Lawrence Berkeley National Laboratory
(University of California, University of California)

Year 2006 *Paper LBNL*›59949

Minimizing I/O Costs of Multi-Dimensional Queries with Bitmap Indices

Doron Rotem Kurt Stockinger

Kesheng Wu

Minimizing I/O Costs of Multi-Dimensional Queries with Bitmap Indices

Abstract

Bitmap indices have been widely used in scientific applications and commercial systems for processing complex, multi-dimensional queries where traditional tree-based indices would not work efficiently. A common approach for reducing the size of a bitmap index for high cardinality attributes is to group ranges of values of an attribute into bins and then build a bitmap for each bin rather than a bitmap for each value of the attribute. Binning reduces storage costs, however, results of queries based on bins often require additional filtering for discarding \it false positives, i.e., records in the result that do not satisfy the query constraints. This additional filtering, also known as "candidate checking," requires access to the base data on disk and involves significant I/O costs. This paper studies strategies for minimizing the I/O costs for "candidate checking" for multi-dimensional queries. This is done by determining the number of bins allocated for each dimension and then placing bin boundaries in optimal locations. Our algorithms use knowledge of data distribution and query workload. We derive several analytical results concerning optimal bin allocation for a probabilistic query model. Our experimental evaluation with real life data shows an average I/O cost improvement of at least a factor of 10 for multi-dimensional queries on datasets from two different applications. Our experiments also indicate that the speedup increases with the number of query dimensions.

Minimizing I/O Costs of Multi-Dimensional Queries with Bitmap Indices

Doron Rotem, Kurt Stockinger and Kesheng Wu
Computational Research Division
Lawrence Berkeley National Laboratory
University of California
1 Cyclotron Road, Berkeley, CA 94720, USA

Abstract

Bitmap indices have been widely used in scientific applications and commercial systems for processing complex, multi-dimensional queries where traditional tree-based indices would not work efficiently. A common approach for reducing the size of a bitmap index for high cardinality attributes is to group ranges of values of an attribute into bins and then build a bitmap for each bin rather than a bitmap for each value of the attribute. Binning reduces storage costs, however, results of queries based on bins often require additional filtering for discarding false positives, i.e., records in the result that do not satisfy the query constraints. This additional filtering, also known as candidate checking, requires access to the base data on disk and involves significant I/O costs.

This paper studies strategies for minimizing the I/O costs for candidate checking for multi-dimensional queries. This is done by determining the number of bins allocated for each dimension and then placing bin boundaries in optimal locations. Our algorithms use knowledge of data distribution and query workload. We derive several analytical results concerning optimal bin allocation for a probabilistic query model. Our experimental evaluation with real life data shows an average I/O cost improvement of at least a factor of 10 for multi-dimensional queries on datasets from two different applications. Our experiments also indicate that the speedup increases with the number of query dimensions.

1 Introduction

Modern data warehouse and scientific applications store large amounts of high-dimensional data. Due to the complexity and the size of these data sets, efficient query processing is vital for retrieving results in real time.

Bitmap indexing is a common technique for processing complex, multi-dimensional ad-hoc queries on read-only data. They have been introduced into several commercial DBMS products by database vendors including Red Brick Systems, Sybase, IBM and Oracle.

The basic bitmap index uses every distinct value of the indexed attribute as a key for each index, and generates one bitmap containing as many bits as the number of records in the dataset for each key [12]. The sizes of these basic bitmap indices are relatively small for low-cardinality attributes, such as "gender", "types of houses sold in the San Francisco Bay Area", or "car models produced by Ferrari." However, for high-cardinality attributes such as "temperature values in a supernova explosion", the index sizes may be too large to be of any practical use. In this case, bitmap indices are often designed with bins [17]. This bitmap index strategy partitions the attribute values into a number of ranges, called bins, and uses bitmap vectors to represent bins (attribute ranges) rather than distinct values. Although binning typically reduces storage costs for high-cardinality attributes, it may increase the access costs of queries that do not fall on exact bin boundaries. For this kind of queries the original data values associated with edge bins must be accessed, in order to check them against the query constraints.

In this paper we are focusing on aggregation queries that are common in data warehousing and scientific applications. These types of queries do not return result records but rather statistical information on the result set, e.g. compute the size of the result set. Figure 1 shows a small example of evaluating such queries with binned bitmap indices. In this example we assume that an attribute A has values between 0 and 50. The values of the attribute A are given in the second leftmost column. The range of possible values of A is partitioned into five sub-ranges (*bins*), namely [0,10], [11,20] etc., with a bin allocated to each sub-range. The values of the sub-ranges are called *bin boundaries*. In this example, the width of each bin is of the same size. A "1-bit" indicates the attribute value falls into the range, and "0-bit" otherwise.

Assume that we want to evaluate the query "Count the number of rows where $8 < A < 37$". The correct result

Record ID	Original Values	0-10	11-20	21-30	31-40	41-50
1	5	1	0	0	0	0
2	34	0	0	0	1	0
3	23	0	0	1	0	0
4	9	1	0	0	0	0
5	12	0	1	0	0	0
6	5	1	0	0	0	0
7	34	0	0	0	1	0
8	42	0	0	0	0	1
9	11	0	1	0	0	0
10	22	0	0	1	0	0
11	44	0	0	0	0	1
12	23	0	0	1	0	0
13	41	0	0	0	0	1
14	18	0	1	0	0	0
15	39	0	0	0	1	0

Edge bin
Internal bin

8 < A < 37

Figure 1. Two-sided range query $8 < A < 37$ on a bitmap index with binning.

should be 9. We know that all records that fall into *internal bins* (highlighted in light gray) are sure hits (*qualifying records*). These records are indicated by a "1-bit" and are calculated by performing a *Boolean OR operation* on all internal bins. On the other hand, records that fall into so-called *edge bins* (highlighted in dark gray) contain both qualifying and non-qualifying values. In order to prune the *false positives*, the original data values need to be checked against the query constraint. In particular, all records of the edge bins with a bit set to "1", need to be checked. Such a check may involve additional accesses to disk pages depending on how the attribute values are stored.

Given the query $8 < A < 37$, let us look at the candidate check for the left edge bin. The candidate records are the records with IDs 1, 4 and 6. The values of these records are 5, 9 and 5, respectively. The only qualifying record is record 4 that represents the value 9. The other two records do not fulfill the query constraint and do not qualify. As we can see from this small example, the cost of performing a *candidate check* on an edge bin is related to the number of "1-bits" in that bin. The larger the number of candidates that need to be checked, the higher the total query processing costs.

Our bitmap indexing software called *FastBit* [7] uses the binning strategy of this example. *FastBit* has been used for High-Energy Physics experiments over the last several years [20]. Recently we integrated our bitmap index technology directly into the ROOT analysis framework [16] that has a user community of some 10,000 scientists around the world. Based on the experience we gained with the integration of our software and the feedback from the user community, we identified the *candidate check cost* as the main bottleneck of query processing. Moreover, by studying the query workloads of these experiments, we identified certain patterns that could be helpful for designing more efficient bitmap indices that take into account both data distributions and query distributions. Given a fixed number of bins[1], the goal is to find the optimal bin boundaries such that the number of candidates and thus the query processing costs is minimized.

Before we outline the main contributions of this paper, we provide a brief taxonomy of the major work on optimizing bitmap indices based on query access patterns. We discuss further related work in Section 2.

1.1 Brief Taxonomy of Optimizing Bitmap Indices Based on Query Access Patterns

In Table 1, most of the previous results in this area are classified according to the problem dimensionality and the type of queries considered. Earlier work in the area attempted to adjust bin sizes dynamically as the data is updated ([21]). Most of the later work assumes read-only data and uses dynamic programming algorithms to achieve optimal bin boundary placement ([10],[15],[14]). In the multi-dimensional case, additional strategies for speeding up the queries were studied. These include sophisticated strategies for pruning the potential subset of candidates using Boolean operations [17] and dynamically reordering the scanning of the bitmap indices based on estimated attribute selectivities [15]. Some more discussion about related work can be found in Section 2.

1.2 Main Contributions of this Work

In our previous work we introduced *OptBin*, an algorithm for optimally determining the bin boundaries for bitmap indices based on data distribution statistics and query workloads for one-dimensional queries [14]. We also derived a model *OptBinMulti* for multi-dimensional queries and studied the behavior for sythnetic data [15]. In this paper, we extend our research in the following directions:

- We present a detailed performance analysis of *MultiOptBin* for multi-dimensional queries on two different data sets from applications that are used in production. The goal is to show that the algorithm not only works for synthetic data [15] but also for the often more complex case of real data. The results show that our binning strategy yields significant improvements over traditional binning methods.

- In the past we allocated the same number of bins for each attribute (*fixed bin allocation*). In this paper we

[1] For practical reasons the number of bins is often fixed. This guarantees that the size of the bitmap index is below a certain storage threshold.

	One- Dimensional	**Multi-dimensional**
Queries not considered		Dynamic bucket expansion and contraction [21]
Point Queries	Optimal binning: dynamic programming [10]	
One-sided range queries		Optimal evaluation strategies for fixed binning and fixed evaluation order [17]
Two-sided range queries	Optimal binning: dynamic programming using only query endpoints [14]	-General problem NP complete [15] -Optimal query evaluation reordering [15] -Bin allocation for probabilistic model (**current paper**) -Systematic and extensive experimental evaluation with real world application data (**current paper**)

Table 1. Taxonomy of results on optimal binning for bitmap indices.

introduce a *probabilistic query model* that optimizes the number of allocated bins (*variable bin allocation*). The idea is to use more bins for attributes that appear more frequently in queries and to use fewer bins for attributes that appear less frequently. We evaluate this novel algorithm for real data. The results show further performance improvements over the previous algorithm *MultiOptBin*.

The rest of the paper is organized as follows. In Section 2 we discuss related work on bitmap indices and binning strategies. In Section 3 some of our previous results on single and multi attribute optimal binning are presented. In Section 4 we present results for the optimal choice of bin allocations for a probabilistic query model. In Section 5 we evaluate our novel strategies for optimizing multi-dimensional queries based on data sets from two different applications. Finally, in Section 6 we present our conclusions and discuss some future work.

2 Related Work

Bitmap indices are used for speeding up complex, multi-dimensional queries for On-Line Analytical Processing and data warehouse [6] as well as for scientific applications [17]. The first commercial product to use the name bitmap index is Model 204 [12]. Improvements on this approach called *bit sliced-index* are discussed in [13].

In [4, 5] three bitmap encoding strategies are introduced: *equality, range* and *interval* encoding. Equality-encoded bitmap indices show the best performance for processing *equality queries* such as $velocity = 108$. *Range encoding* and *interval encoding* are optimized for *one-sided* and *two-sided range queries*, respectively. An example of a one-sided range query is $density < 10^9$. A two-sided range query, for instance, is $10^8 < density < 10^9$.

The authors of [22] represented attribute values in binary form that yields indices with only $\lceil log_2 |A| \rceil$ bitmaps, where $|A|$ is the attribute cardinality. The advantage of this encoding scheme is that the storage overhead is smaller than for *interval-encoding*. However, in most cases query processing is more efficient with *interval encoding* since in the worst case only two bitmaps need to be read whereas with binary encoding always all bitmaps have to be read.

Various bitmap compression schemes were studied in [2, 9]. The authors demonstrated that the scheme named Byte-aligned Bitmap Code (BBC) [3] shows the best overall performance characteristics. More recently a new compression scheme called Word-Aligned Hybrid (WAH) [19] was introduced. This compression algorithm significantly reduces the overall query processing time compared to BBC. The main reason for the efficiency of WAH is that it uses a much simpler compression algorithm.

The bitmap indices discussed so far encode each distinct attribute value as one bitmap. This technique is very efficient for data values with low attribute cardinalities (low number of distinct values). However, scientific applications are often based on data values with high attribute cardinalities. The work presented in [17] demonstrated that bitmap indices with binning can significantly speed up multi-dimensional queries on high-cardinality attributes.

A further bitmap index with binning called *range-based* bitmap indexing was introduced in [21]. The idea is to evenly distribute skewed attribute values onto various bins in order to achieve uniform search times for different queries. The authors demonstrated that the algorithm efficiently redistributes highly skewed data. However, performance results about query response times were not discussed.

The work in [10] focuses on one-dimensional point (equality) queries rather than range queries discussed in this paper. We extend the work of [10] by analyzing multi-

dimensional range queries.

Binning strategies could also be used to provide histogram information. The optimal construction of histograms for range queries that uses binning algorithms and is discussed in [11, 8]. The main difference between binning for histograms and our work is that for bitmap indices precise answers are required. Therefore the objective is to minimize disk access costs to edge bins. However, in the histogram case, some statistical techniques can be used to estimate errors without actual access to original data on disk.

3 Choosing Optimal Bin Boundaries

3.1 Single-Attribute Case

The *OptBin* problem for the single attribute case is defined as follows:

Assume a dataset D with one attribute in the range of [1,n], a set of range queries Q and a constraint k on the number of bins. Find a set of $k-1$ optimal bin boundaries such that the query processing costs, i.e. the costs of the candidate check, are minimized.

This problem is solved with a dynamic programming algorithm introduced in [14]. Its time complexity is $O(kr^2)$ where r is the number of distinct query endpoints of queries in Q and k is the constraint on the number of bins. As expected, the amount of cost savings (in terms of reduced I/O costs for candidate check) achieved by the algorithm increases with the degree of accuracy of our estimations of data and query distributions. Fortunately, as bitmaps are mainly used for read-only data, histograms representing data distribution information can be collected at a minimal cost during bitmap index construction. Our experience with scientific data also shows that query distribution can be collected effectively by analyzing workload traces and understanding the kind of phenomena the scientists are studying.

3.2 Multi-Attribute Case

The general *MultiOptBin* problem is defined as follows:

Given a multi-dimensional dataset D, a set of range queries Q and a constraint k on the total number of bins, find t integers $k_1, k_2, ..., k_t$ where $k = \sum_{i=1}^{t} k_i$ and locations for bin boundaries such that k_i bins are allocated for the bitmap index for attribute A_i and the total expected I/O cost of candidate check is minimized.

The multi-dimensional candidate check problem is much more complex then the single attribute case as several new factors are introduced. First, before we can deal with bin boundary placement, we need to decide how many bins must be allocated for each attribute. This in turn is dependent on several factors such as the likelihood of an attribute to appear in a query as well as its selectivity.

In general, the total cost of multi-dimensional candidate check is a weighted sum of candidate checks costs, $cost(A_i)$, for each attribute A_i appearing in the query. The weights depend on attribute selectivities as each candidate check results in pruning of the potential candidate subset. We only need to check candidates that survived all previous prunings.

The candidate check cost $cost(A_i)$ for each attribute A_i is a non-increasing function of k_i, the number of bins allocated to A_i. Unfortunately, in the general case the function is not known and depends on the data and query distribution. It is therefore not surprising that the exact optimal solution to *MultiOptBin* is a NP-hard problem as shown in the next theorem.

Theorem 1 *The MultiOptBin is NP-hard even if all queries in Q include a range for only one attribute and all s_i's (attribute selectivities) are equal.*

The proof of this theorem can be found in [15].

A closed-form solution to the multi-dimensional bin allocation problem can be computed if all $cost(A_i)$'s are differentiable functions under a probabilistic model of query distributions. In Section 4 we show how this solution is computed.

The strategy we use in our system is an approximation to the optimal solution and consists of the following stages. We first determine the number of bins that must be allocated to the bitmap index of each attribute using data and query statistics and applying the closed form solution mentioned above. Using this bin allocation, we proceed to compute optimal bin boundaries by applying the single attribute dynamic-programming algorithm explained in Section 3.1 separately on each attribute. In our experiments with real application data presented in Section 5 we show a significant speed-up over naive bin allocations using this strategy.

4 Probabilistic Query Model

4.1 The Model

As we showed in Section 3, the cost of candidate check is reduced when more bins are allocated to the bitmap index. The reason for this is that with more bins the likelihood of queries to fall on bin boundaries increases and the expected number of records per bin (including edge bins) decreases. Given a constraint on the total number of bins available to all attributes, the question is how many bins should be allocated to each attribute. In previous work [14, 15], the index for each attribute uses the same number of bins. However, it seems intuitive to allocate more bins to attributes which

are more likely to show up in queries as their respective indices will be used more often. In this section we develop a strategy for allocating bins to attributes based on some knowledge of the likelihood of these attributes to show up in queries and also taking into account the selectivities of the attributes.

Many works on multi-dimensional range queries (see for example [1]) describe the query set Q by a probabilistic model that assumes attribute independence. This means that the probability that an attribute A_i will show up in a query, denoted by p_i, depends on the attribute itself but is independent of the other attributes appearing in the query. For example, in the case of four attributes A_1, A_2, A_3 and A_4, the probability that a random query will specify only the attributes A_1 and A_3 is $p_1(1-p_2)p_3(1-p_4)$. More formally, for a query q, the probability, p_q, it is submitted to the system satisfies $p_q = \left[\prod_{A_i \in q} p_i\right]\left[\prod_{A_i \notin q} (1-p_i)\right]$ where the notation $A_i \in q$ ($A_i \notin q$) denotes that attribute A_i is specified (not specified) in the query q. As shown below, another factor that affects the bin allocation strategy is attribute selectivity. The selectivity of an attribute A_i, denoted by s_i, measures the expected number of records that satisfy the conditions on A_i over all queries of Q. We assume that the query evaluation is done in phases where in each phase the range condition on one of the attributes is evaluated using the appropriate bitmap index. In phase 1 all records are still candidates and in general in phase $i+1$ a condition on one attribute is evaluated for all records which are still candidates after phase i. This is illustrated in Figure 2 for 3 attributes where the initial number of records in the database is N and the number of candidates that are checked in each phase is equal to the size of the initial databases multiplied by the product of the attribute selectivities checked in all the previous phases. The analysis presented in this section quantifies this observation.

We assume that the cost of candidate check on attribute A_i, $\cos t(A_i)$, is a function $f(k_i)$ of the number of bins, k_i, allocated to it. We denote by $C(j, N)$ the expected cost of candidate check on j attributes in the order $A_j, A_{j-1}, .., A_1$ with k_i bins allocated to attribute A_i on a database of N records. It is also assumed that this cost shrinks linearly with the number of records in the database such that reducing the size of the database by a factor of s ($0 < s \leq 1$) reduces the cost by the same factor, i.e, $C(j, sN) = sC(j, N)$.

In the following analysis we simply use the notation $C(j)$ instead of $C(j, N)$ when the size of the database is known to be N.

Lemma 1: The candidate check cost satisfies the recursive equations

$$C(1) = f(k_1) \quad (1)$$

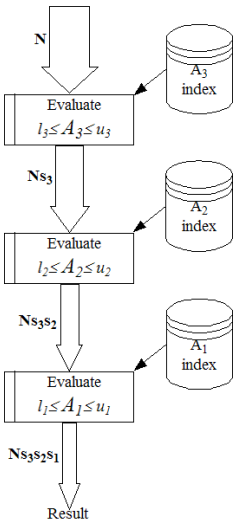


Figure 2. Query evaluation phases.

$$C(t) = p_t(f(k_t) + C(t-1)s_t) + (1-p_t)C(t-1) \quad (2)$$

Proof: The equation (1) simply denotes the cost of candidate checking a single attribute A_1. In equation (2), the first term corresponds to the probability of attribute A_t appearing in a random query. In this case we will search its bitmap index at a cost $f(k_t)$. We then pay the cost of $C(t-1)$ on searching the rest of the index on the $t-1$ attributes $A_{t-1}, A_{t-2}.., A_1$ with a database size that is a fraction s_t of the original one leading to a cost of $C(t-1)s_t$ by the linearity assumption of the cost function. The second term denotes the cost in case attribute A_t is not mentioned in the query; this occurs with probability $(1-p_t)$. Note that in this case there is no "shrinkage" in the candidate database and the cost is simply $C(t-1)$. □

Lemma 2:

$$C(2) = p_1 f(k_1)(1 - p_2(1-s_2)) + p_2 f(k_2) \quad (3)$$

$$C(t) = \sum_{i=1}^{t} p_i f(k_i) \prod_{j=i+1}^{t} (1 - p_j(1-s_j)) \quad (4)$$

Proof: Proof is by induction on t. For the base case of $t = 2$ we have bin allocations k_2 and k_1 to the two attributes A_2 and A_1, respectively. Assuming the evaluation starts with attribute A_2 we get directly from the definitions

$$C(2) = p_2(f(k_2) + p_1 s_2 f(k_1)) + (1-p_2)p_1 f(k_1) \quad (5)$$

Rearranging the right-hand side, we get equation (3). Equation (4) follows by assuming equation (4) holds for $C(t-1)$ by induction hypothesis and substituting it in equation (2). □

Theorem 2 *Given a query set Q on t attributes. We assume the order of candidate check evaluation is $A_t, A_{t-1}, ..., A_1$ where A_i has selectivity s_i and probability p_i and the candidate check cost function $f(k_i)$ has the derivative $f'(k_i)$*

The optimal bin allocation satisfies:

$$\frac{f'(k_i)}{f'(k_{i+1})} = \frac{p_{i+1}}{p_i(1-p_{i+1}(1-s_{i+1}))} \quad \text{for } 1 \leq i < t. \tag{6}$$

Proof: The result of the theorem follows by finding the minimum of $C(t)$ subject to the constraint $k = \sum_{i=1}^{t} k_i$ using Lagrange multiplier techniques. That involves solving the set of equations

$$\frac{\partial}{\partial k_i}\left[\sum_{i=1}^{t} p_i f(k_i) \prod_{j=i+1}^{t}(1-p_j(1-s_j)) + \lambda(\sum_{i=1} k_i - k)\right] = 0 \tag{7}$$

$$\text{for } i = 1, 2, .., t$$

$$k = \sum_{i=1}^{t} k_i \tag{8}$$

From equation 7 we get

$$p_i f'(k_i) \prod_{j=i+1}^{t}(1-p_j(1-s_j)) = \lambda \quad \text{for } 1 \leq i \leq t \tag{9}$$

It follows that

$$p_i f'(k_i) \prod_{j=i+1}^{t}(1-p_j(1-s_j))$$

$$= p_{i+1} f'(k_{i+1}) \prod_{j=i+2}^{t}(1-p_{j+1}(1-s_{j+1})) \quad \text{for } 1 \leq i < t \tag{10}$$

from which the theorem follows. □

In the special case that the cost function $f(k_i)$ of searching an attribute A_i satisfies $f(k_i) = O(1/k_i)$ equation (6) in the above theorem can be presented as:

$$\frac{k_{i+1}}{k_i} = \sqrt{\frac{p_{i+1}}{p_i(1-p_{i+1}(1-s_{i+1}))}} \quad \text{for } 1 \leq i < t \tag{11}$$

(12)

This kind of function behavior is intuitively reasonable for attributes with close to uniform data distribution where the cost of candidate checking for an attribute is inversely proportional to the number of bins allocated to that attribute.

To illustrate the above theorem, we show bin allocations of 1000 bins among four attributes with probabilities 0.1, 0.2, 0.7 and 0.8 (see Figure 3). For all attributes we assume the same selectivity s. We show the optimal bin allocation for s ranging between 0 and 1. For example, for selectivity s = 0.4, the attributes A_1 to A_4 get the following number of bins allocated 85, 129, 317 and 469 (see dashed line). Also note that with smaller values of s, the ratio between the number of bins allocated to A_4 and A_3 is larger. Further experimental results are given in Section 5.

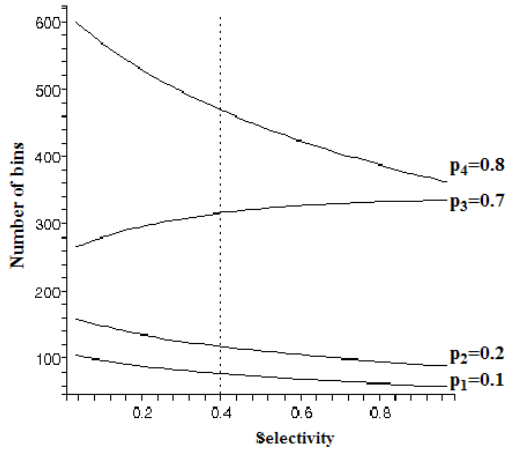


Figure 3. Optimal bin allocation as a function of candidate selectivity.

5 Experimental Results

In this section we evaluate the performance of multi-dimensional queries that are optimized with our novel bitmap index strategies. The performance measurements are based on two different data sets from applications that are used in production. This guarantees that we evaluate our strategy on large real data sets as opposed to simplified synthetic data sets.

The first data set contains network traffic data that was collected at Berkeley Lab in May 2005. The second data set is based on a supernova explosion from the Tera Scale Supernova Initiative [18]. The goal of our experiments is to compare our binning approach with one of the most commonly used binning strategies in production environments that is called *Equi-depth*. This binning strategy places the

bin boundaries in such a way that each bin has approximately the same number of entries. The advantage of equi-depth binning is that it reduces the worst-case query processing costs.

As we have pointed out in the introductory section, the bottleneck of bitmap indices with bins is the *candidate check*. In other words, the higher the number of candidates that need to be checked against the query constraint, the higher the query processing costs. Thus, in this section we use the term *query processing costs* as a synonym for *candidate check costs*.

5.1 Network Traffic Data

5.1.1 Data Characteristics

The network traffic data set we used for our experiments contains incoming and outgoing network traffic to and from Berkeley Lab. The data set includes attributes such as *DestinationIP*, *SourceIP*, *DestinationPort*, *SourcePort*, *SourceBytesPerPacket*, *DestinationBytesPerPacket*, *StartTime*, etc. The data set contains 10.2 million records and 8 attributes.

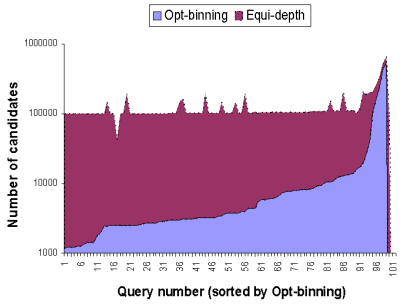
5.1.2 One-Dimensional Queries

For our experiments we generated 1,000 random uniformly distributed queries that cover the whole domain space for each attribute. For these 1000 queries we ran our optimization algorithm and calculated the optimal bin boundaries for 100 bins per attribute. Next, we built the bitmap indices accordingly. Figure 4 shows the query processing costs for the binning strategies *Opt-binning* and *Equi-depth* binning. The costs are expressed in terms of the number of candidates that need to be accessed in the candidate check. The graph shows the performance of 100 randomly sampled queries that are sorted according to the costs of *Opt-binning*. For all measured attributes, the processing costs of *Opt-binning* are, on average, a factor of 3 smaller compared with *Equi-depth*.

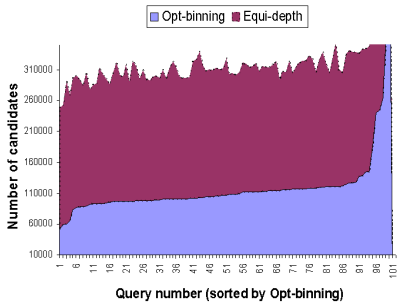
To show the effect of *Opt-binning* on the location of the bin boundaries, we plotted the boundaries of *Opt-binning* and compared them with *Equi-depth* binning (see Figure 5). For instance, for attribute *DestinationPort* in Figure 5 (a), the bin boundaries change significantly starting from bin 25. *Equi-depth* binning keeps the bin boundaries equally sized up to bin 60, whereas *Opt-binning* changes the bin boundaries already with bin 25 to reflect the query distributions.

5.1.3 Multi-Dimensional Queries

Next, we measured the query processing costs for multi-dimensional queries. Figure 6 shows the cost improvement factor of *Opt-binning* over *Equi-depth*. For 4-dimensional



(a) *DestinationPort*



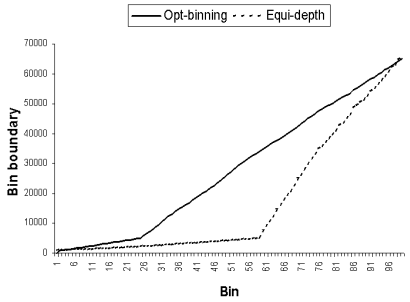
(b) *SourceBytesPerPacket*

Figure 4. Query processing costs for attribute *DestinationPort* and *SourceBytesPerPacket*. Note: The query processing costs are proportional to the area of the graph.

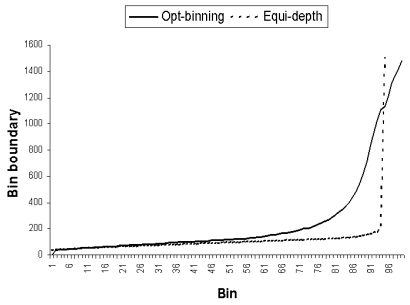
queries, the cost improvement of *Opt-binning* over *equi-depth binning* is about a factor of 9. We can also see that as the number of query dimensions increases, the cost improvement increases as well.

5.1.4 Probabilistic Queries

In the next experiment we evaluated our probabilistic query model for further reducing the query processing costs. In our previous experiments we used the same number of bins for each attribute. Now we calculate the optimal number of bins for each attribute depending on the probability of it being contained in a multi-dimensional query expression. This optimization strategy is motivated by the observations we made during the analysis of real query workloads [14]. In multi-dimensional queries the number of attributes per query often changes. Thus, different attributes often show different probabilities of being contained in a certain query expression. The key idea of calculating the optimal number of bins is to increase the number of bins for those attributes that have a higher probability of being contained in a set of queries. For attributes that have a lower probability, the



(a) *DestinationPort*



(b) *SourceBytesPerPacket*

Figure 5. Bin boundaries for attribute (a)*DestinationPort* and (b) *SourceBytesPerPacket*.

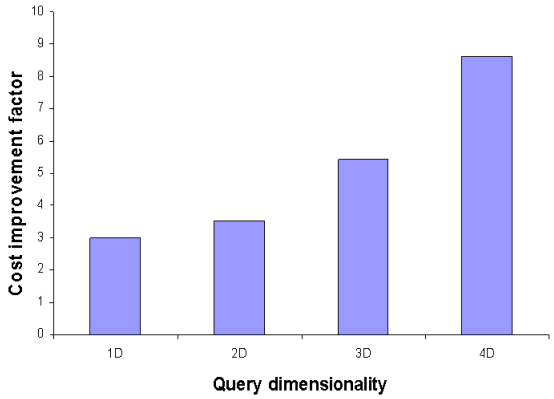


Figure 6. Cost improvement factor of *Opt-binning* over *equi-depth binning* for multi-dimensional queries.

number of bins is reduced.

In order to study this strategy we generated 5000 queries with up to four dimensions. For each attribute we assumed a different probability of being contained in the query expression. Given these probabilities, the optimal number of bins is calculated according to Equation 11. The selectivity for all attributes is 0.3. Table 2 shows the probabilities and the respective optimal number of bins for the four attributes of our query workload.

Attribute	Probability	Opt. #bins
SourcePort	0.8	199
DestinationPort	0.7	123
StartTime	0.2	47
SrcBytesPerPacket	0.1	31

Table 2. Probabilities for attributes to be contained in a query expression along with the respective optimal number of bins.

Next, we computed the optimal bin boundaries where each attribute has a different number of bins (as shown in Table 2). In addition, we computed the optimal bin boundaries where each attribute has 100 bins (as we did in our previous experiments). The experiments show that calculating the optimal number of bins reduces the I/O costs by another 30% compared with the optimal strategy where each attribute has the same number of bins.

5.2 Astrophysics Data

5.2.1 Data Characteristics

The astrophysics data set is one order of magnitude larger than the network traffic data set. The challenge was to measure how *Opt-binning* performs for this large data set of 110 million records and 6 attributes such as *x-velocity*, *y-velocity*, *z-velocity*, *density*, *pressure* and *entropy*.

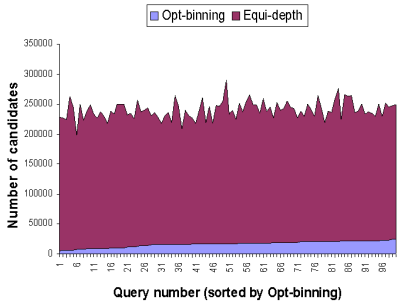
Since the number of records of this data set is much larger than the previous one, we increased the number of bins to 1000. We also increased the number of queries to 5000. In addition, we made an important change to the query distribution according to the following observation. In our previous work we studied the work load of real queries from the Sloan Digital Sky Survey [14] and High-Energy Physics [16]. The main observation is that the query distributions are often not uniform but centered around either the peaks of the data distribution or around the tails. Thus, we experimented with different query distributions that follow these basic rules.

5.2.2 One-Dimensional Queries

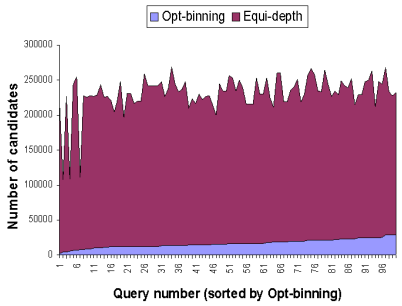
The distribution of attribute *x-velocity* follows a Gauss distribution with 3 peaks centered around 0 (due to space limitations we do not provide a figure). For this attribute we

generated 5000 queries that are centered around 0 and calculated the optimal bin boundaries for 1000 bins accordingly. Figure 7 (a) shows that the query processing costs for *Opt-binning* are about a factor of 13 lower than the costs for *Equi-depth*.

Next, we generated 5000 queries for attribute *y-velocity* that has a similar distribution as attribute *x-velocity*. However, rather than centering the query distribution around 0, we produced a right-skewed distribution where most of the queries hit the right side (tail) of the data. Also for this kind of query workload, *Opt-binning* reduces the query processing costs by more than a factor of 10 compared with *Equi-depth* binning (see Figure 7 (b)).



(a) *x-velocity*



(b) *y-velocity*

Figure 7. Query processing costs for attribute *x-velocity* and *y-velocity*.

5.2.3 Multi-Dimensional Queries

Finally, we measured the query processing costs for multi-dimensional queries on two and three attributes. Figure 8 shows the cost improvement factor of *Opt-binning* over *Equi-depth* which is between 11.5 and 17.5. Similar to the network traffic queries, we can see that as the number of query dimensions increases, the cost improvement of *Opt-binning* increases even more significantly.

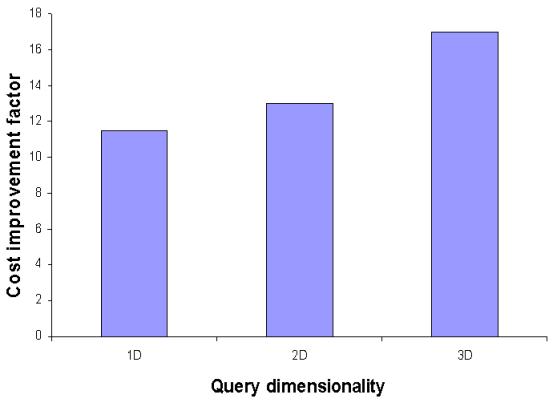


Figure 8. Cost improvement factor of *Opt-binning* over *Equi-depth* for multi-dimensional queries.

5.2.4 Probabilistic Queries

In our last set of experiments we analyzed the performance of our probabilistic query model. We generated 5000 multi-dimensional queries with randomly chosen query ranges covering the whole domain space. For each attribute we assumed a different probability of being contained in the query expression. We assumed an overall bin constraint of 600 bins. For the set of generated queries we calculated the average attribute selectivities. We calculated the optimal bin locations such that each of the six attributes gets 100 bins allocated. Next, we calculated the optimal number of bins by taking into account the attribute probabilities and selectivities using Equation 11. Table 3 shows the attribute probabilities and selectivities as well as the optimal number of bins per attribute. Our experiments showed a 38% improvement of the probabilistic optimization over the strategy *Opt-binning* where each attribute has the same number of bins.

Attribute	Probability	Selectivity	Opt. #bins
x_velocity	0.8	0.39	216
y_velocity	0.8	0.39	156
z_velocity	0.8	0.42	113
pressure	0.3	0.06	50
density	0.2	0.09	39
entropy	0.2	0.07	31

Table 3. Probabilities for attributes to be contained in a query expression along with the respective optimal number of bins.

6 Conclusions

In this paper we presented an algorithm for optimizing the costs of multi-dimensional queries with bitmap indices. Our approach is based on the following two steps: a) Given a set of data distributions and a set of query distributions, find bin allocation for each attribute. b) Find an optimal placement for the bin boundaries such that the number of candidates that need to be checked against the query constraints is minimized.

We performed both analytical and experimental studies to evaluate the efficiency of our strategy. Our experiments were based on two data sets from applications that are used in production. For the analyzed data sets we achieved a performance improvement in the range of 3 to 17. The results show that as the number of query dimensions increases, the efficiency of our algorithm increases as well.

Future work involves testing our optimization techniques against other binning strategies. A further direction of future work is to design bin allocation algorithms for probabilistic queries with attribute dependencies.

7 Acknowledgments

This work was supported by the Director, Office of Science, Office of Advanced Scientific Computing Research, of the U.S. Department of Energy under Contract No. DE-AC03-76SF00098.

References

[1] A. V. Aho and J. D. Ullman. Optimal Partial-Match Retrieval When Fields Are Independently Specified. *ACM Trans. Database Syst.*, 4(2):168–179, 1979.

[2] S. Amer-Yahia and T. Johnson. Optimizing Queries on Compressed Bitmaps. In *International Conference on Very Large Data Bases (VLDB)*, Cairo, Egypt, September 2000. Morgan Kaufmann.

[3] G. Antoshenkov. Byte-aligned Bitmap Compression. Technical report, Oracle Corp., 1994. U.S. Patent number 5,363,098.

[4] C.-Y. Chan and Y. E. Ioannidis. Bitmap Index Design and Evaluation. In *SIGMOD*, Seattle, Washington, USA, June 1998. ACM Press.

[5] C. Y. Chan and Y. E. Ioannidis. An Efficient Bitmap Encoding Scheme for Selection Queries. In *SIGMOD*, Philadelphia, Pennsylvania, USA, June 1999. ACM Press.

[6] S. Chaudhuri and U. Dayal. An Overview of Data wharehousing and OLAP Technology. *ACM SIGMOD Record*, 26(1):65–74, March 1997.

[7] FastBit - An Efficient Compressed Bitmap Index Technology. http://sdm.lbl.gov/fastbit/.

[8] S. Guha, N. Koudas, and D. Srivastava. Fast Algorithms For Hierarchical Range Histogram Construction. In *PODS 2002*, Madison, Wisconsin, USA, June 2002. ACM Press.

[9] T. Johnson. Performance Measurements of Compressed Bitmap Indices. In *International Conference on Very Large Data Bases (VLDB)*, Edinburgh, Scotland, September 1999. Morgan Kaufmann.

[10] N. Koudas. Space Efficient Bitmap Indexing. In *International Conference on Information and Knowledge Management (CIKM)*, McLean, Virginia, USA, November 2000. ACM Press.

[11] N. Koudas, S. Muthukrishnan, and D. Srivastava. Optimal Histograms for Hierarchical Range Queries. In *PODS*, Dallas, Texas, USA, 2000. ACM Press.

[12] P. O'Neil. Model 204 Architecture and Performance. In *2nd International Workshop in High Performance Transaction Systems*, Asilomar, California, USA, 1987. Springer-Verlag.

[13] P. O'Neil and D. Quass. Improved Query Performance with Variant Indexes. In *Proceedings International Conference on Management of Data (SIGMOD)*, Tucson, Arizona, USA, May 1997. ACM Press.

[14] D. Rotem, K. Stockinger, and K. Wu. Optimizing Candidate Check Costs for Bitmap Indices. In *International Conference on Information and Knowledge Management (CIKM)*, Bremen, Germany, November 2005. ACM Press.

[15] D. Rotem, K. Stockinger, and K. Wu. Optimizing I/O Costs of Multi-Dimensional Queries using Bitmap Indices. In *International Conference on Database and Expert Systems Applications (DEXA)*, Kopenhagen, Denmark, August 2005. Springer-Verlag.

[16] K. Stockinger, K. Wu, R. Brun, and P. Canal. Bitmap Indices for Fast End-User Physics Analysis in ROOT. In *Nuclear Instruments and Methods in Physics Research*. Elsevier. to appear.

[17] K. Stockinger, K. Wu, and A. Shoshani. Evaluation Strategies for Bitmap Indices with Binning. In *International Conference on Database and Expert Systems Applications (DEXA)*, Zaragoza, Spain, September 2004. Springer-Verlag.

[18] TeraScale Supernova Initiative. http://www.phy.ornl.gov/tsi/.

[19] K. Wu, E. J. Otoo, and A. Shoshani. On the Performance of Bitmap Indices for High Cardinality Attributes. In *International Conference on Very Large Data Bases (VLDB)*, Toronto, Canada, September 2004. Morgan Kaufmann.

[20] K. Wu, W.-M. Zhang, V. Perevoztchikov, J. Lauret, and A. Shoshani. The Grid Collector: Using an Event Catalog to Speedup User Analysis in Distributed Environment. In *Computing in High Energy and Nuclear Physics (CHEP) 2004*, Interlaken, Switzerland, September 2004.

[21] K.-L. Wu and P.S. Yu. Range-Based Bitmap Indexing for High Cardinality Attributes with Skew. In *COMPSAC*, pages 61–67, 1998.

[22] M.-C. Wu and A. P. Buchmann. Encoded Bitmap Indexing for Data Warehouses. In *International Conference on Data Engineering (ICDE)*, Orlando, Florida, USA, February 1998. IEEE Computer Society Press.