_1 \le 0 \text{ and} \\ x_1 + y_1 = x_2 + y_2. \end{aligned} \tag{26}$$

- *Assume $\rho(x_1, y_1, m) < \rho(x_2, y_2, m)$.*
- *Since $\rho(x_1, y_1, m) < \rho(x_2, y_2, m)$, $\rho(x_1, y_1, m) = 0$ and $\rho(x_2, y_2, m) = 1$. (Note that the codomain of ρ is the set $\{0, 1\}$.)*
- *If $\rho(x_1, y_1, m) = 0$, according to the definition of ρ (cf., Lemma 4) and Equation 25, and based on the fact that $y_1 \le 0$, the following statement must be true:*

$$x_1 \ \ mod\ 2^m + y_1 < 0. \tag{27}$$

- *Likewise, if $\rho(x_2, y_2, m) = 1$, according to the definition of ρ (cf., Lemma 4), the following statement must be true:*

$$0 \le x_2 \ \ mod\ 2^m + y_2 < 2^m. \tag{28}$$

- *Since $x_2 = x_1 + y_1 - y_2$ (cf., Equation 26), the following statements are true:*

$$\begin{aligned} x_2\, mod\, 2^m &= \Big(x_1 + y_1 - y_2\Big)\, mod\, 2^m \\ &= \Big(x_1\, mod\, 2^m \\ &\qquad + (y_1 - y_2)\, mod\, 2^m\Big)\, mod\, 2^m. \end{aligned} \tag{29}$$

- *Then, for the same reasons discussed in the proof of Property C,*

$$\begin{aligned} x_2\, mod\, 2^m &\le x_1\, mod\, 2^m + (y_1 - y_2)\, mod\, 2^m \\ x_2\, mod\, 2^m &\le x_1\, mod\, 2^m + y_1 - y_2 \\ x_2\, mod\, 2^m + y_2 &\le x_1\, mod\, 2^m + y_1. \end{aligned} \tag{30}$$

- *Since $x_1 mod 2^m + y_1 < 0$ (cf., Equation 27), according to Equation 30, $x_2\ mod\ 2^m + y_2 < 0$ must also hold, but this statement contradicts Equation 28. Therefore,*

$$\rho(x_1, y_1, m) \ge \rho(x_2, y_2, m). \tag{31}$$

D3. Since Equation 31 holds under the assumption that $y_2 \le y_1 \le 0$ and $x_1 + x_2 = y_1 + y_2$, Property D in Lemma 4 holds. □

Lemma 5. *Let $\delta^M(\vec{r}_i, \vec{r}_j)$ denote the Manhattan distance between any two record utilization vectors $\vec{r}_i$ and $\vec{r}_j$, and let m denote the number of rightmost bits that are dropped by h_2, where h_2 is utilized in h (cf., Definition 1). Furthermore, let $\text{PR}_{==}(\vec{r}_i, \vec{r}_j, x)$ denote the posterior probability that, for any two record utilization vectors $\vec{r}_i$ and $\vec{r}_j$, $h(\vec{r}_i) = h(\vec{r}_j)$ provided that $\delta^M(\vec{r}_i, \vec{r}_j) = x$. Given any four record utilization vectors $\vec{r}_1$, $\vec{r}_2$, $\vec{r}_3$ and $\vec{r}_4$ with size k such that k is even, $\delta^M(\vec{r}_1, \vec{r}_2) = x$ and $\delta^M(\vec{r}_3, \vec{r}_4) = k - x$, the following property holds for any $x \le \lfloor \frac{k}{2} \rfloor$ and $m \in \mathbb{Z}_{0\cdots k-1}$:*

$$\text{PR}_{==}(\vec{r}_1, \vec{r}_2, x) \ge \text{PR}_{==}(\vec{r}_3, \vec{r}_4, k - x) \tag{32}$$

where $h = h_2 \circ h_1$ (cf., Definition 1), and $b = 1$ denotes the number of entries in the record utilization counters produced by h_1.

Proof 5. *The proof of Lemma 5 proceeds in multiple steps.*

S1. It is shown that for any two natural numbers a and Δ such that $a \le \frac{x}{2}$, $\Delta \le (k - x)$ and $\Delta - a \le \frac{k-2x}{2}$, the following property holds:

$$\begin{aligned} &\binom{x}{a}\binom{k-x}{\Delta}\rho(a+\Delta,\ x-2a) \ge \\ &\qquad \binom{k-x}{\Delta}\binom{x}{a}\rho(a+\Delta,\ k-x-2\Delta). \end{aligned} \tag{33}$$

This statement can be proven as follows:

- *For any two natural numbers a and Δ such that $a \le \frac{x}{2}$ and $\Delta \le (k - x)$, the binomials are defined and their products are equal (and positive) on both sides of the inequality.*
- *The condition $a \le \frac{x}{2}$ in S1 implies that $0 \le (x - 2a)$.*
- *The condition $\Delta - a \le \frac{k-2x}{2}$ in S1 implies that*

$$\begin{aligned} \Delta - a &\le \frac{k - 2x}{2} \\ 2(\Delta - a) &\le k - 2x \\ 2\Delta - 2a &\le k - 2x \\ x - 2a &\le k - x - 2\Delta. \end{aligned}$$

- *According to Property A in Lemma 4, $\rho(a+\Delta,\ x-2a) \ge \rho(a + \Delta,\ k - x - 2\Delta)$. Therefore, statement S1 holds.*

S2. It is shown that for any two natural numbers a and Δ such that $\frac{x}{2} \le a \le x$, $\Delta \le (k - x)$ and $\Delta - a \ge \frac{k-2x}{2}$, the following property holds:

$$\begin{aligned} &\binom{x}{a}\binom{k-x}{\Delta}\rho(a+\Delta,\ x-2a) \ge \\ &\qquad \binom{k-x}{\Delta}\binom{x}{a}\rho(a+\Delta,\ k-x-2\Delta). \end{aligned} \tag{34}$$

This statement can be proven as follows:

- *For any two natural numbers a and Δ such that $\frac{x}{2} \le a \le x$ and $\Delta \le (k - x)$, the binomials are defined and their products are equal (and positive) on both sides of the inequality.*
- *The condition $a \ge \frac{x}{2}$ in S2 implies that $(x - 2a) \le 0$.*
- *The condition $\Delta - a \ge \frac{k-2x}{2}$ in S2 implies that*

$$\begin{aligned} \Delta - a &\ge \frac{k - 2x}{2} \\ 2(\Delta - a) &\ge k - 2x \\ 2\Delta - 2a &\ge k - 2x \\ x - 2a &\ge k - x - 2\Delta. \end{aligned}$$

- *According to Property B in Lemma 4, $\rho(a+\Delta,\ x-2a) \ge \rho(a + \Delta,\ k - x - 2\Delta)$. Therefore, statement S2 holds.*

S3. It is shown that for any two natural numbers a and Δ such that $a \le \frac{x}{2}$, $\Delta \le (k-x)$, and $\Delta + a \ge \frac{k}{2}$, the following property holds:

$$\binom{x}{a}\binom{k-x}{\Delta}\rho(a+\Delta,\, x-2a) \ge \binom{k-x}{k-x-\Delta}\binom{x}{x-a}\rho(k-\Delta-a,\, x-k+2\Delta). \quad (35)$$

This statement can be proven as follows:

- *For any two natural numbers a and Δ such that $a \le \frac{x}{2}$ and $\Delta \le (k-x)$, the binomials are defined and their products are equal (and positive) on both sides of the inequality because*
  - $\binom{x}{a} = \binom{x}{x-a}$, *and*
  - $\binom{k-x}{\Delta} = \binom{k-x}{k-x-\Delta}$.
- *Note that $(a+\Delta)+(x-2a) = (k-\Delta-a)+(x-k+2\Delta)$ is true as shown below:*

$$\begin{aligned}(a+\Delta)+(x-2a) &\stackrel{?}{=} (k-\Delta-a)+(x-k+2\Delta)\\ x+\Delta-a &= x+\Delta-a.\end{aligned}$$

- *The condition $a \le \frac{x}{2}$ in S3 implies that $0 \le (x-2a)$.*
- *The condition $\Delta + a \ge \frac{k}{2}$ in S3 implies that $(x-2a) \le (x-k+2\Delta)$ as shown below:*

$$\begin{aligned}\Delta + a &\ge \frac{k}{2}\\ -2(\Delta+a) &\le -k\\ -2\Delta - 2a &\le -k\\ -2a &\le -k+2\Delta\\ x-2a &\le x-k+2\Delta.\end{aligned}$$

- *According to Property C in Lemma 4, $\rho(a+\Delta,\, x-2a) \ge \rho(k-\Delta-a,\, x-k+2\Delta)$ is true because $(a+\Delta)+(x-2a) = (k-\Delta-a)+(x-k+2\Delta)$ and $0 \le (x-2a) \le (x-k+2\Delta)$. Therefore, statement S3 must be true.*

S4. It is shown that for any two natural numbers a and Δ such that $\frac{x}{2} \le a \le x$, $\Delta \le (k-x)$, and $\Delta + a \le \frac{k}{2}$, the following property holds:

$$\binom{x}{a}\binom{k-x}{\Delta}\rho(a+\Delta,\, x-2a) \ge \binom{k-x}{k-x-\Delta}\binom{x}{x-a}\rho(k-\Delta-a,\, x-k+2\Delta). \quad (36)$$

This statement can be proven as follows:

- *For any two natural numbers a and Δ such that $\frac{x}{2} \le a \le x$ and $\Delta \le (k-x)$, the binomials are defined and their products are equal (and positive) on both sides of the inequality because*
  - $\binom{x}{a} = \binom{x}{x-a}$, *and*
  - $\binom{k-x}{\Delta} = \binom{k-x}{k-x-\Delta}$.
- *Note that $(a+\Delta)+(x-2a) = (k-\Delta-a)+(x-k+2\Delta)$ is true as shown below:*

$$\begin{aligned}(a+\Delta)+(x-2a) &\stackrel{?}{=} (k-\Delta-a)+(x-k+2\Delta)\\ x+\Delta-a &= x+\Delta-a.\end{aligned}$$

- *The condition $a \ge \frac{x}{2}$ in S4 implies that $(x-2a) \le 0$.*
- *The condition $\Delta + a \le \frac{k}{2}$ in S4 implies that $(x-2a) \ge (x-k+2\Delta)$ as shown below:*

$$\begin{aligned}\Delta + a &\le \frac{k}{2}\\ -2(\Delta+a) &\ge -k\\ -2\Delta - 2a &\ge -k\\ -2a &\ge -k+2\Delta\\ x-2a &\ge x-k+2\Delta.\end{aligned}$$

- *According to Property D in Lemma 4, $\rho(a+\Delta,\, x-2a) \ge \rho(k-\Delta-a,\, x-k+2\Delta)$ is true because $(a+\Delta)+(x-2a) = (k-\Delta-a)+(x-k+2\Delta)$ and $(x-k+2\Delta) \le (x-2a) \le 0$. Therefore, statement S4 must be true.*

S5. It is shown that for any two natural numbers a and Δ such that $a \le \frac{x}{2}$, $\Delta \le (k-x)$, $\Delta - a > \frac{k-2x}{2}$, and $\Delta + a < \frac{k}{2}$, if $\rho(a+\Delta,\, x-2a) < \rho(a+\Delta,\, k-x-2\Delta)$, there exists two natural numbers a' and Δ' such that

$$0 \le a' = \Delta - \frac{k-2x}{2} \le x \quad \textit{and}$$
$$0 \le \Delta' = a + \frac{k-2x}{2} \le k-x,$$

and the following property holds:

$$\begin{aligned}&\binom{x}{a}\binom{k-x}{\Delta}\rho(a+\Delta,\, x-2a) + \binom{x}{a'}\binom{k-x}{\Delta'}\rho(a'+\Delta',\, x-2a') \ge\\ &\binom{k-x}{\Delta}\binom{x}{a}\rho(a+\Delta,\, k-x-2\Delta) + \binom{k-x}{\Delta'}\binom{x}{a'}\rho(a'+\Delta',\, k-x-2\Delta'). \quad (37)\end{aligned}$$

The statement is proven by showing that the following statements hold under the conditions given in S5:

- $0 \le a' \le x$,
- $0 \le \Delta' \le k-x$,
- $\rho(a+\Delta,\, x-2a) = \rho(a'+\Delta',\, k-x-2\Delta')$,
- $\rho(a'+\Delta',\, x-2a') = \rho(a+\Delta,\, k-x-2\Delta)$, *and*
- $\binom{x}{a}\binom{k-x}{\Delta} \le \binom{k-x}{\Delta'}\binom{x}{a'}$.

The proof steps are as follows:

- $a' \geq 0$ *holds because*

$$\begin{aligned} \Delta - a &> \frac{k-2x}{2} && \textit{cf., S5} \\ \Delta - \frac{k-2x}{2} &> a \\ a' &> a && \textit{cf., S5} \\ a' &\geq 0 && \textit{since } a \geq 0. \end{aligned}$$

- $a' \leq x$ *holds because*

$$\begin{aligned} \Delta + a &< \frac{k}{2} && \textit{cf., S5} \\ \Delta &< \frac{k}{2} - a \\ \Delta - \left(\frac{k-2x}{2}\right) &< \frac{k}{2} - a - \left(\frac{k-2x}{2}\right) \\ \Delta - \left(\frac{k-2x}{2}\right) &< \cancel{\frac{k}{2}} - a - \cancel{\frac{k}{2}} + x \\ \Delta - \left(\frac{k-2x}{2}\right) &< x - a \\ a' &< x - a \\ a' &< x - a \leq x && \textit{since } a \geq 0 \\ a' &\leq x. \end{aligned}$$

- $\Delta' \geq 0$ *holds because*

$$\begin{aligned} \frac{k}{2} &\geq x && \textit{cf., Lemma 5} \\ \frac{k}{2} - x &\geq 0 \\ \frac{k-2x}{2} &\geq 0 \\ a + \frac{k-2x}{2} &\geq 0 && \textit{since } a \geq 0 \\ \Delta' &\geq 0. \end{aligned}$$

- $\Delta' \leq k - x$ *holds because*

$$\begin{aligned} \Delta - a &> \frac{k-2x}{2} && \textit{cf., S5} \\ \Delta &> a + \frac{k-2x}{2} \\ k - x \geq \Delta &> a + \frac{k-2x}{2} && \textit{since } \Delta \leq k - x \\ k - x &\geq a + \frac{k-2x}{2} \\ k - x &\geq \Delta'. \end{aligned}$$

- $\rho(a + \Delta,\, x - 2a) = \rho(a' + \Delta',\, k - x - 2\Delta')$ *holds because* $a + \Delta = a' + \Delta'$ *and* $x - 2a = k - x - 2\Delta'$, *as shown below:*

$$\begin{aligned} &a+\Delta \overset{?}{=} a'+\Delta' \\ &a+\Delta \overset{?}{=} \Delta - \cancel{\left(\frac{k-2x}{2}\right)} + a + \cancel{\left(\frac{k-2x}{2}\right)} \quad \textit{cf., S5} \\ &a+\Delta = \Delta + a, \end{aligned}$$

and

$$\begin{aligned} &x-2a \overset{?}{=} k-x-2\Delta' \\ &x-2a \overset{?}{=} \cancel{k}-x-2a-\cancel{k}+2x \\ &\qquad\qquad \textit{since } \Delta' = a + \frac{k-2x}{2} \\ &x-2a = x-2a. \end{aligned}$$

- $\rho(a' + \Delta',\, x - 2a') = \rho(a + \Delta,\, k - x - 2\Delta)$ *holds because* $a + \Delta = a' + \Delta'$, *as shown above, and* $x - 2a' = k - x - 2\Delta$, *as shown below:*

$$\begin{aligned} x - 2a' &\overset{?}{=} k - x - 2\Delta \\ x - 2\Delta + 2\Delta' - 2a &\overset{?}{=} k - x - 2\Delta \\ &\qquad \textit{since } a' = \Delta - \Delta' + a \\ x - 2\Delta + \cancel{2a} + k - 2x - \cancel{2a} &\overset{?}{=} k - x - 2\Delta \\ &\qquad \textit{since } \Delta' = a + \frac{k-2x}{2} \\ x - 2\Delta + k - 2x &\overset{?}{=} k - x - 2\Delta \\ k - x - 2\Delta &= k - x - 2\Delta. \end{aligned}$$

- *Let* $M = a' - a = \Delta - \Delta' > 0$ *and* $N = \Delta - a' = \Delta' - a = \frac{k-2x}{2} \geq 0$ *(conditions trivially follow from earlier proof steps). Then,* $\binom{x}{a}\binom{k-x}{\Delta} \leq \binom{k-x}{\Delta'}\binom{x}{a'}$ *is true, as shown below:*

$$\binom{x}{a}\binom{k-x}{\Delta} = \frac{x!}{a!(x-a)!} \times \frac{(k-x)!}{(a+M+N)!(k-x-a-M-N)!}$$

and

$$\binom{x}{a'}\binom{k-x}{\Delta'} = \frac{x!}{(a+M)!(x-a-M)!} \times \frac{(k-x)!}{(a+N)!(k-x-a-N)!}.$$

Since the numerators are the same (and positive) in both of the equations above, $\binom{x}{a}\binom{k-x}{\Delta} \leq \binom{k-x}{\Delta'}\binom{x}{a'}$ *iff*

$$\frac{a!(x-a)!}{(a+M)!(x-a-M)!} \times \frac{(a+M+N)!(k-x-a-M-N)!}{(a+N)!(k-x-a-N)!} \geq 1.$$

By expanding the factorials, one obtains

$$\frac{\overbrace{(x-a)\times\cdots\times(x-a-M+1)}^{M \textit{ times}}}{\underbrace{(a+M)\times\cdots\times(a+1)}_{M \textit{ times}}} \times \frac{\overbrace{(a+N+M)\times\cdots\times(a+N+1)}^{M \textit{ times}}}{\underbrace{(k-x-a-N)\times\cdots\times(k-x-a-N-M+1)}_{M \textit{ times}}} \geq 1.$$

The inequality above can be simplified further, as shown below:

$$\prod_{i=1}^{M} \frac{(a+N+i)}{(a+i)} \times \frac{(x+1)-(a+i)}{(k-x+1)-(a+N+i)} \geq 1. \quad (38)$$

Note that

$$\begin{aligned}(a+N+i)+(x+1)-(a+i) &= (a+i)+(k-x+1) \\ &\quad -(a+N+i) \\ N+x+1 &= k-x+1-N \\ \frac{k}{2}-x+x+1 &= k-x+1-\frac{k}{2}+x \\ &\qquad \textit{since } N = \frac{k-2x}{2} \\ \frac{k}{2}+1 &= \frac{k}{2}+1.\end{aligned}$$

Therefore, Inequality 38 can be restated as:

$$\prod_{i=1}^{M} \frac{(a+N+i)}{(a+i)} \times \frac{\left((\frac{k}{2}+1)-(a+N+i)\right)}{\left((\frac{k}{2}+1)-(a+i)\right)} \geq 1. \quad (39)$$

Note that Inequality 39 holds iff

$$\begin{aligned}&\left(\frac{k}{2}+1-\big((a+1)+(a+M)\big)\right)^2 \\ &\geq \left(\frac{k}{2}+1-\big((a+N+1)+(a+N+M)\big)\right)^2\end{aligned}$$

That is,

$$\begin{aligned}\left(\frac{k}{2}+\cancel{1}-\big(2a+M+\cancel{1}\big)\right)^2 &\geq \left(\frac{k}{2}+\cancel{1}-\big(2a+M+\cancel{1}+2N\big)\right)^2 \\ \left(\frac{k}{2}-2a-M\right)^2 &\geq \left(\big(\frac{k}{2}-2a-M\big)-2N\right)^2.\end{aligned}$$

Let $\xi = \frac{k}{2} - 2a - M$*, then, the inequality can be restated as:*

$$\begin{aligned}\left(\xi\right)^2 &\geq \left(\xi - 2N\right)^2 \\ \cancel{\xi^2} &\geq \cancel{\xi^2} - 4N\xi + 4N^2 \\ \cancel{4N}\xi &\geq \cancel{4N} \times N \qquad \textit{since } N \geq 0 \\ \frac{k}{2} - 2a - M &\geq N \\ \cancel{\frac{k}{2}} - 2a - M &\geq \cancel{\frac{k}{2}} - x \qquad \textit{since } N = \frac{k-2x}{2} \\ 2a + M &\leq x. \qquad (40)\end{aligned}$$

Note that $M = \Delta - \Delta' = \Delta - a - \frac{k}{2} + x$*. Therefore,*

$$\begin{aligned}2a + M &= 2a + \Delta - a - \frac{k}{2} + x \\ &= a + \Delta - \frac{k}{2} + x.\end{aligned}$$

Substituting for $2a + M$ *in Inequality 40, it is possible to obtain*

$$\begin{aligned}a + \Delta - \frac{k}{2} + \cancel{x} &\leq \cancel{x} \\ a + \Delta &\leq \frac{k}{2}. \qquad (41)\end{aligned}$$

Since $a + \Delta < \frac{k}{2}$ *is a precondition of S5, Inequalities 38–41 must be true. Consequently,* $\binom{x}{a}\binom{k-x}{\Delta} \leq \binom{k-x}{\Delta'}\binom{x}{a'}$ *is true.*

- *Since* $\rho(a+\Delta,\, x-2a) < \rho(a+\Delta,\, k-x-2\Delta)$*, the following statements must be true:*

  $\rho(a + \Delta,\, x - 2a) = \rho(a' + \Delta',\, k - x - 2\Delta') = 0,$ *and*

  $\rho(a' + \Delta',\, x - 2a') = \rho(a + \Delta,\, k - x - 2\Delta) = 1.$

  Hence, Inequality 37 can be simplified to:

  $$\binom{x}{a'}\binom{k-x}{\Delta'} \geq \binom{k-x}{\Delta}\binom{x}{a},$$

  which was shown to be true. Therefore, S5 holds.

S6. *It is shown that for any two natural numbers* a *and* Δ *such that* $\frac{x}{2} \leq a \leq x$*,* $\Delta \leq (k-x)$*,* $\Delta - a < \frac{k-2x}{2}$*, and* $\Delta + a > \frac{k}{2}$*, if* $\rho(a+\Delta,\, x-2a) < \rho(a+\Delta,\, k-x-2\Delta)$*, there exists two natural numbers* a' *and* Δ' *such that*

$$\begin{aligned}0 \leq a' = \Delta - \frac{k-2x}{2} \leq x \quad \textit{and} \\ 0 \leq \Delta' = a + \frac{k-2x}{2} \leq k - x,\end{aligned}$$

and Inequality 37 holds. The statement is proven by showing that the following statements hold under the conditions given in S6:

- $0 \leq a' \leq x$*,*
- $0 \leq \Delta' \leq k - x$*,*
- $\rho(a + \Delta,\, x - 2a) = \rho(a' + \Delta',\, k - x - 2\Delta')$*,*
- $\rho(a' + \Delta',\, x - 2a') = \rho(a + \Delta,\, k - x - 2\Delta)$*, and*
- $\binom{x}{a}\binom{k-x}{\Delta} \leq \binom{k-x}{\Delta'}\binom{x}{a'}$.

The proof steps are as follows:

- $a' \geq 0$ *holds because*

  $$\begin{aligned}\Delta + a &> \frac{k}{2} \qquad \textit{cf., S6} \\ \Delta + x &> \frac{k}{2} \qquad \textit{since } a \leq x \\ \Delta - \frac{k}{2} + x &> 0 \\ \Delta - \frac{k-2x}{2} &> 0 \\ a' &> 0. \qquad \textit{cf., S6}\end{aligned}$$

- $a' \leq x$ *holds because*

  $$\begin{aligned}\Delta - a &< \frac{k-2x}{2} \qquad \textit{cf., S6} \\ \Delta - \frac{k-2x}{2} &< a \\ \Delta - \frac{k-2x}{2} &< x \qquad \textit{since } a \leq x \\ a' &< x.\end{aligned}$$

- $\Delta' \geq 0$ *holds because*

$$\begin{aligned} \Delta - a &< \frac{k-2x}{2} && \textit{cf., S6} \\ \Delta &< a + \frac{k-2x}{2} \\ 0 &< a + \frac{k-2x}{2} && \textit{since } \Delta \geq 0 \\ 0 &< \Delta'. \end{aligned}$$

- $\Delta' \leq k - x$ *holds because*

$$\begin{aligned} a \leq x &\leq \frac{k}{2} && \textit{cf., S6 and Lemma 5} \\ a &\leq \frac{k}{2} \\ a + \frac{k}{2} - x &\leq \frac{k}{2} + \frac{k}{2} - x \\ a + \frac{k-2x}{2} &\leq k - x \\ \Delta' &\leq k - x. \end{aligned}$$

- $\rho(a + \Delta,\, x - 2a) = \rho(a' + \Delta',\, k - x - 2\Delta')$ *and* $\rho(a' + \Delta',\, x - 2a') = \rho(a + \Delta,\, k - x - 2\Delta)$ *hold for the same reasons discussed in S5 (i.e., the proofs are independent of the conditions in S6).*

- *Let* $M = a - a' = \Delta' - \Delta > 0$ *and* $N = \Delta - a' = \Delta' - a = \frac{k-2x}{2} \geq 0$ *(conditions trivially follow from earlier proof steps). Then,* $\binom{x}{a}\binom{k-x}{\Delta} \leq \binom{k-x}{\Delta'}\binom{x}{a'}$ *is true, as shown below:*

$$\binom{x}{a}\binom{k-x}{\Delta} = \frac{x!}{a!(x-a)!} \times \frac{(k-x)!}{(a+N-M)!(k-x-a-N+M)!}$$

and

$$\binom{x}{a'}\binom{k-x}{\Delta'} = \frac{x!}{(a-M)!(x-a+M)!} \times \frac{(k-x)!}{(a+N)!(k-x-a-N)!}.$$

Since the numerators are the same (and positive) in both of the equations above, $\binom{x}{a}\binom{k-x}{\Delta} \leq \binom{k-x}{\Delta'}\binom{x}{a'}$ *iff*

$$\frac{a!(x-a)!}{(a-M)!(x-a\,!+M)!} \times \frac{(a+N-M)!(k-x-a-N+M)!}{(a+N)!(k-x-a-N)!} \geq 1.$$

By expanding the factorials, one obtains

$$\frac{\overbrace{(a) \times \cdots \times (a-M+1)}^{M \ \textit{times}}}{\underbrace{(x-a+M) \times \cdots \times (x-a+1)}_{M \ \textit{times}}} \times \frac{\overbrace{(k-x-a-N+M) \times \cdots \times (k-x-a-N+1)}^{M \ \textit{times}}}{\underbrace{(a+N) \times \cdots \times (a+N-M+1)}_{M \ \textit{times}}} \geq 1.$$

The inequality above can be simplified further, as shown below:

$$\prod_{i=1}^{M} \frac{(a-M+i)}{(x-a+i)} \times \frac{(k-x-a-N+M+1-i)}{(a+N+1-i)} \geq 1. \tag{42}$$

Note that

$$\begin{aligned} (\cancel{a}-\cancel{M}+\cancel{i})+(k-x-\cancel{a}-N+\cancel{M}+1-\cancel{i}) &\stackrel{?}{=} \frac{k}{2}+1 \\ k-x-N+1 &\stackrel{?}{=} \frac{k}{2}+1 \\ k-\cancel{x}-\frac{k}{2}+\cancel{x}+1 &\stackrel{?}{=} \frac{k}{2}+1 \\ \frac{k}{2}+1 &= \frac{k}{2}+1. \end{aligned}$$

and

$$\begin{aligned} (x-\cancel{a}+\cancel{i})+(\cancel{a}+N+1-\cancel{i}) &\stackrel{?}{=} \frac{k}{2}+1 \\ x+N+1 &\stackrel{?}{=} \frac{k}{2}+1 \\ \cancel{x}+\frac{k}{2}-\cancel{x}+1 &\stackrel{?}{=} \frac{k}{2}+1 \\ \frac{k}{2}+1 &= \frac{k}{2}+1. \end{aligned}$$

Therefore, Inequality 42 can be restated as:

$$\prod_{i=1}^{M} \frac{(a-M+i)}{(x-a+i)} \times \frac{\left(\frac{k}{2}+1\right)-\left(a-M+i\right)}{\left(\frac{k}{2}+1\right)-\left(x-a+i\right)} \geq 1. \tag{43}$$

Note that Inequality 43 holds iff

$$\begin{aligned} &\Big(\big(\frac{k}{2}+1\big)-\big((x-a+1)+(x-a+M)\big)\Big)^2 \geq \\ &\quad \Big(\big(\frac{k}{2}+1\big)-\big((a-M+1)+(a-\cancel{M}+\cancel{M})\big)\Big)^2 \\ &\Big(\big(\frac{k}{2}+\cancel{1}\big)-\big(2x-2a+M+\cancel{1}\big)\Big)^2 \geq \\ &\quad \Big(\big(\frac{k}{2}+\cancel{1}\big)-\big(2a-M+\cancel{1}\big)\Big)^2 \\ &\Big(\frac{k}{2}-2x+2a-M\Big)^2 \geq \\ &\quad \Big(\frac{k}{2}-2a+M\Big)^2. \end{aligned} \tag{44}$$

Since $M = N-(\Delta-a) = \frac{k}{2}-x-\Delta+a$,

$$\Big(\frac{k}{2}-2x+2a-M\Big)^2 = \Big(\cancel{\frac{k}{2}}-\cancel{2}x+\cancel{2}a-\cancel{\frac{k}{2}}+\cancel{x}+\Delta-\cancel{a}\Big)^2 = \Big(a-x+\Delta\Big)^2$$

and

$$\Big(\frac{k}{2}-2a+M\Big)^2 = \Big(\frac{k}{2}-\cancel{2}a+\frac{k}{2}-x-\Delta+\cancel{a}\Big)^2 = \Big(k-a-x-\Delta\Big)^2.$$

Therefore, Inequality 44 can be simplified to:

$$\Big(a-x+\Delta\Big)^2 \geq \Big(k-a-x-\Delta\Big)^2 \tag{45}$$

or to

$$\Big(\Delta+a-x\Big)^2 \geq \Big(k-x-(\Delta+a)\Big)^2. \tag{46}$$

Note that $\Delta+a > \frac{k}{2}$ *and* $\frac{k}{2} \geq x$, *therefore,* $\Delta+a-x > 0$, *and there are only two cases to consider:*

- *Case I:* $(k-x)-(\Delta+a) \geq 0$, *and*
- *Case II:* $(k-x)-(\Delta+a) < 0$.

For Case I, it needs to be shown that

$$(\Delta+a)-x \overset{?}{\geq} (k-x)-(\Delta+a)$$
$$2(\Delta+a) \overset{?}{\geq} k-2x$$
$$\Delta+a \overset{?}{\geq} \frac{k}{2}-x$$

Since $\Delta+a > \frac{k}{2}$ *and* $x \geq 0$,

$$\Delta+a \geq \frac{k}{2}-x.$$

For Case II, it needs to be shown that

$$\cancel{(\Delta+a)}-x \overset{?}{\geq} \cancel{(\Delta+a)}-(k-x)$$
$$-x \overset{?}{\geq} -k+x$$
$$k \overset{?}{\geq} 2x,$$

which is true by definition. Therefore, Inequalities 38–41 must be true. Consequently, $\binom{x}{a}\binom{k-x}{\Delta} \leq \binom{k-x}{\Delta'}\binom{x}{a'}$ *is true.*

- *Since* $\rho(a+\Delta,\, x-2a) < \rho(a+\Delta,\, k-x-2\Delta)$, *the following statements must be true:*

  $\rho(a+\Delta,\, x-2a) = \rho(a'+\Delta',\, k-x-2\Delta') = 0$, *and*
  $\rho(a'+\Delta',\, x-2a') = \rho(a+\Delta,\, k-x-2\Delta) = 1.$

  Hence, Inequality 37 can be simplified to:

  $$\binom{x}{a'}\binom{k-x}{\Delta'} \geq \binom{k-x}{\Delta}\binom{x}{a},$$

  which was shown to be true. Therefore, S6 holds.

S7. According to Equation 5, the probabilities $\text{PR}_{==}\big(\vec{r}_1,\, \vec{r}_2,\, x\big)$ *and* $\text{PR}_{==}\big(\vec{r}_3,\, \vec{r}_4,\, k-x\big)$ *can be expanded as the summation of products that are shown in Table 2. For brevity, the constant multiplier*

$$\frac{\binom{k}{k-x}}{2^{2k}}$$

has been omitted from both equations.

S8. Each term on the left-hand side of Table 2 can be paired up with a term on the right-hand side as follows:

- *Pair-up* $\binom{x}{a}\binom{k-x}{\Delta}\rho(a{+}\Delta,\, x{-}2a)$ *with* $\binom{k-x}{a'}\binom{x}{\Delta'}\rho(a'{+}\Delta',\, k{-}x{-}2a')$, *where* $a' = \Delta$ *and* $\Delta' = a$ *if*

  $\big(0\leq a\leq\frac{x}{2},\quad \Delta\leq(k-x)$ *and* $\Delta-a\leq\frac{k-2x}{2}\big)$ *or*
  $\big(\frac{x}{2}\leq a\leq x,\quad \Delta\leq(k-x)$ *and* $\Delta-a\geq\frac{k-2x}{2}\big)$.

- *Pair-up* $\binom{x}{a}\binom{k-x}{\Delta}\rho(a{+}\Delta,\, x{-}2a)$ *with* $\binom{k-x}{a'}\binom{x}{\Delta'}\rho(a'{+}\Delta',\, k{-}x{-}2a')$, *where* $a' = k-x-\Delta$ *and* $\Delta' = x-a$ *if*

  $\big(0\leq a\leq\frac{x}{2},\quad \Delta\leq(k-x)$ *and* $\Delta+a\geq\frac{k}{2}\big)$ *or*
  $\big(\frac{x}{2}\leq a\leq x,\quad \Delta\leq(k-x)$ *and* $\Delta+a\leq\frac{k}{2}\big)$.

- *Else, pair-up* $\binom{x}{a}\binom{k-x}{\Delta}\rho(a{+}\Delta,\, x{-}2a)$ *with* $\binom{k-x}{a'}\binom{x}{\Delta'}\rho(a'{+}\Delta',\, k{-}x{-}2a')$, *where* $a' = \Delta$ *and* $\Delta' = a$.

S9. Based on the pairings outlined in S8, it can be shown that the summation of products on the left-hand side of Table 2 is always greater than or equal to the summation of products on the right-hand side. The reason is as follows: For the first case in S8, statements S1 and S2 suggest that the product on the left-hand side of a pairing is greater than or equal to the product on the right-hand side. For the second case in S8, the same conclusion can be reached using statements S3 and S4. Whether the product on the left-hand side of a pairing in the third case in S8 is greater than or equal to the product on the right-hand side is questionable. If it is greater than or equal to the product on the right-hand side, then there is no problem. Otherwise, it needs to be checked whether the difference in the products (which is a loss for the summation on the left-hand side) is compensated elsewhere. Fortunately, statements S5 and S6 suggest that there exists another pairing (already covered by the first four cases) where the difference in products compensates the loss in the summation of the left-hand side. It can be easily inferred from the conditions of S5 and S6 that for each questionable (and problematic) case, the pairing that compensates the loss is a distinct one. Therefore, the summation on the left-hand side of Table 2 must be greater than or equal to the summation on the right-hand side, thus, proving Lemma 5. □

Lemma 6. *Let* $\delta^M(\vec{r}_i,\, \vec{r}_j)$ *denote the Manhattan distance between any two record utilization vectors* $\vec{r}_i$ *and* $\vec{r}_j$, *and let* m *denote the number of rightmost bits that are dropped by* h_2, *which is utilized in* h *(cf., Definition 1). Furthermore, let* $\text{PR}_{==}\big(\vec{r}_i,\, \vec{r}_j,\, x\big)$ *denote the posterior probability that, for any two record utilization vectors* $\vec{r}_i$ *and* $\vec{r}_j$, $h(\vec{r}_i) = h(\vec{r}_j)$ *provided that* $\delta^M(\vec{r}_i,\, \vec{r}_j) = x$. *Given any four record utilization vectors* $\vec{r}_1$, $\vec{r}_2$, $\vec{r}_3$ *and* $\vec{r}_4$ *with size* λb *such that*

$\delta^M(\vec{r}_1, \vec{r}_2) = x$ and $\delta^M(\vec{r}_3, \vec{r}_4) = \lambda b - x$, the following property holds for any $x \leq \frac{\lambda}{2}$ and $m = \Upsilon b$:

$$\text{PR}_{==}\big(\vec{r}_1, \vec{r}_2, x\big) \geq \text{PR}_{==}\big(\vec{r}_3, \vec{r}_4, \lambda b - x\big) \qquad (47)$$

where $h = h_2 \circ h_1$ (cf., Definition 1), $b \in \mathbb{Z}_{1\cdots\infty}$ denotes the number of entries in the record utilization counters produced by h_1, λ is an even and positive number, and $\Upsilon \in \mathbb{Z}_{1\cdots\lambda-1}$.

Proof 6. *Lemma 6 is proven by induction on b.*
***Base case**: Equation 47 holds when $b = 1$. The proof sketch is below: The proof trivially follows from Lemma 5.*
***Inductive Step**: Assuming that Lemma 6 holds for $b \leq \alpha$ where α is a natural number greater than or equal to 1, it needs to be shown that it also holds for $b = \alpha + 1$. The proof steps are as follows:*

S1. Let $\vec{r}_i[j]$ denote the group of bits in a record utilization vector r_i that have the same hash value j with respect to the hash function f, which is utilized within h_1 (cf., Definition 47). (Assume that within each $\vec{r}_i[j]$, the ordering of bits in $\vec{r}_i$ is preserved.)

S2. Recall that the Manhattan distance between any two record utilization vectors is the summation of their individual Manhattan distances within each group of bits that share the same hash value with respect to f (cf., Equation 2). Therefore, since $\delta^M(\vec{r}_1, \vec{r}_2) = x$,

$$\text{if } \delta^M(\vec{r}_1[\alpha], \vec{r}_2[\alpha]) = a, \text{ then}$$
$$\delta^M(\vec{r}_1[0]\cdots\vec{r}_1[\alpha-1], \vec{r}_2[0]\cdots\vec{r}_2[\alpha-1]) = x - a$$

where $\vec{r}_i[0]\cdots\vec{r}_i[\alpha-1]$ denotes the concatenation of the corresponding bit vectors.

S3. Also note that $h(\vec{r}_1) = h(\vec{r}_2)$ if and only if $h(\vec{r}_1[j]) = h(\vec{r}_2[j])$ for all $j \in \mathbb{Z}_{0\cdots\alpha}$. The reason is that $h = h_2 \circ h_1$ and $h(\vec{r}_1)$ (respectively, $h(\vec{r}_2)$) corresponds to the bit vector that is produced by interleaving the bits in the binary representations of $h_1(\vec{r}_1[0]) \ldots h_1(\vec{r}_1[\alpha])$ (respectively, $h_1(\vec{r}_2[0]) \ldots h_1(\vec{r}_2[\alpha])$) and cutting off the rightmost Υb bits [55]. Consequently, for $h(\vec{r}_1) = h(\vec{r}_2)$ to be true, for all $j \in \mathbb{Z}_{0\cdots\alpha}$, $h_1(\vec{r}_1[j])$ and $h_1(\vec{r}_2[j])$ must have the same sequence of bits except for the rightmost Υ, which means that $h(\vec{r}_1[j])$ and $h(\vec{r}_2[j])$.

S4. As a consequence of statements S2 and S3, the following property holds:

$$\text{PR}_{==}\big(\vec{r}_1, \vec{r}_2, x\big) = \sum_{a=0}^{x} \begin{array}{l}\text{PR}_{==}\big(\vec{r}_1[0]\cdots\vec{r}_1[\alpha-1], \vec{r}_2[0]\cdots\vec{r}_2[\alpha-1], x-a\big) \times \\ \text{PR}_{==}\big(\vec{r}_1[\alpha], \vec{r}_2[\alpha], a\big) \times \text{PR}_a\end{array}$$

where $\text{PR}_a \in \{\text{PR}_0, \ldots, \text{PR}_x\}$ denotes a constant that represents the probability that $\delta^M(\vec{r}_1[\alpha], \vec{r}_2[\alpha]) = a$ among all possible configurations such that $\delta^M(\vec{r}_1, \vec{r}_2) = x$.

S5. Statement S2 holds also for $\vec{r}_3$ and $\vec{r}_4$; therefore, since $\delta^M(\vec{r}_3, \vec{r}_4) = \lambda b - x$,

$$\text{if } \delta^M(\vec{r}_3[\alpha], \vec{r}_4[\alpha]) = a', \text{ then}$$
$$\delta^M(\vec{r}_3[0]\cdots\vec{r}_3[\alpha-1], \vec{r}_4[0]\cdots\vec{r}_4[\alpha-1]) = \lambda b - x - a'$$

where $\vec{r}_i[0]\cdots\vec{r}_i[\alpha-1]$ denotes the concatenation of the corresponding bit vectors.

S6. Since there are $\lambda b - \lambda$ bits in $\vec{r}_3[0]\cdots\vec{r}_3[\alpha-1]$ and $\vec{r}_4[0]\cdots\vec{r}_4[\alpha-1]$, the edit distance between these two bit vectors can be at most $\lambda b - \lambda$, thus,

$$\delta^M(\vec{r}_3[0]\cdots\vec{r}_3[\alpha-1], \vec{r}_4[0]\cdots\vec{r}_4[\alpha-1]) \leq \lambda b - \lambda.$$

Therefore, $(\lambda - x) \leq a' \leq \lambda$ must hold.

S7. Consequently, similar to S4, the following statement must be true:

$$\text{PR}_{==}\big(\vec{r}_3, \vec{r}_4, \lambda b - x\big) = \sum_{a'=\lambda-x}^{\lambda} \begin{array}{l}\text{PR}_{==}\big(\vec{r}_3[0]\cdots\vec{r}_3[\alpha-1], \vec{r}_4[0]\cdots\vec{r}_4[\alpha-1], \lambda b - x - a'\big) \\ \times\, \text{PR}_{==}\big(\vec{r}_3[\alpha], \vec{r}_4[\alpha], a'\big) \times \text{PR}_{a'}\end{array}$$

where $\text{PR}_{a'} \in \{\text{PR}_{\lambda-x}, \ldots, \text{PR}_\lambda\}$ denotes a constant that represents the probability that $\delta^M(\vec{r}_3[\alpha], \vec{r}_4[\alpha]) = a'$ among all possible configurations such that $\delta^M(\vec{r}_3, \vec{r}_4) = \lambda b - x$.

S8. The possible configurations in the summations in S4 and S7 for $\text{PR}_{==}\big(\vec{r}_1, \vec{r}_2, x\big)$ and $\text{PR}_{==}\big(\vec{r}_3, \vec{r}_4, \lambda b - x\big)$ can be paired up as shown in Table 3.

S9. For each pair of configurations in Table 3, (i) $a \leq x \leq \frac{\lambda}{2} \leq \lambda - a$ and (ii) $a + \lambda - a = \lambda$ hold. Therefore, Lemma 5 suggests that

$$\text{PR}_{==}\big(\vec{r}_1[\alpha], \vec{r}_2[\alpha], a\big) \geq \text{PR}_{==}\big(\vec{r}_3[\alpha], \vec{r}_4[\alpha], \lambda - a\big).$$

S10. For each pair of configurations in Table 3, $(x - a) \leq \frac{\lambda(b-1)}{2} \leq \lambda b - x - \lambda + a$ holds:

- *$(x - a) \leq \frac{\lambda(b-1)}{2}$ is true because*
  - *$x \leq \frac{\lambda}{2}$ by definition (cf., Lemma 6);*
  - *$a \geq 0$, therefore, $(x - a) \leq \frac{\lambda}{2}$; and*
  - *$\frac{\lambda}{2} \leq \frac{\lambda(b-1)}{2}$ for $b \geq 2$, which is the case for b in the inductive step.*
- *$\lambda b - x - \lambda + a \geq \frac{\lambda(b-1)}{2}$ is true because*
  - *$x \leq \frac{\lambda}{2}$ (cf., Lemma 6) and $a \geq 0$, therefore,*

$$\begin{aligned}\lambda b - \lambda - x + a &\geq \lambda b - \lambda - x \\ &\geq \lambda b - \lambda - \frac{\lambda}{2} \\ &\geq \lambda b - \frac{3\lambda}{2}.\end{aligned}$$

  - *For $b \geq 2$, which is true for the inductive step, $\lambda b - \frac{3\lambda}{2} \geq \frac{\lambda(b-1)}{2}$ must also hold, therefore $\lambda b - x - \lambda + a \geq \frac{\lambda(b-1)}{2}$ is true.*

Consequently, according to the inductive assumption, the following statement must be true:

$$\begin{aligned}&\text{PR}_{==}\big(\vec{r}_1[0]\cdots\vec{r}_1[\alpha-1], \vec{r}_2[0]\cdots\vec{r}_2[\alpha-1], x-a\big) \\ &\quad\geq \text{PR}_{==}\big(\vec{r}_3[0]\cdots\vec{r}_3[\alpha-1], \vec{r}_4[0]\cdots\vec{r}_4[\alpha-1], \lambda b - x - \lambda + a\big).\end{aligned}$$

S11. For each pair of configurations in Table 3, $\text{PR}_a = \text{PR}'_{\lambda-a}$

because

$$\text{PR}_a = \frac{\binom{\lambda}{\lambda - x}}{2^\lambda},$$

$$\text{PR}'_{\lambda-a} = \frac{\binom{\lambda}{\lambda-\lambda+a}}{2^\lambda} \qquad = \frac{\binom{\lambda}{a}}{2^\lambda} \qquad \textit{and}$$

$$\binom{\lambda}{\lambda - x} = \binom{\lambda}{x}.$$

S12. According to statements S9, S10 and S11, for each pair of configurations in Table 3, the product on the left hand side is always greater than or equal to the product on the right hand side, which means:

$$\text{PR}_{==}(\vec{r}_1,\, \vec{r}_2,\, x) \geq \text{PR}_{==}(\vec{r}_3,\, \vec{r}_4,\, \lambda b - x).$$

Thus, Lemma 47 holds also for $b = \alpha + 1$.

Theorem 7. *Let $\delta^M(\vec{r}_i, \vec{r}_j)$ denote the Manhattan distance between any two record utilization vectors $\vec{r}_i$ and $\vec{r}_j$ with size λb, and let m denote the number of rightmost bits that are dropped by h_2, which is utilized in h (cf., Definition 1). For any $\delta^M \leq \frac{\lambda}{2}$, h is (δ^M, $\lambda b - \delta^M$, PR_1, PR_2)-sensitive for some $\text{PR}_1 \geq \text{PR}_2$ where $h = h_2 \circ h_1$ (cf., Definition 1), λ is an even and positive number, $b \in \mathbb{Z}_{1\,\cdots\,\infty}$ denotes the number of entries in the record utilization counters produced by h_1, $m = \Upsilon b$, and $\Upsilon \in \mathbb{Z}_{1\,\cdots\,\lambda-1}$.*

Proof 7. *The proof steps are as follows:*

S1. PR_1 corresponds to the posterior probability that $h(\vec{r}_i) = h(\vec{r}_j)$ for any two record utilization vectors $\vec{r}_i$ and $\vec{r}_j$ with size λb such that $\delta^M(\vec{r}_i, \vec{r}_j) \leq \delta^M$ [64].

S2. PR_2 corresponds to the posterior probability that $h(\vec{r}_i) \neq h(\vec{r}_j)$ for any two record utilization vectors $\vec{r}_i$ and $\vec{r}_j$ with size λb such that $\delta^M(\vec{r}_i, \vec{r}_j) \geq \lambda b - \delta^M$ [64].

S3. Note that

$$\text{PR}_1 = \frac{\sum_{x=0}^{\delta^M} \text{PR}_{==}(\vec{r}_i,\, \vec{r}_j,\, x)\, \binom{\lambda b}{\lambda b - x}}{2^{\lambda b}}$$

$$\text{PR}_2 = \frac{\sum_{x=0}^{\delta^M} \text{PR}_{==}(\vec{r}_i,\, \vec{r}_j,\, \lambda b - x)\, \binom{\lambda b}{x}}{2^{\lambda b}}.$$

S4. For all $x \in \mathbb{Z}_{0\,\cdots\,\delta^M}$ $\binom{\lambda b}{\lambda b - x} = \binom{\lambda b}{x}$.

S5. Therefore, for all $x \in \mathbb{Z}_{0\,\cdots\,\delta^M}$

$$\frac{\binom{\lambda b}{\lambda b - x}}{2^{\lambda b}} = \frac{\binom{\lambda b}{x}}{2^{\lambda b}}.$$

S6. According to Lemma 6, $\text{PR}_{==}(\vec{r}_i,\, \vec{r}_j,\, x) \geq \text{PR}_{==}(\vec{r}_i,\, \vec{r}_j,\, \lambda b - x)$. Consequently, $\text{PR}_1 \geq \text{PR}_2$. □

Theorem 7 suggests that h is a function with locality-sensitive properties, and can be used to approximate the clustering problem. However, it must be noted that the sensitivity analysis of h is conservative. In other words, it is believed that stronger statements can be made about h, in particular, due to the empirical observation that $\text{PR}_{==}(\vec{r}_i,\, \vec{r}_j,\, x)$ is a monotonically decreasing function. Proving this conjecture is left as future work.

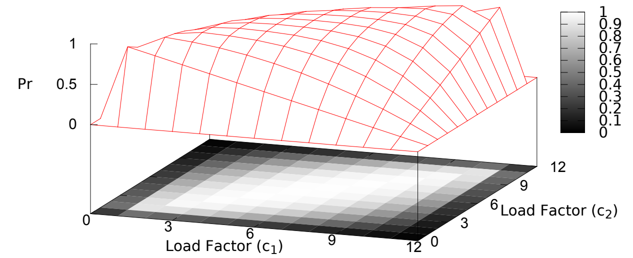


Figure 3: $\text{PR}(\delta^M \neq \delta)$ for $k = 12$, $b = 1$ and across varying load factors

5.2 Achieving and Maintaining Tighter Bounds on Adaptive-LSH

Next, we demonstrate how it is possible to reduce the approximation error of h_1. We first define *load factor* of a record utilization counter entry.

Definition 2 (Load Factor)**.** *Given a record utilization counter $\vec{c} = (c[0], \ldots, c[b-1])$ with size b, the* load factor *of the i^{th} entry is $c[i]$.*

Theorem 8 (Effects of Grouping)**.** *Given two record utilization vectors $\vec{r_1}$ and $\vec{r_2}$ with size k, let $\vec{c_1}$ and $\vec{c_2}$ denote two record utilization counters with size $b = 1$ such that $\vec{c_1} = h_1(\vec{r_1})$ and $\vec{c_2} = h_1(\vec{r_2})$. Then,*

$$\text{PR}\left(\begin{array}{c} \delta^M(\vec{c_1}, \vec{c_2}) \\ = \delta^M(\vec{r_1}, \vec{r_2}) \end{array} \middle| \begin{array}{l} c_1[0] = l_1 \text{ AND} \\ c_2[0] = l_2 \end{array} \right) = \gamma \tag{48}$$

where

$$\gamma = \frac{\binom{l_{max}}{l_{min}}\binom{k}{l_{max}}}{\binom{k}{l_{max}}\binom{k}{l_{min}}} \tag{49}$$

and

$$l_{max} = max(l_1, l_2)$$
$$l_{min} = min(l_1, l_2).$$

Proof 8. *Let $\vec{r}_{max}$ denote the record utilization vector with the most number of 1-bits among $\vec{r_1}$ and $\vec{r_2}$, and let $\vec{r}_{min}$ denote the vector with the least number of 1-bits. When $b = 1$, $\delta^M(\vec{c_1}, \vec{c_2}) = \delta(\vec{r_1}, \vec{r_2})$ holds if and only if the number of 1-bits on which $\vec{r_1}$ and $\vec{r_2}$ are aligned is l_{min} because in that case, both $\delta^M(\vec{c_1}, \vec{c_2})$ and $\delta(\vec{r_1}, \vec{r_2})$ are equal to $l_{max} - l_{min}$ (note that $\delta^M(\vec{c_1}, \vec{c_2})$ is always equal to $l_{max} - l_{min}$). Assuming that the positions of 1-bits in $\vec{r}_{max}$ are fixed, there are $\binom{l_{max}}{l_{min}}$ possible ways of arranging the 1-bits of $\vec{r}_{min}$ such that $\delta(\vec{r}_1, \vec{r}_2) = l_{max} - l_{min}$. Since the 1-bits of $\vec{r}_{max}$ can be arranged in $\binom{k}{l_{max}}$ different ways, there are $\binom{l_{max}}{l_{min}}\binom{k}{l_{max}}$ combinations such that $\delta^M(\vec{c}_1, \vec{c}_2) = \delta(\vec{r}_1, \vec{r}_2)$. Note that in total, the bits of $\vec{r_1}$ and $\vec{r_2}$ can be arranged in $\binom{k}{l_{max}}\binom{k}{l_{min}}$ possible ways; therefore, Eqns. 48 and 49 describe the posterior probability that $\delta^M(\vec{c_1}, \vec{c_2}) = \delta(\vec{r_1}, \vec{r_2})$, given $c_1[0] = l_1$ and $c_2[0] = l_2$.* ■

According to Eqns. 48 and 49 in Thm. 8, the probability that $\delta^M(\vec{c_1}, \vec{c_2})$ is an approximation of $\delta(\vec{r_1}, \vec{r_2})$, but that it is not exactly equal to $\delta(\vec{r_1}, \vec{r_2})$ is lower for load factors that are close or equal to zero and likewise for load factors that are close or equal to $\lceil \frac{k}{b} \rceil$ (cf., Fig. 3). This property suggests that by carefully choosing f, it is possible to achieve even tighter error bounds for h_1. For $b \geq 2$, the probabilities for each group of bits need to be multiplied, which is illustrated in the proof of Lemma 6 in the extended version of this paper (noted earlier). Therefore, the algorithm for tuning f aims to make sure that the load factors are either low or high for as many of the groups as possible.

Contrast the matrices in Fig 2b and Fig 2c, which contain the same query access vectors, but the columns are grouped in two different ways[3]: (i) in Fig. 2b, the grouping is based on the original sequence of execution, and (ii) in Fig. 2c, queries with similar access patterns are grouped together. Fig. 2d and Fig. 2e represent the corresponding record utilization counters for the record utilization vectors in the matrices in Fig. 2b and Fig. 2c, respectively. Take $\vec{r_3}$ and $\vec{r_5}$, for instance. Their actual Hamming distance with respect to q_0–q_7 is 8. Now consider the transformed matrices. According to Fig. 2d, the Hamming distance lower bound is 0, whereas according to Fig. 2e, it is 8. Clearly, the bounds in the second representation are closer to the original. The reason is as follows. Even though $\vec{r_3}$ and $\vec{r_5}$ differ on all the bits for q_0–q_7, when the bits are grouped as in Fig. 2b, the counts alone cannot distinguish the two bit vectors. In contrast, if the counts are computed based on the grouping in Fig. 2c (which clearly places the 1-bits in separate groups), the counts indicate that the two bit vectors are indeed different.

The observations above are in accordance with Thm. 8. Consequently, we make the following optimization. Instead of randomly choosing a hash function, we construct f such that it maps queries with similar access vectors (i.e., columns in the matrix) to the same hash value. This way, it is possible to obtain record utilization counters with entries that have either very high or very low load factors (cf., Def. 1), thus, decreasing the probability of error (cf., Thm. 8).

We develop a technique to efficiently determine groups of queries with similar access patterns and to adaptively maintain these groups as the access patterns change. Our approach consists of two parts: (i) to approximate the similarity between any two queries, we rely on the MIN-HASH scheme [22], and (ii) to adaptively group similar queries, we develop an incremental version of a multidimensional scaling (MDS) algorithm [54].

MIN-HASH offers a quick and efficient way of approximating the similarity, (more specifically, the Jaccard similarity [47]), between two sets of integers. Therefore, to use it, the query access vectors in our conceptualization need to be translated into a set of positional identifiers that correspond to the records for which the bits in the vector are set to 1.[4] For example, according to Fig. 2a, $\vec{q_1}$ should be represented with the set $\{0, 5, 6\}$ because r_0, r_5 and r_6 are the only records for which the bits are set to 1. Note that, we do not need to store the original query access vectors at all. In fact, after the access patterns over a query are determined, we compute and store only its MIN-HASH value. This is important for keeping the memory overhead of our algorithm low.

Queries with similar access patterns are grouped together using a multidimensional scaling (MDS) algorithm [51] that was originally developed for data visualization, and has recently been used for clustering [20]. Given a set of points and a distance function, MDS assigns coordinates to points such that their original distances are preserved as much as possible. In one efficient implementation [54], each point is initially assigned a random set of coordinates, but these coordinates are adjusted iteratively based on a spring-force analogy. That is, it is assumed that points exert a force on each other that is proportional to the difference between their actual and observed distances, where the latter refers to the distance that is computed from the algorithm-assigned coordinates. These forces are used for computing the current velocity (V in Table 4) and the approximated coordinates of a point (X in Table 4). The intuition is that, after successive iterations, the system will reach equilibrium, at which point the approximated coordinates can be reported. Since computing all pairwise distances can be prohibitively expensive, the algorithm relies on a combination of sampling ($S[]$ in Table 4) and maintaining a list of each point's nearest neighbours ($N[]$ in Table 4)—only these distances are used in computing the net force acting on a point. Then, the nearest neighbours are updated in each iteration by removing the most distant neighbour of a point and replacing it with a new point from the random sample if the distance between the point and the random sample is smaller than the distance between the point and its most distant neighbour.

There are multiple reasons for our choice of MDS over other clustering approaches such as single-linkage clustering or k-means [48] (even though ADAPTIVELSH is agnostic to this choice). First of all, MDS is less sensitive to the shape of the underlying clusters (e.g., as opposed to single-linkage) and the initial choice of clusters (e.g., as opposed to k-means) largely due to the reliance on random sampling (i.e., $S[]$). Second, it allows the trade-off between computational overhead vs. quality of clustering to be tuned more precisely by controlling multiple parameters such as (i) the size of vectors $S[]$ and $N[]$ (independently), (ii) the number of iterations, and (iii) the number of output dimensions. Nevertheless, this algorithm cannot be used directly for our purposes because it is not incremental. Therefore, we propose a revised MDS algorithm that incorporates the following modifications:

1. In our case, each point in the algorithm represents a query access vector. However, since we are not interested in visualizing these points, but rather clustering them, we configure the algorithm to place these points along a single dimension (this can be increased for better clustering accuracy, but was not necessary in our

Symbol	*Description*
begin	natural number between $0 \ldots (k-1)$, initial value is 0
size	natural number between $0 \ldots (k-1)$, keeps track of the number of query access vectors that are currently being maintained, initial value is 0
$H_{k\times ?}$	matrix that contains MIN-HASH values for each query access vector
$S[]$	array of *vector*(s), one for each MDS query point, that pairs each MDS query point with a *random* subset of points
$N[]$	array of *max-heap*(s), one for each MDS query point, that pairs each MDS query point with a set of *neighboring* points
$X[]$	array of *float*(s), represents the coordinate (single dimensional) of each MDS query point
$V[]$	array of *float*(s), represents the current (directional) velocity of each MDS query point

Table 4: Data structures referenced in algorithms

[3]Groups are separated by vertical dashed lines.

[4]In practice, this translation is not required because the system maintains positional vectors instead.

Algorithm 4 Reconfigure-F

Require:
$\vec{q_t}$: query access vector produced at time t
Ensure:
Coordinates of MDS points are updated, which are used in determining the outcome of f

```
1: procedure RECONFIGURE-F(q_t)
2:     pos ← (begin + size) % k
3:     S[pos].clear()
4:     N[pos].clear()
5:     X[pos] ← −0.5 + rand() / RAND-MAX
6:     V[pos] ← 0
7:     H[pos] ← MIN-HASH(q_t)
8:     if size < k then
9:         size += 1
10:    else
11:        begin = (begin + 1) % k
12:    end if
13:    for i ← 0, i < size, i++ do
14:        x ← (begin + i) % k
15:        UPDATE-S-AND-N(x)
16:        UPDATE-VELOCITY(x)
17:    end for
18:    for i ← 0, i < size, i++ do
19:        x ← (begin + i) % k
20:        UPDATE-COORDINATES(x)
21:    end for
22: end procedure
```

experiments). Then, by dividing the coordinate space into consecutive regions, we are able to determine similar query access vectors.

2. Instead of computing the coordinates of all of the points at once, our version makes incremental adjustments to the coordinates every time reconfiguration is needed.

The revised algorithm is given in Algorithm 4. First, the algorithm decides which MDS point to assign to the new query access vector $\vec{q_t}$ (line 2). It clears the array and the heap data structures containing, respectively, (i) the randomly sampled, and (ii) the neighbouring set of points (lines 3–4). Furthermore, it assigns a random coordinate to the point within the interval $[-0.5, 0.5]$ (line 5), and resets its velocity to 0 (line 6). Next, it computes the Min-Hash value of $\vec{q_t}$ and stores it in $H[\text{pos}]$ (line 7). Then, it makes two passes over all the points in the system (lines 13–21), while first updating their sample and neighbouring lists (line 15), computing the net forces acting on them based on the Min-Hash distances and updating their velocities (line 16); and then updating their coordinates (line 20).

The procedures used in the last part are implemented in a similar way as the original algorithm [54]; that is, in line 15, the sampled points are updated, in line 16, the velocities assigned to the MDS points are updated, and in line 20, the coordinates of the MDS points are updated based on these updated velocities. However, our implementation of the Update-Velocity procedure (line 16) is slightly different than the original. In particular, in updating the velocities, we use a decay function so that the algorithm forgets "old" forces that might have originated from the elements in $S[]$ and $N[]$ that have been assigned to new query access vectors in the meantime. Note that unless one keeps track of the history of all the forces that have acted on every point in the system, there is no other way of "undoing" or "forgetting" these "old" forces.

Given the sequence number of a query access vector (t), the outcome of the hash function f is determined based on the coordinates of the MDS point that had previously been

Algorithm 5 Hash Function f

Require:
t: sequence number of a query access vector
Ensure:
$f(t)$ is computed and returned

```
procedure F(t)
    pos ← t % k
    (lo, hi) ← GROUP-BOUNDS(X[pos])
    coid ← CENTROID(lo, hi)
    return HASH(coid) % b
end procedure
```

assigned to the query access vector by the Reconfigure procedure. To this end, the k points are sorted based on their coordinates, and the coordinate space is divided into b groups containing points with consecutive coordinates such that there are at most $\lceil \frac{k}{b} \rceil$ points in each group. Then, one option is to use the group identifier, which is a number in $\mathbb{Z}_{0\cdots b-1}$, as the outcome of f, but there is a problem with this naïve implementation. Specifically, we observed that even though the *relative* coordinates of MDS points within the "same" group may not change significantly across successive calls to the Reconfigure procedure, points within a group, as a whole, may shift. This is an inherent (and in fact, a desirable) property of the incremental algorithm. However, the problem is that there may be far too many cases where the group identifier of a point changes just because the absolute coordinates of the group have changed, even though the point continues to be part of the "same" group. To solve this problem, we rely on a method of computing the centroid within a group by taking the Min-Hash of the identifiers of points within that group such that these centroids rarely change across successive iterations. Then, we rely on the identifier of the centroid, as opposed to its coordinates, to compute the group number, hence, the outcome of f. The pseudocode of this procedure is given in Algorithm 5.

We make one last observation. Internally, Min-Hash uses multiple hash functions to approximate the degree to which two sets are similar [22]. It is also known that increasing the number of internal hash functions used (within Min-Hash) should increase the overall accuracy of the Min-Hash scheme. However, as unintuitive as it may seem, in our approach, we use only a single hash function within Min-Hash, yet, we are still able to achieve sufficiently high accuracy. The reason is as follows. Recall that Algorithm 4 relies on multiple pairwise distances to position every point. Consequently, even though individual pairwise distances may be inaccurate (because we are just using a single hash function within Min-Hash), collectively the errors are cancelled out, and points can be positioned accurately on the MDS coordinate space.

5.3 Resetting Old Entries in Record Utilization Counters

Once the group identifier is computed (cf., Algorithm 5), it should be straightforward to update the record utilization counters (cf., line 8 in Algorithm 2). However, unless we maintain the original query access vectors, we have no way of knowing which counters to decrement when a query access vector becomes stale, as maintaining these original query access vectors is prohibitively expensive. Therefore, we develop a more efficient scheme in which old values can also be removed from the record utilization counters.

	0	**1**	**2**	**3**	**4**	**5**
$t = 0$	$\square$	$\square$	$\square$	$\emptyset$		
$t = \lceil ^k/_3 \rceil$		$\square$	$\square$	$\square$	$\emptyset$	
$t = \lceil ^{2k}/_3 \rceil$			$\square$	$\square$	$\square$	$\emptyset$
$t = k$	$\emptyset$			$\square$	$\square$	$\square$
$t = \lceil ^{4k}/_3 \rceil$	$\square$	$\emptyset$			$\square$	$\square$
$t = \lceil ^{5k}/_3 \rceil$	$\square$	$\square$	$\emptyset$			$\square$

Figure 4: Assuming $b = 3$, $\square$ indicates the allowed locations at each time tick, and $\emptyset$ indicates the counter to be reset.

Instead of maintaining b entries in every record utilization counter, we maintain twice as many entries ($2b$). Then, whenever the Tune procedure is called, instead of directly using the outcome of $f(t)$ to locate the counters to be incremented, we map $f(t)$ to a location within an "allowed" region of consecutive entries in the record utilization counter (cf., line 8 in Algorithm 2). At every $\lceil \frac{k}{b} \rceil^{\text{th}}$ iteration, this allowed region is shifted by one to the right, wrapping back to the beginning if necessary. Consider Fig. 4. Assuming that $b = 3$ and that at time $t = 0$ the allowed region spans entries from 0 to $(b - 1)$, at time $t = \lceil \frac{k}{b} \rceil$, the region will span entries from 1 to b; at time $t = k$, the region will span entries from b to $2b - 1$; and at time $t = \lceil \frac{4k}{b} \rceil$, the region will span entries 0 and those from $b + 1$ to $2b - 1$.

Since $f(t)$ produces a value between 0 and $b-1$ (inclusive), whereas the entries are numbered from 0 to $2b - 1$ (inclusive), the Reconfigure procedure in Algorithm 2 uses $f(t)$ as follows. If the outcome of $f(t)$ directly corresponds to a location in the allowed region, then it is used. Otherwise, the output is incremented by b (cf., line 8 in Algorithm 2). Whenever the allowed region is shifted to the right, it may land on an already incremented entry. If that is the case, that entry is reset, thereby allowing "old" values to be forgotten (cf., line 11 in Algorithm 2). These are shown by $\emptyset$ in Fig. 4. This scheme guarantees any query access pattern that is less than k steps old is remembered, while any query access pattern that is more than $2k$ old is forgotten.

6. EXPERIMENTAL EVALUATION

In this section, we evaluate AdaptiveLSH in three sets of experiments. First, we evaluate it within a hashtable implementation since hashtables are used extensively in RDF data management systems. Second, we evaluate it within *System-XYZ*, our prototype RDF data management system. Finally, we evaluate AdaptiveLSH in isolation, to understand how it behaves under different types of workloads. All experiments are performed on a commodity machine with AMD Phenom II $\times 4$ 955 3.20 GHz processor, 16 GB of main memory and a hard disk drive with 100 GB of free disk space. The operating system is Ubuntu 12.04 LTS.

6.1 Self-Clustering Hashtable

The first experiment evaluates an in-memory hashtable that we developed that uses AdaptiveLSH to dynamically cluster records in the hashtable. Hashtables are commonly used in RDF data management systems. For example, the dictionary in an RDF data management system, which maps integer identifiers to URIs or literals (and vice versa), is often implemented as a hashtable [4, 30, 66]. Secondary indexes can also be implemented as hashtables, whereby the hashtable acts as a key-value store and maps tuple identifiers to the content of the tuples. In fact, in our own prototype RDF system, *System-XYZ*, all the indexes are secondary (dense) indexes because instead of relying on any sort order inherent in the data, RDF triples are ordered purely based on the workload.

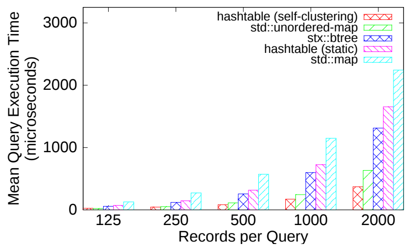
The hashtable interface is very similar to that of a standard one except that users can mark the beginning and end of queries. This information is used to dynamically cluster records such that those that are co-accessed across similar sets of queries also become physically co-located. All of the clustering and re-clustering is transparent to the user, hence, the name, *self-clustering hashtable*.

The self-clustering hashtable has the following advantages and disadvantages: compared to a standard hashtable that tries to avoid hash-collisions, it deliberately co-locates records that are accessed together. If the workloads favour a scenario in which many records are frequently accessed together, then we can expect the self-clustering hashtable to have improved fetch times due to better CPU cache utilization, prefetching, etc. [8]. On the other hand, these optimizations come with three types of overhead. First, every time a query is executed, AdaptiveLSH needs to be updated (cf., Algorithms 2 and 4). Second, compared to a standard hashtable in which the physical address of a record is determined solely using the underlying hash function (which is deterministic throughout the entire workload), in our case the physical address of a record needs to be maintained dynamically because the underlying hash function is not deterministic (i.e., it is also changing dynamically throughout the workload). Consequently, there is the overhead of going to a lookup table and retrieving the physical address of a record. Third, physically moving records around in the storage system takes time—in fact, this is often an expensive operation. Therefore, the objective of this set of experiments is twofold: (i) to evaluate the circumstances under which the self-clustering hashtable outperforms other popular data structures, and (ii) to understand when the tuning overhead may become a bottleneck. Consequently, we report the end-to-end query execution times, and if necessary, break it down into the time to (i) *fetch* the records, and (ii) *tune* the data structures (which includes all types of overhead listed above).

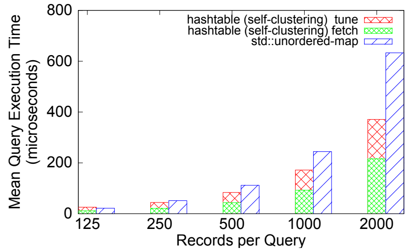
In our experiments, we compare the self-clustering hashtable to popular implementations of three data structures. Specifically, we use: (i) *std::unordered_map* [3], which is the C++ standard library implementation of a hashtable, (ii) *std::map* [2], which is the C++ standard library implementation of a red-black tree, and (iii) *stx::btree* [19], which is an open source in-memory B+ tree implementation. As a baseline, we also include a static version of our hashtable, i.e., one that does not rely on AdaptiveLSH.

We consider two types of workloads: one in which records are accessed *sequentially* (i.e., based on their physical ordering in the storage system) and the other in which records are accessed *randomly*. Each workload consists of 3000 synthetically generated queries. For each data structure, we measure the end-to-end workload execution time and compute the mean time by dividing the total workload execution time by the number of queries in the workload.

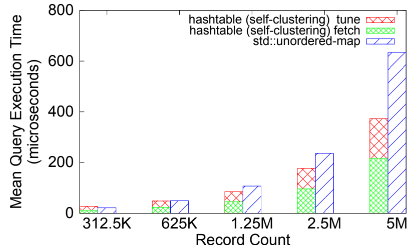
Queries in these workloads consist of changing query access patterns, and in different experiments, we control different parameters such as the number of records that are accessed by queries on average, the rate at which the query



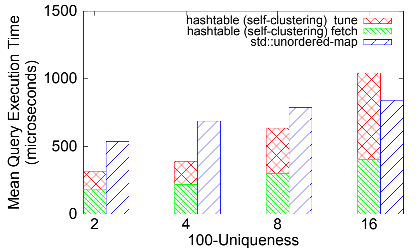
(a) Random Access (All Data Structures) – Control load per query



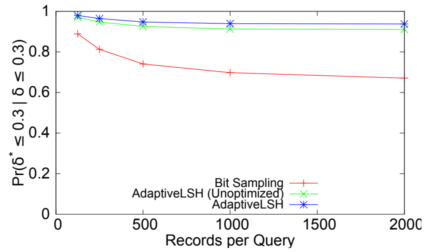
(b) Random Access – Control load per query



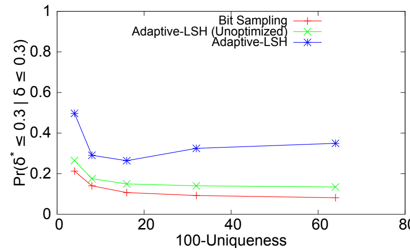
(c) Random Access – Control record count, keep record size constant at 128 bytes



(d) Random Access – Control workload dynamism



(e) Sensitivity analysis of AdaptiveLSH– Control load per query



(f) Sensitivity analysis of AdaptiveLSH– Control workload dynamism

Figure 5: Experimental evaluation of AdaptiveLSH in a self-clustering hashtable and the sensitivity analysis of AdaptiveLSH

access patterns change in the workload, etc. We repeat each experiment 20 times over workloads that are randomly generated with the same characteristics (e.g., average number of records accessed by each query, how fast the workload changes, etc.) and report averages across these 20 runs. We do not report standard errors as they are negligibly small.

For the sequential case, *stx::btree* and *std::map* outperform the hashtables, which is expected because once the first few records are fetched from main-memory, the remaining ones can already be prefetched into the CPU cache (due to the predictability of the sequential access pattern). Therefore, for the remaining part, we focus on the random access scenario, which can be a bottleneck even in systems like RDF-3x [57] that have clustered indexes over all permutations of attributes. For a detailed explanation, we refer the reader to [12].

In this experiment, we control the number of records that a query needs to access (on average), where each record is 128 bytes. Fig. 5a compares all the data structures with respect to their end-to-end (mean) query execution times. Three observations stand out: first, in the random access case, the self-clustering hashtable as well as the standard hashtable perform much better than the other data structures, which is what would be expected. This observation holds also for the subsequent experiments, therefore, for presentation purposes, we do not include these data structures in Fig. 5b–5d. Second, the baseline static version of our hashtable (i.e., without AdaptiveLSH) performs much worse than the standard hashtable, even worse than a B+ tree. This suggests that our implementation can be optimized further, which might improve the performance of the self-clustering hashtable as well (this is left as future work). Third, as the number of records that a query needs to access increases, the self-clustering hashtable outperforms all the other data structures, which verifies our initial hypothesis.

For the same experiment above, Fig. 5b focuses on the self-clustering hashtable versus the standard hashtable, and illustrates why the performance improvement is higher (for the self-clustering hashtable) for workloads in which queries access more records. Note that while the *fetch* time of the self-clustering hashtable scales proportionally with respect to std::unordered_map, the *tune* overhead is proportionally much lower for workloads in which queries access more records. This is because with increasing "records per query count", records can be re-located in batches across the pages in main-memory as opposed to moving individual records around.

Next, we keep the average number of records that a query needs to access constant at 2000, but control the number of records in the database. As in the previous experiment, each record is 128 bytes. As illustrated in Fig. 5c, increasing the number of records in the database (i.e., scaling-up) favours the self-clustering hashtable. The reason is that, when there are only a few records in the database, the records are likely clustered to begin with. We repeat the same experiment, but this time, by controlling the record size and keeping the database size constant at 640 megabytes. Surprisingly, the relative improvement with respect to the standard hashtable remains more or less constant, which indicates that the improvement is largely dominated by the size of the database, and increasing it is to the advantage of the self-clustering hashtable.

Finally, we evaluate how sensitive the self-clustering hashtable is to the dynamism in the workloads. Note that for the self-clustering hashtable to be useful at all, the workloads need to be somewhat predictable. That is, if records are physically clustered but are never accessed in the future, then the clustering efforts are wasted. To verify this hypothesis, we control the expected number of query clusters (i.e., queries with similar but not exactly the same access vectors) in any 100 consecutive queries in the workloads that

	System-XYZ	RDF-3x	VOS [6.1]	VOS [7.1]	MonetDB	4Store
WatDiv 100M	**42.0**	71.4	210.3	96.4	62.7	767.2

Table 5: Query execution time, geometric mean (milliseconds)

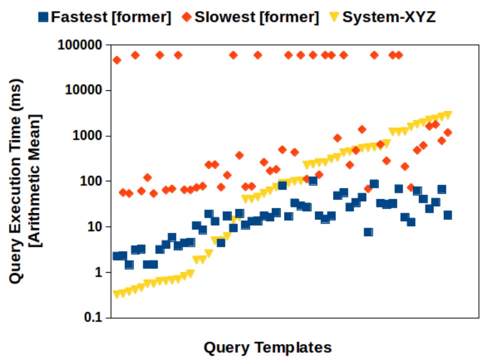


Figure 6: Comparison of System-XYZ to existing solutions using WatDiv 100M triples

we generate. Let us call this property of the workload its 100-*Uniqueness*. Fig. 5d illustrates how the *tuning* overhead can start to become a bottleneck as the workloads become more and more dynamic, to the extent of being completely unique, i.e., each query in the workload accesses a distinct set of records.

6.2 Adaptive-LSH in System-XYZ

The second experiment evaluates AdaptiveLSH within *System-XYZ*, which is our prototype RDF data management system. In our evaluations, the Waterloo SPARQL Diversity Test Suite (WatDiv) is used because it can generate dynamic workloads that are far more diverse than existing benchmarks [11]. In this regard, we use the WatDiv *data generator* to create a dataset with 100 million RDF triples. Then, using the WatDiv *query template generator*, 125 query templates are generated and each query template is instantiated with 600 queries.

We compare our approach with five popular systems: RDF-3x [57], MonetDB [43], 4Store [39] and Virtuoso Open Source (VOS) versions 6.1 [31] and 7.1 [30]. RDF-3x follows the single-table approach and creates multiple indexes; MonetDB is a column-store, where RDF data are represented using vertical partitioning [4]; and the last three systems are industrial systems. Both 4Store and VOS group and index data primarily based on RDF predicates, but VOS 6.1 is a row-store whereas VOS 7.1 is a column-store. We configure these systems so that they make as much use of the available main memory as possible.

We evaluate each system independently on each query template. Specifically, for each query template, we first warm up the system by executing the first 100 queries in the workload, then, we execute the remaining 500. In case of System-XYZ, after the execution of the 100^{th} query, we allow the system to update its physical layout using AdaptiveLSH. For each system, we report average query execution times over the last 500 queries for each query template. In case of System-XYZ, the time to compute the new clustering of triples as well as physically updating the layout are included in these reported averages.

Our experiments indicate that System-XYZ is more robust than existing solutions. As shown in Table 5, overall, System-XYZ is significantly faster than any of the other five RDF data management systems. Our experiments also indicate that on average, the computational overhead of AdaptiveLSH is just 26.1 milliseconds, which is much smaller compared to the tuning overhead reported in some other solutions. For example, Goasdoué et al. [37] report that one could expect up to 30 minutes for computing materialized views in their RDF data management system.

Fig. 6 breaks down the experimental results of Table 5 across different query templates in the workload. For ease of presentation, for each query template, we show only the fastest and the slowest of the five systems in addition to System-XYZ. Note that while a system may be the fastest for a particular query template, it may also be the slowest for another. For queries in which System-XYZ was not the fastest, we perceive opportunities to optimize System-XYZ further by improving its query optimization algorithms as well as switching to a pipelined execution model. These optimizations are well beyond the scope of this paper and its focus on AdaptiveLSH, hence, are left as future work.

6.3 Sensitivity Analysis of Adaptive-LSH

In the final set of experiments, we evaluate the sensitivity of AdaptiveLSH in isolation, that is, without worrying about how it affects physical clustering, and compare it to three other hash functions: (i) a standard non-locality sensitive hash function [1], (ii) bit-sampling, which is known to be locality-sensitive for Hamming distances [46], and (iii) AdaptiveLSH without the optimizations discussed in Section 5. These comparisons are made across workloads with different characteristics (i.e., dense vs. sparse, dynamic vs. stable, etc.) where parameters such as the average number of records accessed per query and the expected number of query clusters within any 100-consecutive sequence of queries in the workload are controlled.

Our evaluations indicate that AdaptiveLSH generally outperforms its alternatives. Due to space considerations, we cannot present all of our results in detail. Therefore, we will summarize our most important observations.

Fig. 5e shows how the probability that *the evaluated hash functions place records with similar utilization vectors to nearby hash values* changes as the average number of records that each query accesses is increased. In computing these probabilities, both the original distances (i.e., δ) and the distances over the hashed values (i.e., δ^*) are normalized with respect to the maximum distance in each geometry. It can be observed that both versions of AdaptiveLSH are better than bit-sampling especially when the number of records in a query increases.

Fig. 5f shows how the quality of clustering changes as the workloads become more and more dynamic. As illustrated in Fig. 5f, AdaptiveLSH achieves higher probability even when the workloads are dynamic. Note that the unoptimized version of AdaptiveLSH behaves no worse than a static locality-sensitive hash function, such as bit sampling, which is aligned with the theorems in Section 5.1. We have not included the results on the standard non-locality sensitive hash function, because, as one might guess, it has a probability distribution that is completely unaligned with our clustering objectives.

7. CONCLUSIONS AND FUTURE WORK

In this paper, we introduce ADAPTIVELSH, which is a locality-sensitive hashing scheme, and demonstrate its use in clustering records in an RDF data management system. In particular, we keep track of records that are accessed by the same query but are fragmented across the pages in the database and use ADAPTIVELSH to decide, in constant-time, where a record needs to be placed in the storage system. ADAPTIVELSH takes into account the most recent query access patterns over the database, and uses this information to auto-tune such that records that are accessed across similar sets of queries are hashed as much as possible to the same or nearby pages in the storage system. This property distinguishes ADAPTIVELSH from existing locality-sensitive hash functions, which are static. Our experiments with (i) a version of our prototype RDF data management system that uses ADAPTIVELSH, (ii) a hashtable that relies on ADAPTIVELSH to dynamically cluster its records, and (iii) workloads that rigorously test the sensitivity of ADAPTIVELSH verify the significant benefits of ADAPTIVELSH.

As future work, it would be beneficial to address two challenges. First, the assumption that the last k queries are representative of the future queries in the workload can be relaxed. Second, it is worth exploring if ADAPTIVELSH can be used in a more general setting than just RDF systems. In fact, it should be possible to extend the idea of the self-clustering in-memory hashtable that we have implemented to a more general, distributed key-value store.

8. REFERENCES

[1] std::hash, 2015. http://www.cplusplus.com/reference/functional/hash/.

[2] std::map, 2015. http://www.cplusplus.com/reference/map/map/.

[3] std::unordered_map, 2015. http://www.cplusplus.com/reference/unordered_map/unordered_map/.

[4] D. J. Abadi, A. Marcus, S. R. Madden, and K. Hollenbach. SW-Store: a vertically partitioned DBMS for semantic web data management. *VLDB J.*, 18:385–406, 2009.

[5] M. Ackerman, S. Ben-David, and D. Loker. Towards property-based classification of clustering paradigms. In *Advances in Neural Information Proc. Systems 23, Proc. 24th Annual Conf. on Neural Information Proc. Systems*, pages 10–18, 2010.

[6] C. C. Aggarwal. A survey of stream clustering algorithms. In *Data Clustering: Algorithms and Applications*, pages 231–258. 2013.

[7] S. Agrawal, S. Chaudhuri, and V. R. Narasayya. Automated selection of materialized views and indexes in SQL databases. In *Proc. 26th Int. Conf. on Very Large Data Bases*, pages 496–505, 2000.

[8] A. Ailamaki, D. J. DeWitt, M. D. Hill, and D. A. Wood. DBMSs on a modern processor: Where does time go? In *Proc. 25th Int. Conf. on Very Large Data Bases*, pages 266–277, 1999.

[9] R. Al-Harbi, Y. Ebrahim, and P. Kalnis. Phd-store: An adaptive SPARQL engine with dynamic partitioning for distributed RDF repositories. *CoRR*, abs/1405.4979, 2014.

[10] G. Aluç, D. DeHaan, and I. T. Bowman. Parametric plan caching using density-based clustering. In *Proc. 28th Int. Conf. on Data Engineering*, pages 402–413, 2012.

[11] G. Aluç, O. Hartig, M. T. Özsu, and K. Daudjee. Diversified stress testing of rdf data management systems. pages 197–212, 2014.

[12] G. Aluç, M. T. Özsu, and K. Daudjee. Workload matters: Why RDF databases need a new design. *Proc. VLDB Endowment*, 7(10):837–840, 2014.

[13] G. Aluç, M. T. Özsu, K. Daudjee, and O. Hartig. chameleon-db: a workload-aware robust RDF data management system. Technical Report CS-2013-10, University of Waterloo, 2013.

[14] A. Andoni and P. Indyk. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. In *Proc. 47th Annual Symp. on Foundations of Computer Science*, pages 459–468, 2006.

[15] M. Arias, J. D. Fernández, M. A. Martínez-Prieto, and P. de la Fuente. An empirical study of real-world SPARQL queries. *CoRR*, abs/1103.5043, 2011.

[16] V. Athitsos, M. Potamias, P. Papapetrou, and G. Kollios. Nearest neighbor retrieval using distance-based hashing. In *Proc. 24th Int. Conf. on Data Engineering*, pages 327–336, 2008.

[17] R. G. Bello, K. Dias, A. Downing, J. J. Feenan, Jr., J. L. Finnerty, W. D. Norcott, H. Sun, A. Witkowski, and M. Ziauddin. Materialized views in oracle. In *Proc. 24th Int. Conf. on Very Large Data Bases*, pages 659–664, 1998.

[18] B. Berendt, L. Dragan, L. Hollink, M. Luczak-Rösch, E. Demidova, S. Dietze, J. Szymanski, and J. G. Breslin, editors. *Joint Proc. of the 5th International Workshop on Using the Web in the Age of Data and the 2nd International Workshop on Dataset PROFIling and fEderated Search for Linked Data*, volume 1362 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2015.

[19] T. Bingmann. STX B+ tree C++ template classes, 2007. https://panthema.net/2007/stx-btree/.

[20] B. Bislimovska, G. Aluç, M. T. Özsu, and P. Fraternali. Graph search of software models using multidimensional scaling. In *Proc. of the Workshops of the EDBT/ICDT 2015 Joint Conference*, pages 163–170, 2015.

[21] M. A. Bornea, J. Dolby, A. Kementsietsidis, K. Srinivas, P. Dantressangle, O. Udrea, and B. Bhattacharjee. Building an efficient RDF store over a relational database. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, pages 121–132, 2013.

[22] A. Z. Broder. On the resemblance and containment of documents. In *Proc. Compression and Complexity of Sequences*, pages 21–29, 1997.

[23] A. Z. Broder, M. Charikar, A. M. Frieze, and M. Mitzenmacher. Min-wise independent permutations. *J. Comput. Syst. Sci.*, 60(3):630–659, 2000.

[24] N. Bruno and S. Chaudhuri. To tune or not to tune? a lightweight physical design alerter. In *Proc. 32nd Int. Conf. on Very Large Data Bases*, pages 499–510, 2006.

[25] J. J. Carroll, I. Dickinson, C. Dollin, D. Reynolds, A. Seaborne, and K. Wilkinson. Jena: implementing the semantic web recommendations. In *Proc. 13th Int. World Wide Web Conf. - Alternate Track Papers & Posters*, pages 74–83, 2004.

[26] S. Ceri, S. B. Navathe, and G. Wiederhold. Distribution design of logical database schemas. *IEEE Trans. Softw. Eng.*, 9(4):487–504, 1983.

[27] M. Charikar. Similarity estimation techniques from rounding algorithms. In *Proc. 34th Annual ACM Symp. on Theory of Computing*, pages 380–388, 2002.

[28] S. Chaudhuri and V. Narasayya. Self-tuning database systems: a decade of progress. In *Proc. 33rd Int. Conf. on Very Large Data Bases*, pages 3–14, 2007.

[29] M. Datar, N. Immorlica, P. Indyk, and V. S. Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. In *Proc. 20th Annual Symp. on Computational Geometry*, pages 253–262, 2004.

[30] O. Erling. Virtuoso, a hybrid RDBMS/graph column store. *IEEE Data Eng. Bull.*, 35(1):3–8, 2012.

[31] O. Erling and I. Mikhailov. RDF support in the Virtuoso DBMS. *Networked Knowledge-Networked Media*, pages 7–24, 2009.

[32] H. Ferhatosmanoglu, E. Tuncel, D. Agrawal, and A. El Abbadi. Approximate nearest neighbor searching in multimedia databases. In *Proc. 17th Int. Conf. on Data Engineering*, pages 503–511, 2001.

[33] J. D. Foley, A. van Dam, S. K. Feiner, and J. F. Hughes. *Computer Graphics: Principles and Practice (2^{nd} edition)*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1990.

[34] K. R. French, G. W. Schwert, and R. F. Stambaugh. Expected stock returns and volatility. *Journal of Financial Economics*, pages 3–30, 1987.

[35] L. Galarraga, K. Hose, and R. Schenkel. Partout: a distributed engine for efficient RDF processing. In *Proc. 23rd Int. World*

Wide Web Conf., Companion Volume, pages 267–268, 2014.

[36] A. Gionis, P. Indyk, and R. Motwani. Similarity search in high dimensions via hashing. In *Proc. 25th Int. Conf. on Very Large Data Bases*, pages 518–529, 1999.

[37] F. Goasdoué, K. Karanasos, J. Leblay, and I. Manolescu. View selection in semantic web databases. *Proc. VLDB Endowment*, 5(2):97–108, 2011.

[38] R. W. Hamming, editor. *Coding and Information Theory*. Prentice-Hall, Englewood Cliffs, 1986.

[39] S. Harris, N. Lamb, and N. Shadbolt. 4store: The design and implementation of a clustered RDF store. In *Proc. 5th Int. Workshop on Scalable Semantic Web Knowledge Base Systems*, pages 81–96, 2009.

[40] A. Harth, J. Umbrich, A. Hogan, and S. Decker. Yars2: A federated repository for querying graph structured data from the web. In *Proc. 6th Int. Semantic Web Conference*, pages 211–224, 2007.

[41] K. Hose and R. Schenkel. WARP: workload-aware replication and partitioning for RDF. In *Proc. Workshops of the 29th IEEE Int. Conf. on Data Engineering*, pages 1–6, 2013.

[42] M. E. Houle and J. Sakuma. Fast approximate similarity search in extremely high-dimensional data sets. In *Proc. 21st Int. Conf. on Data Engineering*, pages 619–630, 2005.

[43] S. Idreos, F. Groffen, N. Nes, S. Manegold, K. S. Mullender, and M. L. Kersten. MonetDB: Two decades of research in column-oriented database architectures. *IEEE Data Eng. Bull.*, 35(1):40–45, 2012.

[44] S. Idreos, M. L. Kersten, and S. Manegold. Database cracking. In *Proc. 3rd Biennial Conf. on Innovative Data Systems Research*, pages 68–78, 2007.

[45] S. Idreos, S. Manegold, H. A. Kuno, and G. Graefe. Merging what's cracked, cracking what's merged: Adaptive indexing in main-memory column-stores. *PVLDB*, 4(9):585–597, 2011.

[46] P. Indyk and R. Motwani. Approximate nearest neighbors: Towards removing the curse of dimensionality. In *Proc. 30th Annual ACM Symp. on Theory of Computing*, pages 604–613, 1998.

[47] P. Jaccard. The distribution of flora in the alpine zone. *New phytologist*, 11(2):37–50, 1912.

[48] A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: a review. 31:264–323, 1999.

[49] M. Kirchberg, R. K. L. Ko, and B.-S. Lee. From linked data to relevant data – time is the essence. *CoRR*, abs/1103.5046, 2011.

[50] E. F. Krause, editor. *Taxicab Geometry: An Adventure in Non-Euclidean Geometry*. Dover, New York, 1986.

[51] J. B. Kruskal. Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis. *Psychometrika*, 29:1–27, 1964.

[52] V. I. Levenshtein. *Soviet Physics Doklady*, 10(8):707–710, 1966.

[53] S. Lightstone, T. J. Teorey, and T. P. Nadeau. *Physical Database Design: the database professional's guide to exploiting indexes, views, storage, and more*. Morgan Kaufmann, 2007.

[54] A. Morrison, G. Ross, and M. Chalmers. Fast multidimensional scaling through sampling, springs and interpolation. *Information Visualization*, 2(1):68–77, 2003.

[55] G. M. Morton. A computer oriented geodetic data base; and a new technique in file sequencing. Technical report, IBM Ltd., Ottawa, Canada, 1966.

[56] F. F.-H. Nah. A study on tolerable waiting time: how long are Web users willing to wait? *Behaviour & IT*, 23(3):153–163, 2004.

[57] T. Neumann and G. Weikum. The RDF-3X engine for scalable management of RDF data. *VLDB J.*, 19(1):91–113, 2010.

[58] T. Neumann and G. Weikum. x-RDF-3X: fast querying, high update rates, and consistency for RDF databases. *Proc. VLDB Endowment*, 3(1):256–263, 2010.

[59] N. Papailiou, I. Konstantinou, D. Tsoumakos, and N. Koziris. H2RDF: adaptive query processing on RDF data in the cloud. In *Proc. 21st Int. World Wide Web Conf., Companion Volume*, pages 397–400, 2012.

[60] N. Papailiou, D. Tsoumakos, P. Karras, and N. Koziris. Graph-aware, workload-adaptive SPARQL query caching. 2015. Forthcoming.

[61] W. Reed. The normal-Laplace distribution and its relatives. In *Proc. Advances in Distribution Theory, Order Statistics, and Inference*, pages 61–74, 2006.

[62] T. Sakaki, M. Okazaki, and Y. Matsuo. Earthquake shakes twitter users: real-time event detection by social sensors. In *Proc. 19th Int. World Wide Web Conf.*, pages 851–860, 2010.

[63] L. Sidirourgos, R. Goncalves, M. Kersten, N. Nes, and S. Manegold. Column-store support for RDF data management: not all swans are white. *Proc. VLDB Endowment*, 1(2):1553–1563, 2008.

[64] Y. Tao, K. Yi, C. Sheng, and P. Kalnis. Efficient and accurate nearest neighbor and closest pair search in high-dimensional space. *ACM Trans. Database Syst.*, 35(3), 2010.

[65] C. Weiss, P. Karras, and A. Bernstein. Hexastore: sextuple indexing for semantic web data management. *Proc. VLDB Endowment*, 1(1):1008–1019, 2008.

[66] K. Wilkinson. Jena property table implementation. Technical Report HPL-2006-140, HP-Labs, 2006.

[67] D. C. Zilio, J. Rao, S. Lightstone, G. M. Lohman, A. J. Storm, C. Garcia-Arellano, and S. Fadden. DB2 design advisor: Integrated automatic physical database design. In *Proc. 30th Int. Conf. on Very Large Data Bases*, pages 1087–1097, 2004.

[68] L. Zou, J. Mo, D. Zhao, L. Chen, and M. T. Özsu. gStore: Answering SPARQL queries via subgraph matching. *Proc. VLDB Endowment*, 4(1):482–493, 2011.

		$\mathrm{PR}_{==}\big(\vec{r}_1,\ \vec{r}_2,\ x\big)$				$\mathrm{PR}_{==}\big(\vec{r}_3,\ \vec{r}_4,\ k-x\big)$	
a	$\boldsymbol{\Delta}$			$\mathbf{a}'$	$\boldsymbol{\Delta}'$		
0	0	$\binom{x}{0}\binom{k-x}{0}$	$\rho(0,x)$	0	0	$\binom{k-x}{0}\binom{x}{0}$	$\rho(0,k-x)$
0	1	$\binom{x}{0}\binom{k-x}{1}$	$\rho(1,x)$	1	0	$\binom{k-x}{1}\binom{x}{0}$	$\rho(1,k-x-2)$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
0	$k-x-1$	$\binom{x}{0}\binom{k-x}{k-x-1}$	$\rho(k-x-1,x)$	$k-x-1$	0	$\binom{k-x}{k-x-1}\binom{x}{0}$	$\rho(k-x-1,-k+x+2)$
0	$k-x$	$\binom{x}{0}\binom{k-x}{k-x}$	$\rho(k-x,x)$	$k-x$	0	$\binom{k-x}{k-x}\binom{x}{0}$	$\rho(k-x,-k+x)$
1	0	$\binom{x}{1}\binom{k-x}{0}$	$\rho(1,x-2)$	0	1	$\binom{k-x}{0}\binom{x}{1}$	$\rho(1,k-x)$
1	1	$\binom{x}{1}\binom{k-x}{1}$	$\rho(2,x-2)$	1	1	$\binom{k-x}{1}\binom{x}{1}$	$\rho(2,k-x-2)$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
1	$k-x-1$	$\binom{x}{1}\binom{k-x}{k-x-1}$	$\rho(k-x,x-2)$	$k-x-1$	1	$\binom{k-x}{k-x-1}\binom{x}{1}$	$\rho(k-x,-k+x+2)$
1	$k-x$	$\binom{x}{1}\binom{k-x}{k-x}$	$\rho(k-x+1,x-2)$	$k-x$	1	$\binom{k-x}{k-x}\binom{x}{1}$	$\rho(k-x+1,-k+x)$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$x-1$	0	$\binom{x}{x-1}\binom{k-x}{0}$	$\rho(x-1,-x+2)$	0	$x-1$	$\binom{k-x}{0}\binom{x}{x-1}$	$\rho(x-1,k-x)$
$x-1$	1	$\binom{x}{x-1}\binom{k-x}{1}$	$\rho(x,-x+2)$	1	$x-1$	$\binom{k-x}{1}\binom{x}{x-1}$	$\rho(x,k-x-2)$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$x-1$	$k-x-1$	$\binom{x}{x-1}\binom{k-x}{k-x-1}$	$\rho(k-2,-x+2)$	$k-x-1$	$x-1$	$\binom{k-x}{k-x-1}\binom{x}{x-1}$	$\rho(k-2,-k+x+2)$
$x-1$	$k-x$	$\binom{x}{x-1}\binom{k-x}{k-x}$	$\rho(k-1,-x+2)$	$k-x$	$x-1$	$\binom{k-x}{k-x}\binom{x}{x-1}$	$\rho(k-1,-k+x)$
x	0	$\binom{x}{x}\binom{k-x}{0}$	$\rho(x,-x)$	0	x	$\binom{k-x}{0}\binom{x}{x}$	$\rho(x,k-x)$
x	1	$\binom{x}{x}\binom{k-x}{1}$	$\rho(x+1,-x)$	1	x	$\binom{k-x}{1}\binom{x}{x}$	$\rho(x+1,k-x-2)$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
x	$k-x-1$	$\binom{x}{x}\binom{k-x}{k-x-1}$	$\rho(k-1,-x)$	$k-x-1$	x	$\binom{k-x}{k-x-1}\binom{x}{x}$	$\rho(k-1,-k+x+2)$
x	$k-x$	$\binom{x}{x}\binom{k-x}{k-x}$	$\rho(k,-x)$	$k-x$	x	$\binom{k-x}{k-x}\binom{x}{x}$	$\rho(k,-k+x)$

Table 2: Pairings of products

a		$\mathbf{a}'$	
0	$\mathrm{PR}_0\ \times$ $\mathrm{PR}_{==}\big(\vec{r}_1[\alpha],\ \vec{r}_2[\alpha],\ 0\big)\ \times$ $\mathrm{PR}_{==}\big(\vec{r}_1[0]\cdots\vec{r}_1[\alpha-1],\ \vec{r}_2[0]\cdots\vec{r}_2[\alpha-1],\ x\big)$	λ	PR'_{λ} $\mathrm{PR}_{==}\big(\vec{r}_3[\alpha],\ \vec{r}_4[\alpha],\ \lambda\big)\ \times$ $\mathrm{PR}_{==}\big(\vec{r}_3[0]\cdots\vec{r}_3[\alpha-1],\ \vec{r}_4[0]\cdots\vec{r}_4[\alpha-1],\ \lambda b-x-\lambda\big)$
1	$\mathrm{PR}_1\ \times$ $\mathrm{PR}_{==}\big(\vec{r}_1[\alpha],\ \vec{r}_2[\alpha],\ 1\big)\ \times$ $\mathrm{PR}_{==}\big(\vec{r}_1[0]\cdots\vec{r}_1[\alpha-1],\ \vec{r}_2[0]\cdots\vec{r}_2[\alpha-1],\ x-1\big)$	$\lambda-1$	$\mathrm{PR}'_{\lambda-1}\ \times$ $\mathrm{PR}_{==}\big(\vec{r}_3[\alpha],\ \vec{r}_4[\alpha],\ \lambda-1\big)\ \times$ $\mathrm{PR}_{==}\big(\vec{r}_3[0]\cdots\vec{r}_3[\alpha-1],\ \vec{r}_4[0]\cdots\vec{r}_4[\alpha-1],\ \lambda b-x-\lambda+1\big)$
$\vdots$		$\vdots$	
x	$\mathrm{PR}_x\ \times$ $\mathrm{PR}_{==}\big(\vec{r}_1[\alpha],\ \vec{r}_2[\alpha],\ x\big)\ \times$ $\mathrm{PR}_{==}\big(\vec{r}_1[0]\cdots\vec{r}_1[\alpha-1],\ \vec{r}_2[0]\cdots\vec{r}_2[\alpha-1],\ 0\big)$	$\lambda-x$	$\mathrm{PR}'_{\lambda-x}\ \times$ $\mathrm{PR}_{==}\big(\vec{r}_3[\alpha],\ \vec{r}_4[\alpha],\ \lambda-x\big)\ \times$ $\mathrm{PR}_{==}\big(\vec{r}_3[0]\cdots\vec{r}_3[\alpha-1],\ \vec{r}_4[0]\cdots\vec{r}_4[\alpha-1],\ \lambda b-\lambda\big)$

Table 3: Possible pairs of configurations