An Entropy Maximizing Geohash for Distributed Spatiotemporal Database Indexing

Taylor B. Arnold
AT&T Labs Research
33 Thomas Street
New York, NY 10007
taylor@research.att.com

Abstract

We present a modification of the standard geohash algorithm based on maximum entropy encoding in which the data volume is approximately constant for a given hash prefix length. Distributed spatiotemporal databases, which typically require interleaving spatial and temporal elements into a single key, reap large benefits from a balanced geohash by creating a consistent ratio between spatial and temporal precision even across areas of varying data density. This property is also useful for indexing purely spatial datasets, where the load distribution of large range scans is an imporant aspect of query performance. We apply our algorithm to data generated proportional to population as given by census block population counts provided from the US Census Bureau.

1 Introduction

There has been a rapid increase in petabyte-scale data sources across a range of industry, government, and academic applications. Many of these sources include data tagged with both geospatial and temporal attributes. Website HTTP request logs, along with geocoded IP addresses, are a common example where even modestly sized sites can quickly produce large-scale datasets. Inexensive embedded GPS devices are another common source, and are typically found in everything from automobiles to cellphones. Sensor data from Radio-frequency identification (RFID) devices, which have been readily adapted to monitor complex supply-chain management operations, amongst other applications, also generate large datasets with both location and time-based features.

General purpose distributed filesystems such as the Google Filesystem [9] and the Hadoop Filesystem [15], provide adequate support for the raw storage of geospatial data sources. These alone however only provide minimal support for querying file chunks. Analysis, aggregations, and filtering often require processing through a large subset of the raw data. Database functionality is provided by additional software, often modelled off of Google's BigTable [3] design. Data are stored lexicographically by a primary key; queries take the form of a scan over a range of contiguous primary keys along with optional, additional query filters. Popular open source implementations of this form include Amazon's DynamoDB [5], Apache Accumulo [8], Apache HBase [17], and Apache Cassandra [12].

For geospatial data sources which are typically queried on an entity–by–entity basis, the standard distributed database applications are straightforward to implement. A uniquely identifying serial number for each entity can be used as the primary key, with spatial and temporal filtering accomplished with a combination of in-database filters; a single entity will typically constitute only a negligable proportion of the overall data volume. This application model for example would satisfy the needs of a user-facing supply chain system in which end-users search for the current location of an expected delivery. Unfortunately, more complex spatial queries do not fit neatly into the dis-

tributed database design. A single primary key is natively one-dimensional, whereas geospatial data is typically two-dimensional and, with the addition of a temporal component, generally no less than three. A common method for alleviating the dimensionality problem of having a single primary index is through the use of a space-filling curves such as Peano, Gosper, and Dragon curves. A geohash is a particular algorithm which uses a space-filling curve for encoding latitude and longitude tuples as a single string. It has had fairly wide adoption for this purpose; for example, it has been included as a core functionality in the popular distributed document store MongoDB [4].

The geohash concept adapts well for purely geospatial distributed datasets, allowing for querying spatial regions of vastly different scales through the use of a single index; larger regions specify a longer range of geohashes whereas smaller regions specify a shorter range. A temporal component can be incorporated into the space-filling curve underlying a standard geohash, however this requires an unavoidable, fixed database design decision as to the appropriate scaling factor between space and time. For instance: should a square kilometre bucket in space coorispond to a month, day, or hour bucket in time? The decision has important performance implications as spatial queries take the form of a collection of range queries of primary keys. If the buckets are not appropriately sized queries will either include many false positives (data returned outside of the area of interest) or require stitching together a large number of small scans. Either of these can cause significant performance deterioration, due to additional I/O time in the case of false positives and additional overhead in running a large number of small jobs.

In many cases the question of how to balance space and time into a single key is complicated by the fact that data are distributed very unevenly over the spatial dimensions. Sources which are generated roughly proportional to population density will produce 4 orders of magnitude or more in urban environments compared to rural ones. In these common cases, it is often more natural to scale the spatial buckets to be proprotional to the data volume. In this case, we instead only need to decide: should a bucket with 1 million data points coorispond to month, day, or hour bucket in time? This question will typically have a more natural answer for a given database system than the previous one, particularly when the query results are used as inputs to complex algorithms such as recommender systems or anomaly detection routines.

In order to implement a data balanced alternative to the standard space filling curves, we present a novel entropy maximizing geohash. Our method is necessarily model based and learned from a small subset of spatial data. It is robust both to random noise and model drift. It can implemented with only a modest increase in computational complexity and is generic enough to be implemented as-is in conjuction with several recently proposed geohash based schemes for storing large geospatial–temporal datasets. As a side effect of the entropy based encoding it has also reduces the size of the raw database files; this has been observed even after applying aggressive compression techniques.

The remainder of this article is organized as follows: Section 2 provides a specific formulation of the problems we are trying to solve and the deficiencies of addressing them using a the standard geohash. Section 3 provides a brief literature review of geospatial database techniques with a particular focus on how these can be used in conjuction with our methods. In Section 4 we give a thorough mathematical description of the standard geohash algorithm and present the formulation for entropy maximizing geohash. In particular, a theoretical result establishes an upper bound on how well the maximized geohash can be learnt from a sample of spatial data. Section 5 applies the entropic geohash to data from the US Census Bureau, empirically demonstrating how robust the method is to model drift and Section 6 provides avenues for future extensions.

2 Problem Formulation

Consider partitioning an area of space into equally sized square boxes. If these regions were given a unique identifier, a basic geospatial-temporal index can be constructed by concatinating the box id with a timestamp. Querying a region in spacetime, assuming that the time dimension of the query is small compared to the time window of the database, would be done by calculating a minimal covering of the spatial region with the square boxes. Then a range query could be placing for each box id between the desired start and end timestamps. Several inefficencies may result from this method:

1. If the boxes are too small relative the query area, the number of queries (each of which will lead to increased overhead) may be very large. While these can be placed in parallel, there will eventually be an I/O bottleneck on the machines storing the database when the number of parallel queries is large.

2. If the query area is smaller than the size of a box, the resultant queries will have a large number of false positives: points returned which are not in the desired query region. This leads to significant increases in both computational and I/O costs.

3. Even when the number of covering boxes is neither too small nor too large, the parallel execution of the queries over each box will not necessarily be balanced. The speedup will be sub-linear, with the query time driven by the densest box which commonly has an order of magnitude more data than the average data volume per block.

Issues (1) and (2) are dual to one another, and if all queries have similarly sized spatial regions the database can be constructed to use boxes with an area optimized for these queries. However, in the typical case that queries of varying sizes must be supported, any choice will run into one of these problems. Regardless, the load balancing associated with (3) continues to be a problem for any dataset with significant spatial bias.

As a solution to these problems, we propose a method for partitioning space as regions with equal data volumes rather than boxes with equally sized volumes. This is done by modifying the standard geohash algorithm without sacrificing any main benefits of the geohash algorithm or substantially increasing the computation burden of encoding or decoding input data points.

3 Related Work

There is a long history of work on data structures for optimizing geospatial and geospatial-temporal queries such as k-nearest neighbors and spatial joins. Common examples include R-trees [10], R^*-trees [2], and quadtrees [14]. These algorithms often offer superior performance to simple space filling curves by replicating the true dimensionality of the problem by an abstraction of trees structures as linked lists. Unfortunately linked lists do not fit into the primary key design scheme of the distributed databases mentioned in the previous section, making these more sophisticated algorithms out of reach for productionalized systems.

Some research has been done towards producing a distributed database which would be able to natively implement the geospatial temporal database structures which have been successful on single machine scale applications. These include VegaGiStore [20], VegaCache [18] and VegaCI [19], [21]. However at this time these remain merely theoretical as no distribution of any of these has even been publicly distributed, let alone developed to the maturity needed for a production system.

The simplier approach of adapting geohashes by interleavings spatial and temporal dimensions has also been explored. Given that these can be quickly implemented on top of existing industry standard databases this is the approach we have choosen to augment with our work. Fox et al. developed a generic approach for blending these two dimensions, along with methods for extending to lines and polygons and for producing fast queries in Apache Accumulo [7]. The HGrid data model is a similar application which takes advantage of the specific secondary indicies and related optimizations present in HBase [11]. Both rely on specifying the relative resolution of the spatial and temporal dimensions, and are able to use our entropy maximizing geohash with no major changes.

Finally, we note that the desire for a data balanced design which motivated our work is a natural parallel to the problems associated with efficent query design in the related tree-based work. Red-black trees [1] and the Day-Stout-Warren algorithm for binary trees [16] are important examples of load balancing used in relation to tree-based data structures. In this context, the entropy maximizing geohash can be thought of as the curve filling analogue to balanced trees.

4 Entropy Balanced Geohash

4.1 A Formulation of the Standard Geohash Encoding

As mentioned, a geohash is a simple scheme for mapping two-dimensional coordinates into a hierarchical, one-dimensional encoding. It is explained in several other sources, but we re-construct it here in a format which will be most conducive to generalizations. The first step is to map latitude and longitude coordinates in a standard unit square; this is done by the following linear mapping:

$$x = \frac{\text{lon} + 180}{360} \tag{1}$$

$$y = \frac{\text{lat} + 90}{180} \tag{2}$$

This choice is by convention, and any other method for mapping coordinantes into the unit square is equally viable.

The x and y coordinates need to be expressed in as a binary decimals. Formally, we define the $x_i \in \{0, 1\}$ and $y_i \in \{0, 1\}$ (with the restriction that neither is allowed to have an infinite trailing tail with all '1's, for uniqueness) such that

$$x = \sum_{i=1}^{\infty} \frac{x_i}{2^i}, \tag{3}$$

$$y = \sum_{i=1}^{\infty} \frac{y_i}{2^i}. \tag{4}$$

A geohash representation of (x, y) is constructed by interleaving these binary digits. The q-bit geohash $g_q(\cdot, \cdot)$ can symbolically be defined as

$$g_q(x, y) := \sum_{i=1}^{\lceil q/2 \rceil} \frac{x_i}{2^{2i-1}} + \sum_{i=1}^{\lfloor q/2 \rfloor} \frac{y_i}{2^{2i}}. \tag{5}$$

It is fairly easy to show that the geohash function is monotone increasing in q, with the growth strictly bounded by 2^{-q}, so that

$$0 \leq g_{q+m}(x, y) - g_q(x, y) < \frac{1}{2^q} \tag{6}$$

For all m greater than zero.

4.2 Entropy

A geohash is typically used as an index in the storing and querying of large spatial processes. A simple theoretical model for a stream of spatial data can be constructed by assuming that each observation is an independent identically distributed random variable $\mathfrak{F}$ from some distribution over space. Borrowing a concept from information theory, we can define the entropy of a geohash over a given spatial distribution by the equation

$$H(g_q) := -1 \sum_{v \in \mathcal{R}(g_q)} \mathbb{P}\left[g_q(\mathfrak{F}) = v\right] \cdot \log_2 \left\{\mathbb{P}\left[g_q(\mathfrak{F}) = v\right]\right\} \tag{7}$$

Where $\mathcal{R}(g_q)$ is the range of the q-bit geohash. It is a standard result that the entropy of a discrete distribution is maximized by the uniform distribution. Therefore we can use the entropy per bit as a proxy for how balanced a geohash is for a given distribution of spatial data.

4.3 The Generalized Geohash

As the q-bit geohash function is bounded and monotonic in q, we can define the infinite precision geohash, which we denote as simply $g(\cdot, \cdot)$, to be the limit

$$\lim_{q \to \infty} g_q(x, y) := g(x, y). \tag{8}$$

With this continuous format, one can see that if we compose g with an appropriate new function h, the composition $h \circ g(x, y)$ can be thought of as a rescaled version of the traditional geohash. To be precise, we would like a function h to have the following properties:

$$h : [0, 1] \to [0, 1], \tag{9}$$
$$h(0) = 0, \tag{10}$$
$$h(1) = 1, \tag{11}$$
$$x < y \iff h(x) < h(y). \tag{12}$$

Note that Equation 12 implies that h is also continuous. From here, we can define the analogue to a q-bit geohash by truncating the binary representation of $h(z) = w$,

$$h(z) = \sum_{i=1}^{\infty} \frac{w_i}{2^i} \tag{13}$$

To the its first q-bits

$$h_q(z) := \sum_{i=1}^{q} \frac{w_i}{2^i}. \tag{14}$$

In the remainder of this paper, we refer to $h_q \circ g(x, y)$ as a *generalized geohash*.

4.4 The Empirical Entropic Geohash

We have introduced the concept of a generalized geohash in order to construct a spatial encoding scheme which better optimizes the entropy as defined in Equation 7. Assume $\{z_i\}_{i=0}^{N}$ is a set of independent samples from realizations of the random variable $\mathfrak{F}$. The empirical cumulative distribution function G of the standard geohash function $g(\cdot, \cdot)$ is given by

$$G(t) := \frac{1}{N} \cdot \sum_{i=0}^{N} 1_{g(z_i) \leq t},\ t \in [0, 1]. \tag{15}$$

From Equation 15 we can define the *entropy maximizing geohash* function b (balanced), assuming that every point z_i has a geohash $g(z_i)$ strictly between 0 and 1, to be

$$b^q(t) := \begin{cases} 0 & \text{if } t = 0 \\ 1 & \text{if } t = 1 \\ \frac{N}{N+2} \cdot G^{-1}(t) & \text{if } \exists\, i \in \mathbb{Z} \text{ s.t. } t = i/2^q \\ \text{linear interpolation of the above points} & \text{else} \end{cases} \tag{16}$$

The balanced geohash is essentialy the inverse of the empirical function G, with some minor variations to satisfy Equations 9-12. If the points $\{z_i\}$ are unique, and N is sufficently large, the q-bit analogue b_q^q of Equation 16 will have an entropy $H(b_q^q)$ of approximately equal to q.

More formally, we can prove the following bound on the entropy of the balanced geohash:

Theorem 1. *Let b_q^q be the entropy balanced geohash estimated from a sample of N unique data points. Then, with probability at least $1 - 2e^{-0.49 \cdot 2^{-2q} N \cdot [1-A]^2}$ the entropy $H(b_q^q)$ is bounded by the following simultaneously for all values $A \in [0, 1]$:*

$$H(b_q^q) \geq q \cdot \frac{N}{N+2} \cdot A \tag{17}$$

Proof. Let $F(\cdot)$ be the true cumulative distribution function of the variable $\mathfrak{F}$, and $F_N(\cdot)$ be the empirical cumulative distribution function from a sample of N independent observations. Setting $\epsilon = [1 - A] \cdot \frac{N}{N+2} \cdot 2^{-(q+1)}$, the Dvoretzky-Kiefer-Wolfowitz inequality states [6] that the following holds for all values $A \in [0, 1]$ with probability $1 - e^{-2N\epsilon}$:

$$|F(x) - F_n(x)| \leq [1 - A] \cdot \frac{N}{N+2} \cdot 2^{-(q+1)} \tag{18}$$

Therefore, the empirical entropy should be bounded as follows:

$$H(b_q^q) \geq -1 \cdot \sum_{i=0}^{2^q-1} \left[F(i/2^q) - F((i+1)/2^q)\right] \times \log_2 \left[F(i/2^q) - F((i+1)/2^q)\right] \tag{19}$$

$$\geq -1 \cdot \sum_{i=0}^{2^q-1} \left[F_n(i/2^q) - F_n((i+1)/2^q) - 2\epsilon\right] \times \log_2 \left[F_n(i/2^q) - F_n((i+1)/2^q) - 2\epsilon\right] \tag{20}$$

$$= -2^q \cdot \left[\frac{N}{N+2} \cdot \frac{1}{2^q} - 2\epsilon\right] \times \log_2 \left[\frac{N}{N+2} \cdot \frac{1}{2^q} - 2\epsilon\right] \tag{21}$$

$$= -1 \cdot \frac{N}{N+2} A \times \log_2 \left[\frac{N}{N+2} \cdot \frac{1}{2^q} \cdot A\right] \tag{22}$$

$$\geq q \cdot \frac{N}{N+2} A \tag{23}$$

Which, plugging in the appropriate ϵ into the probablity bound, yields the desired result. □

Plugging in $A = 2/3$, the bound in Theorem 1 holds with probability greater than 0.99 for a 5-bit balanced geohash when N is at least $1e5$ and for a 10-bit geohash when N is at least $1e8$. We will see in the our empirical examples that the rate of convergence of the entropy to q is typically much faster.

5 Census Data Example

Many large geospatial datasets are distributed roughly proportional to population. Log files geocoded by ip addresses for instance will tend to have more activity in dense urban areas and less in rural regions. We use aggregate count data from the US Census data as a natural example for exploring the balanced geohash because these counts are also proportional to population (exactly equal, in fact). The Census Bureau distributes decennial aggregate counts across thousands of attributes at various levels of grainularity. The smallest region with un-sampled counts are census blocks, small geographic regions containing the residences of typically somewhere between 600 and 1300 people. There were over 11 million such regions in the 2010 Census. We use census blocks here in order to have the maximum level of geospatial grainularity available. Due to the smaller aggregation size, only a small subset of summary statistics are provided for census blocks. The total population will be used to construct the balanced geohash, with the total number of households renting their primary residence (contrasted with owning) being used to show the relative robustness of the balanced hash.

To help visualize the balanced geohash, the entropy balanced hash up to 3,4,5, and 6-bits were calculated from the population data in the state of Ohio. In Figure 1 the geohash 'buckets', all points in space with a common prefix, are illustrated over a map of the state. Technically the buckets extend over the entire globe, but are truncated here to increase readability of the graphic. Notice that each region in the q-bit bucket is bifurcated into two regions in the coorisponding $q+1$-bit bucket. Also, as is common, the smaller set of 3-bit buckets all have roughly the same area. However, the 6-bit buckets differ vastly in size such as the small area of region 21 and large area of region 51. As expected, the buckets tend to clump up around the major cities in Ohio, such as the 6-bit buckets 20-22, 28, 29 around Columbus.

Continuing with the census example in Ohio, Figure 2 plots the average entropy for a standard geohash and several balanced geohashes for census block population and rental household data. The rental household data uses a geohash balanced on population, and illustrates the degree to which our method is robust to changes in the underlying data model. In both cases, as expected, the balanced geohashes have higher average entropies the when more bits are used for balancing. There are significantly diminishing returns of increasing q, however, with the improvement from 2 to 3 bits being larger than the jump from 5 to 20. The entropy from the rental unit-based data is generally quite similar to the population-based data, indicating that the balanced hash is fairly robust to moderate changes in the generating model. Rental units are correlated with population, but do

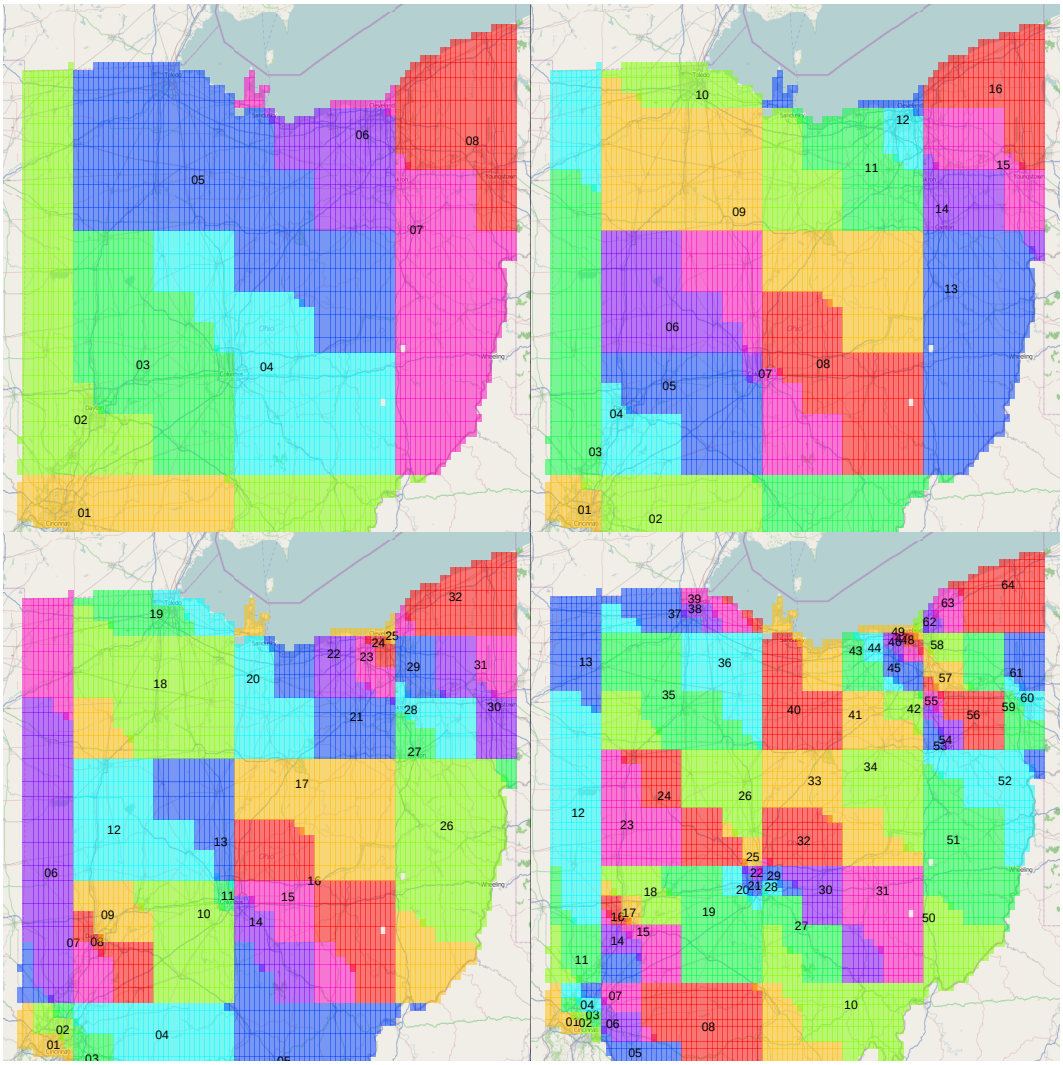

Figure 1: Example entropy balanced geohash buckets (those which have a common prefix) of 3,4,5, and 6-bits learned from population data in the state of Ohio. Numbers are sequentially ordered from the smallest to the largest geohash buckets, and the buckets were truncated to the state boundaries to increase readability even though they theoretically extend outside as well.

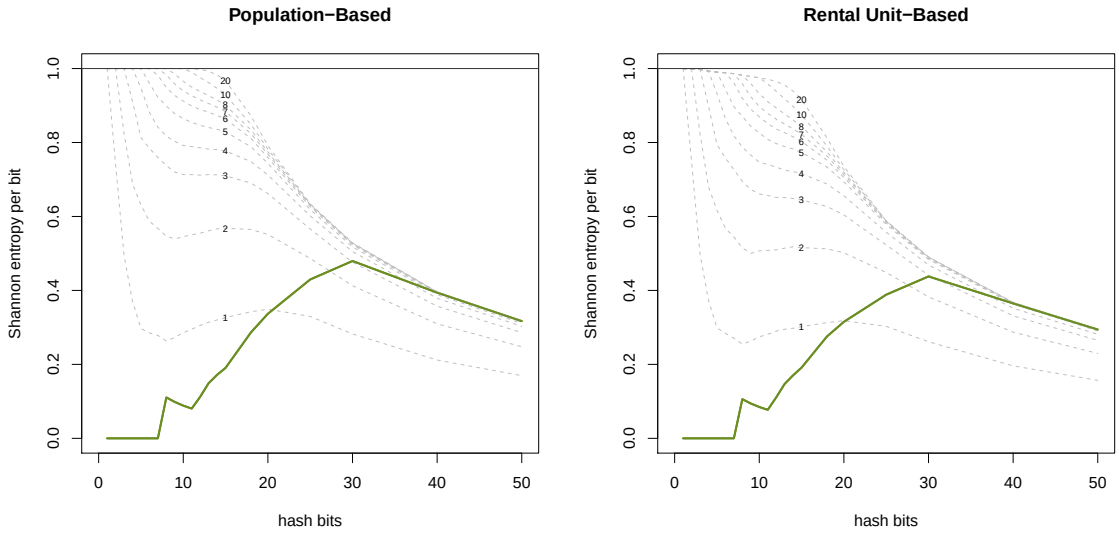

Figure 2: Average entropy for population and rental household data in Ohio. The green line is the standard geohash; dashed lines are balanced geohashes, and the overlaid numbers give the number of bits to which is is balanced. The rental unit balanced geohashes are balanced on population data rather than the rental data.

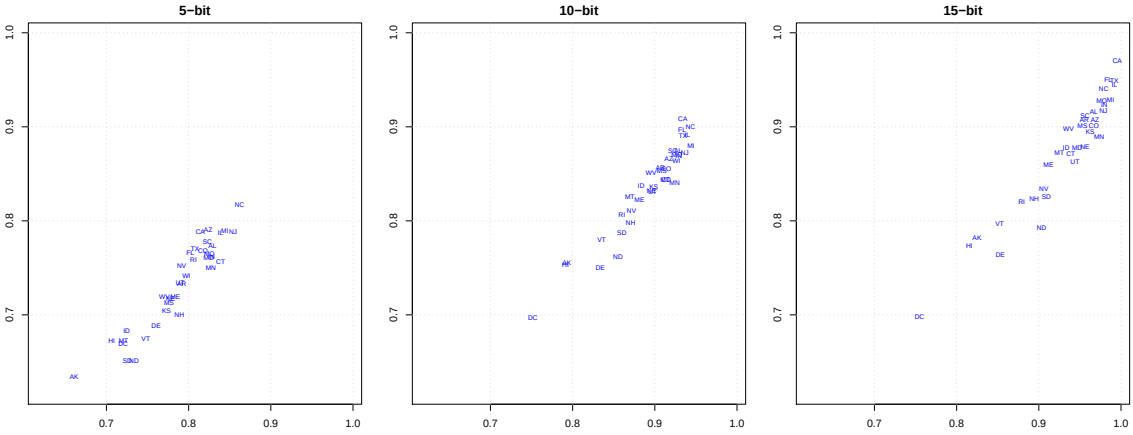

Figure 3: Plots of entropy per bit for population (x-axis) versus rental units (y-axis) for 5,10, and 15-bit balanced geohashes. Only 34 states (and the District of Columbia) are shown because over-plotting otherwise made the image too difficult to read.

exhibit spatial dependence with a higher rate of renting in more urban and less affluent areas as well as spikes near large universities.

With significantly higher levels of precision, across all of the hashes, the entropies converge and the entropy per bit begins to decay to zero. The reason for this is that there are less than 2^{19} census blocks in the entire state of Ohio, and even the standard geohash manages to put every block's centroid into its own bucket after using at least 35-bits. Ultimately the discreteness of the data limits our ability to further increase the entropy even with additional splits. Additionally, the 20-bit balanced geohash would in theory have a near perfect entropy per bit up to 20-bits of hash length if the data were continuous. In fact, the 20-bit geohash at 15-bits of precision (note that this is indistingusihable from a 15-bit geohash) has only and entropy of 0.9 per hash pbit.

Figure 3 plots the entropy of three balanced geohashes for 34 other states at 15-bits of precision. The leftmost panel shows the outcome of only balancing on the first 5-bits. The worst states here are Alaska, Hawaii, South Dakota, and North Dakota. These are all states which have a very low population density (for Hawaii, the density is low when averaged over the area of a bounding box for the entire island chain). The rightmost panel shows the pure effects of the discreteness of the data, as all of the x-values would otherwise be nearly perfect values near 1. Not surprisingly, the most populous state of California fairs the best followed closesly by other large states such as Texas and Florida. The smallest states have the worst entropy here, with DC being the worst at only 0.75 average entropy per bit. In all cases, the hash balanced on population performs similarly in regards to the number of rental properties.

6 Future Extensions

We have presented an extension on the traditional geohash which is balanced on the data volume rather than boxes of constant latitude and longitude. The benefits of this approach have been shown for both spatial and spatial-temporal databases. Theorem 1 establishes the robustness of our model-based approach due to random noise and an empirical study of US Census data demonstrates robustness to model drift. The balanced geohash is usable for indexing large databases as-is; in most applications it will perform most queries in less time, with fewer scans, and lower overhead compared to a standard geohash. There are several extensions of our work which would further increase the applicability of the entropy balanced geohash.

Our balanced geohash $b_m^q(\cdot)$ has two free parameters which give the number of bits for which the hash is balanced (q) and length of the entire hash (m). The latter is largely application specific, but the choice of q has some important efficency aspects. If q is too large the geohash can become quite noisy and overfit to the sample of data used to learn the underlying model. Additionally, in order to map a standard geohash to a balanced geohash we need to know the 2^q break points to the linear interpolation given in Equation 16. So if q is large the memory overhead of the balanced geohash may become prohibitive. In order to better estimate an optimal q, tighter two-way theoretical bounds are needed as compared to those given in Theorem 1. This will likely require replacing the distribution-free approach employeed in our proof with a specific family of models such as Gaussian mixtures. In the meantime we can safely pick a small q without concern, knowing that the noise level is low but that long hash prefixes may be mildly unbalanced.

The paper [11] provides a generic approach for interleaving geospatial and temporal components utilizing the specific implementation optimizations present in HBase. As mentioned, the geohash used in their paper can be seamlessly replaced by the entropy balanced geohash. The exact method of choosing how to incorporate the time and space dimensions are not specified as this will largely be a function of the specific underlying dataset. In the case of the balanced geohash, since the data boxes are already balanced by the distribution of the data, it should be possible to at least specify a methodology for choosing how to interleave these components (with perhaps one or two tuning parameters to tweak based on the specific use-cases). For now the method of combining these two dimensions can be done fairly well manually, particularly if the scheme is kept simple.

Finally, while the balanced geohash is fairly robust to model drift, a method akin to self-balancing trees whereby the balanced geohash could adapt over time would be a great addition to fixed hash presented here. This would be particularly important when implementing a large q-bit balanced geohash. Possible methods for doing this include adaptively splitting large prefixes and joining

small prefixes; this would require re-writting data, but would hopefully not be required very often. Alternatively a scheme could be used to only balance new data as it arrives. In either case, care would need to be taken to avoid large increases to the computational complexity or memory overhead of an adaptive modification to the entropy maximized geohash design.

References

[1] Rudolf Bayer. Symmetric binary b-trees: Data structure and maintenance algorithms. *Acta informatica*, 1(4):290–306, 1972.

[2] Norbert Beckmann, Hans-Peter Kriegel, Ralf Schneider, and Bernhard Seeger. *The R*-tree: an efficient and robust access method for points and rectangles*, volume 19. ACM, 1990.

[3] Fay Chang, Jeffrey Dean, Sanjay Ghemawat, Wilson C Hsieh, Deborah A Wallach, Mike Burrows, Tushar Chandra, Andrew Fikes, and Robert E Gruber. Bigtable: A distributed storage system for structured data. *ACM Transactions on Computer Systems (TOCS)*, 26(2):4, 2008.

[4] Kristina Chodorow. *MongoDB: the definitive guide*. " O'Reilly Media, Inc.", 2013.

[5] Giuseppe DeCandia, Deniz Hastorun, Madan Jampani, Gunavardhan Kakulapati, Avinash Lakshman, Alex Pilchin, Swaminathan Sivasubramanian, Peter Vosshall, and Werner Vogels. Dynamo: amazon's highly available key-value store. In *ACM SIGOPS Operating Systems Review*, volume 41:6, pages 205–220. ACM, 2007.

[6] Aryeh Dvoretzky, Jack Kiefer, and Jacob Wolfowitz. Asymptotic minimax character of the sample distribution function and of the classical multinomial estimator. *The Annals of Mathematical Statistics*, pages 642–669, 1956.

[7] Anthony Fox, Chris Eichelberger, James Hughes, and Skylar Lyon. Spatio-temporal indexing in non-relational distributed databases. In *Big Data, 2013 IEEE International Conference on*, pages 291–299. IEEE, 2013.

[8] Adam Fuchs. Accumulo–extensions to googles bigtable design. Technical report, Technical report, National Security Agency, 2012.

[9] Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung. The google file system. In *ACM SIGOPS Operating Systems Review*, volume 37, pages 29–43. ACM, 2003.

[10] Antonin Guttman. *R-trees: a dynamic index structure for spatial searching*, volume 14. ACM, 1984.

[11] Dan Han and Eleni Stroulia. Hgrid: A data model for large geospatial data sets in hbase. In *Proceedings of the 2013 IEEE Sixth International Conference on Cloud Computing*, pages 910–917. IEEE Computer Society, 2013.

[12] Avinash Lakshman and Prashant Malik. Cassandra: a decentralized structured storage system. *ACM SIGOPS Operating Systems Review*, 44(2):35–40, 2010.

[13] Youzhong Ma, Yu Zhang, and Xiaofeng Meng. St-hbase: a scalable data management system for massive geo-tagged objects. In *Web-Age Information Management*, pages 155–166. Springer, 2013.

[14] Hanan Samet and Robert E Webber. Storing a collection of polygons using quadtrees. *ACM Transactions on Graphics (TOG)*, 4(3):182–222, 1985.

[15] Konstantin Shvachko, Hairong Kuang, Sanjay Radia, and Robert Chansler. The hadoop distributed file system. In *Mass Storage Systems and Technologies (MSST), 2010 IEEE 26th Symposium on*, pages 1–10. IEEE, 2010.

[16] Quentin F. Stout and Bette L. Warren. Tree rebalancing in optimal time and space. *Communications of the ACM*, 29(9):902–908, 1986.

[17] Ronald C Taylor. An overview of the hadoop/mapreduce/hbase framework and its current applications in bioinformatics. *BMC bioinformatics*, 11(Suppl 12):S1, 2010.

[18] Yunqin Zhong, Jinyun Fang, and Xiaofang Zhao. Vegacache: Efficient and progressive spatio-temporal data caching scheme for online geospatial applications. In *Geoinformatics (GEOINFORMATICS), 2013 21st International Conference on*, pages 1–7. IEEE, 2013.

[19] Yunqin Zhong, Jizhong Han, Tieying Zhang, and Jinyun Fang. A distributed geospatial data storage and processing framework for large-scale webgis. In *Geoinformatics (GEOINFORMATICS), 2012 20th International Conference on*, pages 1–7. IEEE, 2012.

[20] Yunqin Zhong, Jizhong Han, Tieying Zhang, Zhenhua Li, Jinyun Fang, and Guihai Chen. Towards parallel spatial query processing for big spatial data. In *Parallel and Distributed Processing Symposium Workshops & PhD Forum (IPDPSW), 2012 IEEE 26th International*, pages 2085–2094. IEEE, 2012.

[21] Yunqin Zhong, Xiaomin Zhu, and Jinyun Fang. Elastic and effective spatio-temporal query processing scheme on hadoop. In *Proceedings of the 1st ACM SIGSPATIAL International Workshop on Analytics for Big Geospatial Data*, pages 33–42. ACM, 2012.