Eliezer de Souza da Silva

“Metric space indexing for nearest neighbor search in multimedia context”

“*Indexação de espaços métricos para busca de vizinho mais próximo em contexto multimídia*”

CAMPINAS
2014

University of Campinas
School of Electrical and Computer Engineering

Universidade Estadual de Campinas
Faculdade de Engenharia Elétrica e de Computação

Eliezer de Souza da Silva

"Metric space indexing for nearest neighbor search in multimedia context"

Supervisor:
Orientador(a): **Prof. Dr. Eduardo Alves do Valle Junior**

"Indexação de espaços métricos para busca de vizinho mais próximo em contexto multimídia"

Master Thesis presented to the Post Graduate Program of the School of Electrical and Computer Engineering of the University of Campinas to obtain a MSc degree in Electrical Engineering in the concentration area of Computer Engineering.

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Engenharia Elétrica da Faculdade de Engenharia Elétrica e de Computação da Universidade Estadual de Campinas para obtenção do título de Mestre em Engenharia Elétrica na área de concentração Engenharia de Computação

THIS VOLUME CORRESPONDS TO THE FINAL VERSION OF THE THESIS DEFENDED BY ELIEZER DE SOUZA DA SILVA, UNDER THE SUPERVISION OF PROF. DR. EDUARDO ALVES DO VALLE JUNIOR.

Este exemplar corresponde à versão final da Dissertação defendida por Eliezer de Souza da Silva, sob orientação de Prof. Dr. Eduardo Alves do Valle Junior.

Supervisor's signature / *Assinatura do Orientador(a)*

CAMPINAS
2014

Ficha catalográfica
Universidade Estadual de Campinas
Biblioteca da Área de Engenharia e Arquitetura
Rose Meire da Silva - CRB 8/5974

Si38m

Silva, Eliezer de Souza da, 1988-
Metric space indexing for nearest neighbor search in multimedia context / Eliezer de Souza da Silva. – Campinas, SP : [s.n.], 2014.

Orientador: Eduardo Alves do Valle Junior.
Dissertação (mestrado) – Universidade Estadual de Campinas, Faculdade de Engenharia Elétrica e de Computação.

1. Método k-vizinho mais próximo. 2. Hashing (Computação). 3. Estruturas de dados (Computação). I. Valle Junior, Eduardo Alves do. II. Universidade Estadual de Campinas. Faculdade de Engenharia Elétrica e de Computação. III. Título.

Informações para Biblioteca Digital

Título em outro idioma: Indexação de espaços métricos para busca de vizinho mais próximo em contexto multimídia
Palavras-chave em inglês:
k-nearest neighbor
Hashing (Computer science)
Data structures (Computer)
Área de concentração: Engenharia de Computação
Titulação: Mestre em Engenharia Elétrica
Banca examinadora:
Eduardo Alves do Valle Junior [Orientador]
Agma Juci Machado Traina
Romis Ribeiro de Faissol Attux
Data de defesa: 26-08-2014
Programa de Pós-Graduação: Engenharia Elétrica

COMISSÃO JULGADORA - TESE DE MESTRADO

Candidato: Eliezer de Souza da Silva

Data da Defesa: 26 de agosto de 2014

Título da Tese: "Indexação de Espaços Métricos para Busca de Vizinho mais Próximo em Contexto Multimídia"

Prof. Dr. Eduardo Alves do Valle Junior (Presidente): E Valle

Profa. Dra. Agma Juci Machado Traina: Agma Traina

Prof. Dr. Romis Ribeiro de Faissol Attux: Romis

Abstract

The increasing availability of multimedia content poses a challenge for information retrieval researchers. Users want not only have access to multimedia documents, but also make sense of them — the ability of finding specific content in extremely large collections of textual and non-textual documents is paramount. At such large scales, Multimedia Information Retrieval systems must rely on the ability to perform search by similarity efficiently. However, Multimedia Documents are often represented by high-dimensional feature vectors, or by other complex representations in metric spaces. Providing efficient similarity search for that kind of data is extremely challenging. In this project, we explore one of the most cited family of solutions for similarity search, the *Locality-Sensitive Hashing* (LSH), which is based upon the creation of hashing functions which assign, with higher probability, the same key for data that are similar. LSH is available only for a handful distance functions, but, where available, it has been found to be extremely efficient for architectures with uniform access cost to the data. Most existing LSH functions are restricted to vector spaces. We propose two novel LSH methods (VoronoiLSH and VoronoiPlex LSH) for generic metric spaces based on metric hyperplane partitioning (random centroids and K-medoids). We present a comparison with well-established LSH methods in vector spaces and with recent competing new methods for metric spaces. We develop a theoretical probabilistic modeling of the behavior of the proposed algorithms and show some relations and bounds for the probability of hash collision. Among the algorithms proposed for generalizing LSH for metric spaces, this theoretical development is new. Although the problem is very challenging, our results demonstrate that it can be successfully tackled. This dissertation will present the developments of the method, theoretical and experimental discussion and reasoning of the methods performance.

Keywords: Similarity Search; Nearest-neighbor Search; Locality-sensitive Hashing; Quantization; Metric Space Indexing; Geometric Data Structure; Content-Based Multimedia Information Retrieval.

Resumo

A crescente disponibilidade de conteúdo multimídia é um desafio para a pesquisa em Recuperação de Informação. Usuários querem não apenas ter acesso aos documentos multimídia, mas também obter semântica destes documentos, de modo que a capacidade de encontrar um conteúdo específico em grandes coleções de documentos textuais e não textuais é fundamental. Nessas grandes escalas, sistemas de informação multimídia de recuperação devem contar com a capacidade de executar a busca por semelhança de forma eficiente. No entanto, documentos multimídia são muitas vezes representados por descritores multimídia representados por vetores de alta dimensionalidade, ou por outras representações complexas em espaços métricos. Fornecer a possibilidade de uma busca por similaridade eficiente para esse tipo de dados é extremamente desafiador. Neste projeto, vamos explorar uma das famílias mais citadas de soluções para a busca de similaridade, o *Hashing Sensível à Localidade* (LSH - *Locality-sensitive Hashing* em inglês), que se baseia na criação de funções de hash que atribuem, com maior probabilidade, a mesma chave para os dados que são semelhantes. O LSH está disponível apenas para um punhado funções de distância, mas, quando disponíveis, verificou-se ser extremamente eficiente para arquiteturas com custo de acesso uniforme aos dados. A maioria das funções LSH existentes são restritas a espaços vetoriais. Propomos dois métodos novos para o LSH, generalizando-o para espaços métricos quaisquer utilizando particionamento métrico (centróides aleatórios e k-medoids). Apresentamos uma comparação com os métodos LSH bem estabelecidos em espaços vetoriais e com os últimos concorrentes novos métodos para espaços métricos. Desenvolvemos uma modelagem teórica do comportamento probalístico dos algoritmos propostos e demonstramos algumas relações e limitantes para a probabilidade de colisão de hash. Dentre os algoritmos propostos para generelizar LSH para espaços métricos, esse desenvolvimento teórico é novo. Embora o problema seja muito desafiador, nossos resultados demonstram que ele pode ser atacado com sucesso. Esta dissertação apresentará os desenvolvimentos do método, a formulação teórica e a discussão experimental dos métodos propostos.

Palavras-chave: Busca de Similaridade; Busca de Vizinho Mais Próximo; Hashing Sensível à Localidade; Quantização; Indexação de espaços métricos; Estruturas de Dados Geométricas; Recuperação de Informação Multimídia.

Contents

In the memory of what we yet have to see

Acknowledgements

I would like to thank all the support I have received during this years of study. The support we need and receive are multivariate and changes over time and I must acknowledge that I would never come this far without all the help and love and have received over these years.

I thank my supervisor, Prof. Dr. Eduardo Valle, for the patience, the trust and the lessons taught about a wide range of aspects in academic and professional life.

I thank and appreciate all the love my family has given me. Thank you dad (Jose), mom (Marta) and little brother (Elienai). Thank my love Juliana Vita. You have endured with me the hard times and always given me encouragement to keep walking in the right track.

I thank this wide extended family of good friends and colleagues that I have around the world. A special thank to friends from Unicamp (specially people at APOGEEU, LCA, IEEE-CIS): Alan Godoy, David Kurka, Rafael Figueiredo, Raul Rosa, Paul Hidalgo, Micael Carvalho, Carlos Azevedo, Michel Fornacielli, Roberto Medeiros, Ricardo Righetto, and many others. I thank my friends from ABU-Campinas, residents at Casa Douglas and visitors: Lois McKinney Douglas, Marcos Bacon, Jonas Kienitz, Tiago Bember, Carla Bueno, Esther Alves, Daniel Franzolin, Pedro Ivo, Maria (Mauê), Edu Oliveira, Nathan Maciel, Micael Poget, Gabriel Dolara, Patrick Timmer, Paulo Castro, Fernanda Longo, and many others. I thank you all for the warm welcome in Campinas, for the friendly environment, the good and bad times spend together. It was really nice to share some of my time in Campinas with you all.

I thank the University of Campinas and the School of Electrical and Computer Engineering for being a high-level educational and research environment. A special thank to Prof. Dr. George Teodoro and Thiago Teixeira for the opportunity of excellent collaborative work. A special thank to CAPES for the financial support.

I thank God for guiding me and being the lenses through-with I see everything.

List of Figures

Chapter 1

Introduction

The success of the Internet and the popularization of personal digital multimedia devices (digital cameras, mobile phones, etc.) has spurred the availability of online multimedia content. Meanwhile, the need to make sense from ever growing data collections has also increased. We face the challenge of processing very large collections, in many media (photos, videos, text and sound), geographically dispersed, with assorted appearance and semantics, available instantaneously at the fingertips of the users. Information Retrieval, Data Mining and Knowledge Management must attend to the needs of the "new wave" of multimedia data [Datta et al., 2008].

Scientific interest in Multimedia Information Retrieval has been steadily increasing, with the convergence of various disciplines (Databases, Statistics, Computational Geometry, Information Systems, Data Structures, etc.), and the appearance of a wide range of potential applications, for both private and public sectors. As an example of this interest we refer to the *Multimedia Grand Challenge*[1]), opened in the ACM Multimedia conference of 2009, which consists of a competition with a set of problems and issues brought by the industry to the scientific community. Amongst others, Datta et al. [2008] present results showing an exponential growth in the number of articles containing the keyword "Image Retrieval" as indicative of the growing interest in the area.

Similarity search is a key step in most of these systems (Information Retrieval, Machine Learning and Pattern Recognition Systems) and there is the need for supporting different distance functions and data formats, as well as designing fast and scalable algorithms, specially for achieving the possibility of processing billions or more multimedia items in a tractable time. The strategy for this task may be twofold: improving data-structure and algorithms for sequential processing and adapting algorithms and data structure for parallel and distributed processing.

The idea of comparing the similarity of two (or more) abstract objects can be formally specified and intuitively comprehended using the concept of *distance* between points in some generic space. Thus, if we entertain the possibility of representing our abstract objects as points in some generic space equipped with a distance measure, we may follow the intuition that the closer the distance between the points, the more similar the objects. This has been established as a standard theoretical and applied framework in Content-based Multimedia Retrieval, Data Mining, Pattern Recognition,

[1] http://www.acmmm12.org/call-for-multimedia-grand-challenge-solutions/

Computer Vision and Machine Learning. Although such approach requires a certain level of abstraction and approximation, it is very practical and appropriate for the task.

However, since there are many possible distance measures for different types of data (strings, text documents, image, video, sounds, etc.), solutions that are effective and efficient only for specific distances measures are useful, but limited. Our purpose is to study existing specific solutions in order to generalize them to generic metric space without loosing efficiency and effectiveness.

In this chapter we will define formally similarity search, discuss the available solutions and present its most common applications.

1.1 Defining the Problem

A definition of Metric Access Methods (MAM) is given by Skopal [2010]:

> Set of algorithms and data structure(s) providing efficient (fast) similarity search under the metric space model.

The metric space model assumes that the domain of the problem is captured by a metric space and that the measure of similarity between the objects of that domain can be represented using some distance function in the metric space. Thus, the problem of finding, classifying or grouping similar objects from a given domain is translated to a geometric problem of finding nearest-neighbor points in a metric space. The challenge is to provide data structures and algorithms that can accomplish this task efficiently and effectively in a context of large scale search.

The obvious approach for the nearest neighbor search is a linear sequential algorithm that scan the whole dataset. However this approach is not scalable in realistic set-ups with large dataset and large query sets. It would be an interesting result that using some refined data structure and algorithm we achieve a much more efficient query performance.

Another challenge for the naive approach is the case of dataset with high-dimensionality, meaning, concentrated histogram of distances, sparsity of points and various other non-trivial phenomena related to high dimensionality. Exact algorithms has failed to tackle with this challenge and approximate methods has showed to be the most promising approach.

Finally we can enunciate our specific problem statement:

> *The development of effective and efficient methods (data structures and algorithms) for approximate similarity search in generic metric space. The question is whether it would be possible to offer better efficiency/effectiveness trade-off than available methods on the literature and the condition that this improvement could be achieved.*

1.2 Applications

Content-based multimedia information retrieval (CBMIR) is an alternative to keyword-based or tag-based retrieval, which works by extracting *features* based on distinctive properties of the multimedia

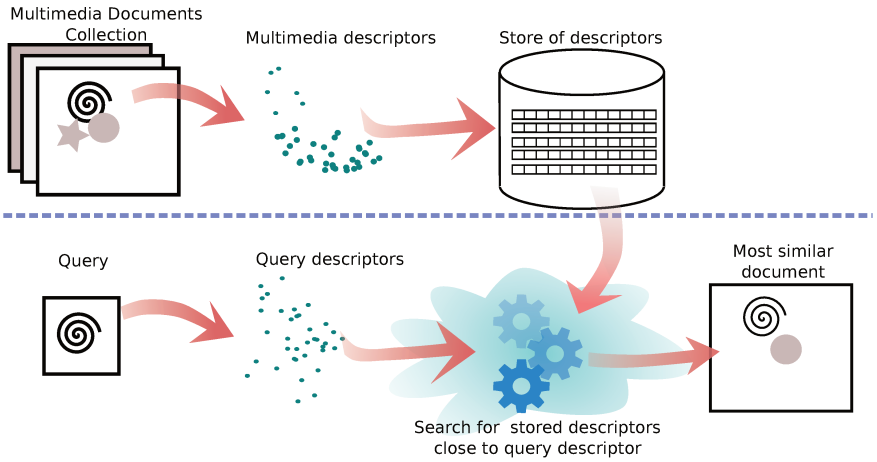


Figure 1.1: Schematics of a CMIR system

objects. Those features are organized in *multimedia descriptors*, which are used as surrogates of the multimedia object, in such a way that the retrieval of similar objects is based solely on that higher level representation (Figure 1.1), without need to refer to the actual low-level encoding of the media. The descriptor can be seen as a compact and distinctive representation of multimedia content, encoding some invariant properties of the content. For example, in image retrieval the successful *Scale Invariant Feature Transform* (SIFT) [Lowe, 2004] encode local gradient patterns around Points-of-Interest, in a way that is (partially) invariant to illumination and geometric transformations.

The descriptor framework also allows to abstract the media details in multimedia retrieval systems. The operation of looking for similar multimedia documents becomes the more abstract operation of looking for multimedia descriptors which have small distances. The notion of a "feature space", that organizes the documents in a geometry, putting close-by those that are similar emerges. Of course, CBIR systems are usually much more complex than that, but nevertheless, looking for similar descriptors often plays a critical role in the processing chain of a complex system.

1.3 Our Approach and Contributions

The field of Similarity Search and Metric data structures are very broad fields of research accumulating decades of effort and a great variety of results. It is not our intention to dismiss all the accumulated research, but rather to offer a modest contribution by presenting a generalization of a successful technique that has been used mostly for Euclidean and Hammming spaces. Locality-Sensitive Hashing is a technique invented more than a decade ago, is based on the existence of functions that map points in a metric space to scalar (integer) values, with a probabilistic bound that points nearby in the original space are mapped to equal or similar values. Until now most existing LSH functions were designed for a specific space – the most common ones for Euclidean and Hamming space. Nevertheless, it has been one of the most cited techniques for nearest-neighbor search in practical and theoretical communities of research.

Taking advantage of the success of Locality-Sensitive Hashing in Euclidean and Hamming

spaces, we sought to investigate a configuration of the technique for metric spaces in general. For this purpose we use data-dependent partitioning of the space (for example, clustering) and hashing associated with the partitions of the space. Intuitively, the idea is that points that are assigned the same partition have a high probability of being relevant nearest neighbors, and points with distinct partitions have a low probability of being real nearest neighbors (and conversely, a high probability of being "far" points). This idea was explored in the course of the research and resulted in two methods that generalizes the LSH for arbitrary metric spaces. This dissertation will present an intuitive description and discussion of the proposed methods, as long with empirical results, but also, as far as possible, will describe our contributions in a formalistic fashion, presenting and demonstrating time and space complexities and proving some important and significant bounds on the hashing probabilities under reasonable assumptions. As far as we know, among the works proposing LSH for general metric space [Kang and Jung, 2012; Tellez and Chavez, 2010; Novak et al., 2010; Silva and Valle, 2013; Silva et al., 2014], this is the first to develop a theoretical characterization of the hashing collision probabilities (Sections 3.1.4, 3.1.3, 3.2.3).

1.3.1 Publications

Some of the contributions in this dissertation has been already reported for the research community. A preliminary work describing VoronoiLSH was accepted and presented at the major national conference on Databases – Brazilian Symposium on Databases. In this work, we introduce the method, review significant part of the literature and report some experimental data supporting the viability of the method. The paper was accepted as a short papers (only for poster session) but was invited to a small group of short paper that was given oral presentation time in the technical sessions. The parallel version of VoronoiLSH using multistage dataflow programming, developed in collaboration between Prof. Dr. George Teodoro, Thiago Teixeira and Prof. Dr. Eduardo Valle, was accepted for the 7th International Conference on Similarity Search and Applications (SISAP 2014) and should be presented there in October, 2014. Besides we are planning for an additional publication reporting VoronoiPlex LSH and taking a more theoretical stance, reporting and extending the theoretical results presented in this dissertation. The complete list of publications related to this research is:

- Eliezer Silva, Thiago Teixeira, George Teodoro, and Eduardo Valle. Large-scale distributed locality-sensitive hashing for general metric data. In *Similarity Search and Applications - Proceedings of the 7th International Conference on Similarity Search and Applications*. Springer, October 2014[Silva et al., 2014]

- Eliezer Silva and Eduardo Valle. K-medoids lsh: a new locality sensitive hashing in general metric space. In *Proceedings of the 28th Brazilian Symposium on Databases*, pages 127–132, Brazil, 2013. SBC. URL `http://sbbd2013.cin.ufpe.br/Proceedings/artigos/sbbd_shp_22.html`[Silva and Valle, 2013]

Chapter 2

Theoretical Background and Literature Review

In this chapter we will present and discuss extensively the concepts necessary to understand our contributions and the state-of-art on the subject. At first, we will introduce the basic mathematical and algorithmic notions and notations. In the sequence, we will discuss how the broader problem of Similarity Search in Metric Spaces has been approached in the specialized literature, however not diving deeply in each specific method references. We will focus the discussion on the main ideas applied to metric indexing and refer the reader to detailed surveys of the algorithms. Further, Locality-Sensitive Hashing is addressed and discussed in details, including a formal presentation of the algorithms and a mathematical demonstrations of selected properties.

2.1 Geometric Notions

In this dissertation we will use the framework of Metric Space to address the problem of Similarity Search. We adopt this model because of it generality and versatility. Any collection of objects equipped with a function measuring the similarity of the objects and obeying a small sets of axioms (the metric axioms) are enabled to be analyzed and processed using the tools developed for metric spaces.

Another advantage is the possibility of using geometric reasoning and intuition in the analysis of objects that are not trivially thought as geometric (for example, strings, or text documents). So, in this setting, it is possible to speak of a "ball" around a string using Edit distances, for example. In this section we will develop some of these intuitions, formalizing geometric notions as balls and triangular inequality over generic metric space.

2.1.1 Vector and Metric Space

We will briefly present the definition and axioms of metric space and the fundamental operations we are interested in metric spaces.

Metric space are general sets equipped with a metric (or distance), which is a real-valued positive function between pairs of points. Given that we have a definition of sets of objects and a definition of a function comparing pairs of objects (obeying certain properties), we have a metric space. The distance function must obey basically four axioms: the image of the distance function of non-negative, the function is symmetric in relation to the order of the points, points with distance zero are equals, and the triangle inequality.

Definition 2.1 (**Metric space properties**). *Metric space: given a set U (domain) and a function $d : U \times U \to \mathbb{R}$ (distance or dissimilarity function), a pair (U, d) is a metric space if the distance function d have the following properties [Chávez et al., 2001]:*

- $\forall x, y \in U$, $d(x, y) \geq 0$ *(non-negativeness)*
- $\forall x, y \in U$, $d(x, y) = d(y, x)$ *(symmetry)*
- $\forall x, y \in U$, $d(x, y) = 0 \Leftrightarrow x = y$ *(identity of indiscernibles)*
- $\forall x, y, z \in U$, $d(x, y) \leq d(x, z) + d(z, y)$ *(triangle inequality)*

If we want to build a notion of neighborhood for points in metric space there is only one tool to use: the distance information between pairs of points. A natural way of accomplish that is to define a distance range around of a point and analyze the region of the space in that range; this idea is similar to looking to an interval with a central point in the real line, using distance between points, generalized to metric space. This definition is useful for a set of indexing method that partition the space using metric balls.

Definition 2.2 (**Open and Closed ball**). *Given a set X in a metric space (U,d) and a point $p \in X$ we define the Open Ball of radius r around p as the set $B_X(p, r) = \{x | d(x, p) < r, x \in X\}$ and the Closed Ball of radius r around p as $B_X[p, r] = \{x | d(x, p) \leq r, x \in X\}$.*

Definition 2.3 (**Finite Vector Spaces**). *A given metric space (U, d) is a finite-dimensional vector space (which for the sake of briefness we will refer simply as a vector space) with dimension D if each $x \in U$ can be represented as a tuple of D real values, $x = (x_1, \ldots, x_D)$. The most common distance for vector space are the L_p distances [Chávez et al., 2001; Skopal, 2010].*

- $\forall x, y \in U$, *where* $x = (x_1, \ldots, x_D)$ *and* $y = (y_1, \ldots, y_D)$,
  $L_p(x, y) = \sqrt[p]{\sum_{i=1}^{D} |x_i - y_i|^p}$

Now we define three fundamental search problems central for similarity search in metric spaces: Range Search, K-Nearest Neighbor Search and (R, c)-Nearest Neighbor Search. Given a subset of metric space and a subset of queries, the Range Search problem is of finding efficiently a metric ball of data points around query points given radius as parameter (called *range*).

Definition 2.4 (**Range search**). *Given the metric space (U, d), a dataset $X \subset U$, a query point $q \in U$ and a range r, find the set $R(q, r) = \{x \in X | d(q, x) \leq r\}$ [Clarkson, 2006].*

Definition 2.5 (K-Nearest Neighbors (kNN) search). *Given the metric space (U, d), a dataset $X \subset U$, a query point $q \in U$ and an integer k, find the set $NN_X(q, k)$, defined as $|NN_X(q, k)| = k$ and $(\forall x \in X \backslash NN_X(q, k))(\forall y \in NN_X(q, k)) : d(y, q) \geq d(x, q)$ (the k closest points to q) [Clarkson, 2006].*

A related problem is the (c, R)-Nearest Neighbor, defined as the approximate nearest neighbor for a given radium.

Definition 2.6 (c-approximate R-near neighbor, or (c, R)-NN [Andoni and Indyk, 2006]). *Given a set X of points in a metric space (U, d) , and parameters $R > 0$, $\delta > 0$, construct a data structure such that, given any query point $q \in U$, if there is $p \in X$ with $d(p, q) \leq R$ (p is a R-near point of q), it reports some point $p^* \in X$ with $d(p^*, q) \leq cR$ (p^* is a cR-near neighbor of q in X) with probability at least $1 - \delta$.*

Given many challenges for large scale search in metric spaces, the choice for approximate and random algorithms is justified because of the possibility of quality/time (efficiency/efficacy) trade-off which may be favorable for a scaling of the algorithms under acceptable error rates. Patella and Ciaccia [2009] presents a survey of approximate methods and the major challenges of approximate search in spatial (vector) and metric data.

2.1.2 Distances

There are a variety of possible distance definitions over a variety of objects. We will restrain ourselves to present just a small sample of the population of metric distances. Our aim is just to illustrate and contextualize our discussion of Similarity Search presenting some distances that are useful in applications, specially in Content-Based Multimedia Retrieval, Machine Learning, Pattern Recognition, Databases and Data Mining.

The usual distances functions in coordinate spaces (in special Euclidean) are L_p distance, known also as Minkowski distance. L_p distances are related to our most elementary geometric intuitions and are widely applied in models of similarity and distance-based algorithms; for example, Skopal [2010] says that more than 80% of relevant literature in metric indexing apply L_p distances.

Definition 2.7 (L_p distances on finite dimensional coordinate spaces)**.** *Given a coordinate metric space (U, d), of dimensionality D, the L_p distance of two points $p = (p_1, \cdots, p_D)$,$q = (q_1, \cdots, q_D)$ is defined as*

$$L_p(p, q) = \left(\sum_{i=1}^{D} |p_i - q_i|^p \right)^{1/p}$$

Euclidean metric space with D dimensions equipped with a L_p metric may be referred as L_p^D space.

Three L_p distances with widespread use are the Euclidean ($p = 2$), Manhattan ($p = 1$, also known as Taxicab metric), and the Chebyshev ($p = \infty$, also known as the maximum metric). The

Euclidean distance is related with common analytic geometry, and is the generalization for higher dimensions of the Pythagorean distance of two points in the plane – the square root of a sum of squares.

Another import definition is the distance from a point to a set, a composite of distance to individual points in the set. This concept will be applied in the analysis of our proposed methods

Definition 2.8 (Point distance to sets)**.** *Given a metric space (U, d), a point $x \in U$ and set $C \subseteq U$, $d(x, C)$ is defined as the distance from x to the nearest point in C.*

- $d(x, C) = \min\{d(x, c)|c \in C\}$

These are restricted examples of a long and diverse set of distances; we choose to define here only the ones that are necessary for the understanding of concepts and algorithms described in this dissertation. Deza and Deza [2009] has done a compendious work of cataloging an exhaustive collection of distance and metric; that is proper reference for the reader interested in further details and more examples of distances and metric.

2.1.3 Curse of Dimensionality

The "Curse of Dimensionality" (CoD) is a generic term associated with intractability and challenges with growing dimensionality of the search space in algorithms for statistical analysis, mathematical optimization, geometric operations and other areas. It is related to the fact that the geometric intuition in lower dimensionality sometimes are totally changed in higher dimensionality, and many times in a way that undermines the strategies that worked in lower dimensions. However not all properties associated with the curse are always negative: depending on the subject area the curse can be a blessing [Donoho et al., 2000]. It was first coined by Bellman [1961] as an argument against the strategy that use a discretization and a brute force over the search space, in the context of optimization and dynamic programming. The argument is simple: the number of partition grows exponentially with the dimensionality, meaning, to approximately optimize of function over a space with dimensionality D using grid search, to achieve a error ϵ we would need search over $(1/\epsilon)^D$ grids [Donoho et al., 2000].

In Similarity Search (Metric Access Methods and Spatial Access Methods) the curse has been related to the difficulty to prune the search space as the (intrinsic) dimensionality grows – the performance of many methods in high-dimensions are no better than linear scan. Also the very notion of nearest-neighbor in high-dimensional space becomes blurred, specially when the distance to the nearest point and the distance to the farthest point in the dataset becomes indistinguishable: it has been demonstrated general conditions over the distance distribution, covering a wide class of data and query workload, implying with high probability that as the dimensionality increases the distance to farthest point and to the closest point are practically the same [Aggarwal et al., 2001; Shaft and Ramakrishnan, 2006; Beyer et al., 1999]. This effect has been related to concentration of distance and the intrinsic dimensionality of the dataset [Chávez et al., 2001; Shaft and Ramakrishnan, 2006; Pestov, 2000, 2008], but it is still an open question what is the formulation that explains how

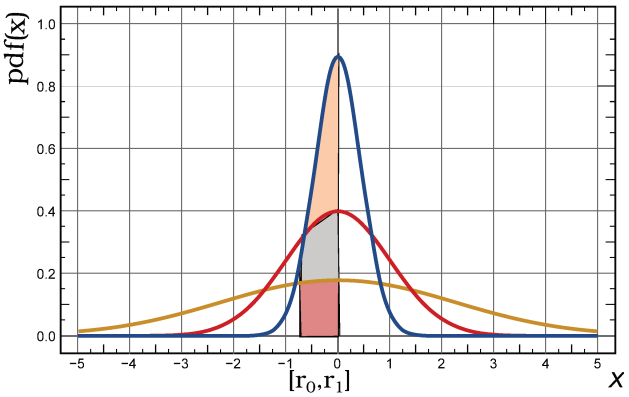


Figure 2.1: Illustration of a hypothetical situation where the distance distribution gets more concentrated as the dimensionality increases and the filtered portion of the dataset using a distance bound $r_0 \leq d(p,q) \leq r_1$. If in a low dimensional space this bound implied in searching over 10% of the dataset, in a higher dimensional space this bound could lead to approximately 50% of the dataset being evaluated.

the difficulties for similarity search emerges. In fact Volnyansky and Pestov [2009] demonstrates that, under several general workloads and appropriate assumptions, pivot-based methods presents concentration of measure using the fact that these methods can be seen as a 1-Lipschitz embedding (the mapping to pivot space) and this can degenerate the performance of a large class of metric indexing methods. Intuitively is possible to see that if we are using bounds on the distance distribution to perform a fast similarity search, a sharp concentrated distribution can be much more difficult to search: as the distribution get concentrated around some value, a larger portion of the dataset is not going to be filtered by the bounds – in the limit we would have a full scan of the dataset. If in a low dimensional space, a distance bound $[r_0, r_1]$ ($r_0 \leq d(p,q) \leq r_1$) can be applied to filter out a huge portion of the dataset, in a higher space with concentrated distance distribution, this same bound could filter out very few points; in order to achieve an equivalent selectivity, we would need a more sharper bound, a process that could lead to very unstable query processing, where a small numerical perturbation on the distances would imply large performance loss (see Figure 2.1 for a illustration).

Recently the *hubness* phenomenon has been studied and associated with high-dimensionality and the CoD. Hubs are points that consistently appear in the kNN set of many other points in the dataset. Although hubness and concentration of measure are distinct phenomena, both emerges with increasing dimensionality. However, this property can also be exploited positively to avoid problem of high-dimensionality. In fact, if there is natural clusters in the dataset, and the hubness property holds, it is expected that with increasing dimensionality the clusters become more well-defined [Radovanović et al., 2009, 2010], in this situation metrics based on shared nearest-neighbors can offer good potential results [Flexer and Schnitzer, 2013].

2.1.4 Embeddings

Since there is long standing solutions for proximity search in more specific space (for example Euclidean or Vector spaces), a possible general approach to the problem is the mapping from general metric space to those space where a solution already exists – a general technique in Computer Science, reduction of an unsolved problem to another problem with a known solution. In order for those mappings to be useful, some properties of the original space must be preserved, specifically the distance information for any pairs of points in the original space. There are also techniques focusing on preserving *volume* (or content) information between spaces [Rao, 1999], but we will not discuss them.

Our theoretical and practical interest for approximate similarity search relies over a class of embedding that preserve distances between pairs to a certain degree of distortion from the original distance. In special we are interested in Lipschitz embedding with a fixed maximum distortion.

Definition 2.9 (c-Lipschitz mapping [Deza and Deza, 2009])**.** *Given a positive scalar c, a c-Lipschitz mapping is a mapping $f : U \to S$ such that $d_S(f(p), f(q)) \leq cd(p, q)$, for $p, q \in U$.*

Definition 2.10 (Bi-Lipschitz mapping [Deza and Deza, 2009])**.** *Given a positive scalar c, metric spaces (U, d_U) and (S, d_S), a funtion $f : U \to S$ is a c-bi-Lipschitz mapping if exists a scalar $c > 0$ such that: $1/cd_U(p, q) \leq d_S(f(p), f(q)) \leq cd_U(p, q)$, for $p, q \in U$.*

A simple example of 1-Lipschitz embedding is the pivot-space (as we will see later, this is the basis for a whole family of metric indexing methods): given metric space (M, d), take k points as the pivot set $P = \{p_1, \cdots, p_k\}$, and build the mapping $g_P(x) = (d(x, p_1), \cdots, d(x, p_k))$. Calculating the maximum metric over two embedded points $x, y \in M$ we obtain $L_\infty(g_P(x), g_P(y)) = \max_{i \in 1,\cdots k}\{|d(x, p_i) - d(y, p_i)|\}$ and by triangle inequality $|d(x, p_i) - d(y, p_i)| \leq d(x, y), \forall p_i \in M$[1].

In general there are results indicating that an $n-$points metric can be embedded in Euclidean space with $log(n)$ distortion [Matoušek, 1996; Bourgain, 1985; Matoušek, 2002]. The reader interested in further details about metric space embeddings should refer to the works of Deza and Laurent [1997] and Matoušek [2002].

By relying on the general theory of embedding, one can advance the theoretical and practical understanding of a great number of similarity search algorithms in metric space. Although not always explicitly stated, many times embeddings are essential components of those algorithms. In the next section we will discuss how this is accomplished in some classes of algorithms.

2.2 Indexing Metric Data

The basic purpose of indexing metric data is to avoid a full sequential search, decreasing the number of distance computations and points processed. We may think of it as a data structure partitioning the data space in such a way that the query processing is computationally efficient and

[1] $d(x, y) + d(y, p_i) \geq d(x, p_i) \Rightarrow d(x, y) \geq d(x, p_i) - d(y, p_i)$ and $d(x, y) + d(x, p_i) \geq d(y, p_i) \Rightarrow d(x, y) \geq d(y, p_i) - d(x, p_i)$, meaning that $|d(y, p_i) - d(x, p_i)| \leq d(x, y)$

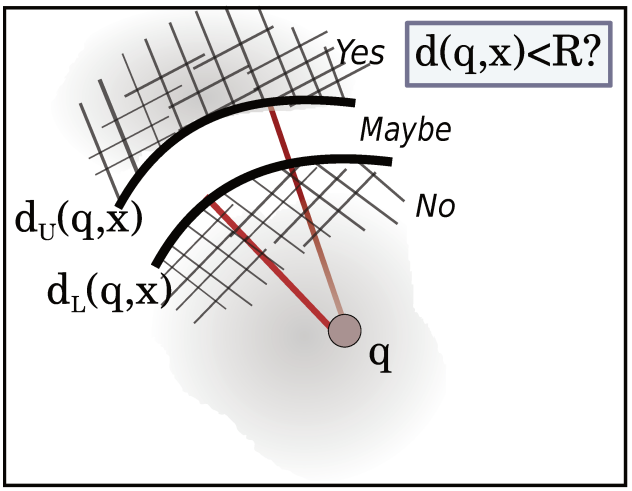


Figure 2.2: Domination mechanism and range queries: which range queries can be answered using the upper and lower bounding distance functions?

effective. Even if we increase the amount of pre-processing, in the long run, in a scenario with multiples queries arriving, the whole process is much more efficient than sequential search. So a good indexing structure should display a polynomial complexity in the pre-processing phase and a sub-linear complexity in the query search phase. We may also understand the index structure as the implementation of some sort of metric filtering strategy.

Hetland [2009], in a more recent survey of metric indexing methods, tries to uncover the most essential principles and mechanisms from the vast literature of metric methods, synthesizing previous surveys and reference work on metric indexing [Chávez et al., 2001; Hjaltason and Samet, 2003; Zezula et al., 2006]. Leaving aside detailed algorithm description and thorough literature discussion, the author focus on the metric properties, distance bounds and index construction mechanisms constituting the building blocks of widespread metric indexing methods. In general, before taking into account specific metric properties, the author highlights three mechanisms for distance index construction: domination, signature and aggregation. Domination is the adoption of computationally cheaper distances, lower and upper bound of the original distance, in order to avoid distance calculation (if the distance function d_U is always greater or equal to another distance d, than d_U *dominates* d). Applying domination mechanisms to range queries is useful for avoiding expensive distance computation. For example, given two distance function bounding the original distance such that for any pairs of points p and q, $d_L(p,q) \leq d(p,q) \leq d_U(p,q)$, if range R is greater than $d_U(p,q)$ the query "$d(p,q) < R$?" can be positively answered without the calculation of $d(p,q)$. Symmetrically, distance computation can be avoided when range R is less than $d_L(p,q)$ (Figure 2.2 illustrates both situation). For instance, if distance functions d_L and d_U is computationally cheaper than distance d, we obtain a mechanism for improving the index efficient. Another lower bounding mechanism is a mapping that take points from the original space to a *signature* space. The *signature* of a point is a new representation of the object in distinct space such that the distance computed using the signature is a lower bound to the distance in the original

space. Take two points p and q in the original space U, a *non-expansive mapping* (also known as *1-Lipschitz mapping*, Definition 2.9) is a function from the original space U to the signature space S, $\sigma : U \to S$, such that $d_S(\sigma(p), \sigma(q)) \leq d(p, q)$; a non-expensive mapping is an example of lower bound in the signature space. It is possible to see that any Lipschitz or Bi-Lipschitz (Definition 2.10) mapping could be used as a signature mapping. The *aggregation* mechanism, used more often in conjunction with signature or dominance mechanism, consists in partitioning the search space in regions such that the distance bounds are applied on the aggregate rather than in individual points of the dataset. Those principles are used in specific metric mechanism for indexing:

- **Pivoting and Pivot Scheme**: a selected set of points $P = \{p_1, \cdots, p_k\}$ and the dataset is stored with precomputed distance information that later is used to lower-bound the distance from queries to points in the dataset. Given a point q in the dataset, a query q and the pivot set, there is a lower-bound to the query to points in the dataset given by $\max_{i \in 1,\cdots k}\{|d(p, p_i) - d(q, p_i)|\} \leq d(p, q)$.

- **Metric balls and shells**: pivot and search spaces are organized in ball and shell partitions in order to avoid distances calculations (aggregation). Generally some information regarding the radius of the aggregate region must be stored with the index in order to apply metric distance bound using a representative point of the region, but taking into consideration all the other points. Most metric tree methods apply this technique, but also methods like List-of-Clusters (a hierarchical list of metric balls inside metric shells) [Chávez and Navarro, 2005; Fredriksson, 2007].

- **Metric Hyperplanes and Dirichlet Domains**: in this case the regions are not of a particular shape, but are the result of dividing the metric dataset using metric hyperplanes. Taking two point p_1 and p_2, we can divide the space using the distance to these points; points closer to p_1 form a region, and points closer to p_2 is another region – the separating metric hyperplane is the set of points with $d(x, p_1) = d(x, p_2)$. This idea can be generalized if we use many reference points, or hierarchical organization of the partitions. Chávez et al. [2001] use the compact partition relation to analyze different techniques relying on metric hyperplanes.

The survey by Chávez et al. [2001], despite being dated and not covering a considerable number of new relevant techniques, is still very relevant since many challenges for similarity search are still not solved and open to new methods and approaches. However, Chávez et al. [2001] shows that those different views of metric indexing are equivalent and can be comprehensively understood using the unifying model of *equivalent relations*, equivalent classes and partitioning. This unified model is applied to the development of a rigorous algorithmic analysis of metric indexing methods, specifically methods based on pivot-spaces and metric hyperplanes (denominated compact partitions also), where lower-bound on the probability of discarding whole partition classes are given and are related to a specific measure of intrinsic dimensionality (square of the mean over the square of the variance of the distance histogram).

Metric trees: Burkhard and Keller [1973] (BK-Tree), in their seminal work, introduced a tree structure for searching with discrete metric distances. More recently, Uhlmann [1991] introduced the concept of a "metric tree", which can be seen as a generalization of the BK-Tree for general metric distances. The Vantage-Point Tree (VPT), by Yianilos [1993] is a binary metric tree that starts with a random root point, and then separates the dataset into left and right subtrees using the median distance as separating criterion. The M-tree [Ciaccia et al., 1997] is a balanced tree constructed using only distance information: at search phase, it used the triangle-inequality property to prune the nodes to be visited. Slim-tree [Traina et al., 2000, 2002] is a dynamic metric tree with enhanced feature including the minimizing the overlap between nodes of the metric tree using a Minimum Spanning Tree algorithm.

Permutation-based Indexing: a recent family of MAM is the permutation-based indexing methods. This methods are based on the idea of taking a reference set (*permutants*), and using the perspective of any point to the permutants, the distance ordering from a point in the dataset to the permutants, as a relevant information for approximate search [Chávez et al., 2005; Gonzalez et al., 2008; Amato and Savino, 2008]. This ordering is very interesting because it is mapping from a general metric space to a permutation space, which a potentially cheaper distance function that can be exploited to render new bound and offer better performance. In fact, this mapping can be used with a inverted-file and compared using Spearman Rho, Kendall Tau or Spearman Footrule measure to perform approximate search in an effective procedure [Gonzalez et al., 2008; Amato and Savino, 2008].

For a comprehensive survey of Metric Access Methods the reader may refer to existing surveys [Chávez et al., 2001; Hjaltason and Samet, 2003; Zezula et al., 2006; Samet, 2005; Hetland, 2009; Clarkson, 2006]. Skopal [2010] and Zezula [2012] offer a critical review of the evolution of the area, and an evaluation of possible future directions, making explicit claims about the necessity of even more scalable algorithms for the future of the area. The PhD thesis of Batko [2006] is also a good recent reference and survey for Metric Access Methods and general principles of metric indexing.

2.3 Locality-Sensitive Hashing

The LSH indexing method relies on a family of locality-sensitive hashing function H [Indyk and Motwani, 1998] to map objects from a metric domain X in a D-dimensional space (usually $\mathbb{R}^d$) to a countable set C (usually $\mathbb{Z}$), with the following property: nearby points in the metric space are hashed to the same value with high probability. It is presented in the seminal article [Indyk and Motwani, 1998] as an efficient and theoretically interesting approach to the Approximate Nearest-Neighbors problem and later also as a solution for the (R, c)-NN problem [Datar et al., 2004; Andoni and Indyk, 2006] (Figure 2.3). A parallel line of work by Broder et al. [2000] developed the idea of MinHash (Min-Wise Independent Permutations) for fast estimation of set and documents similarity and later SimHash [Charikar, 2002] for cosine distance in vectors, using a

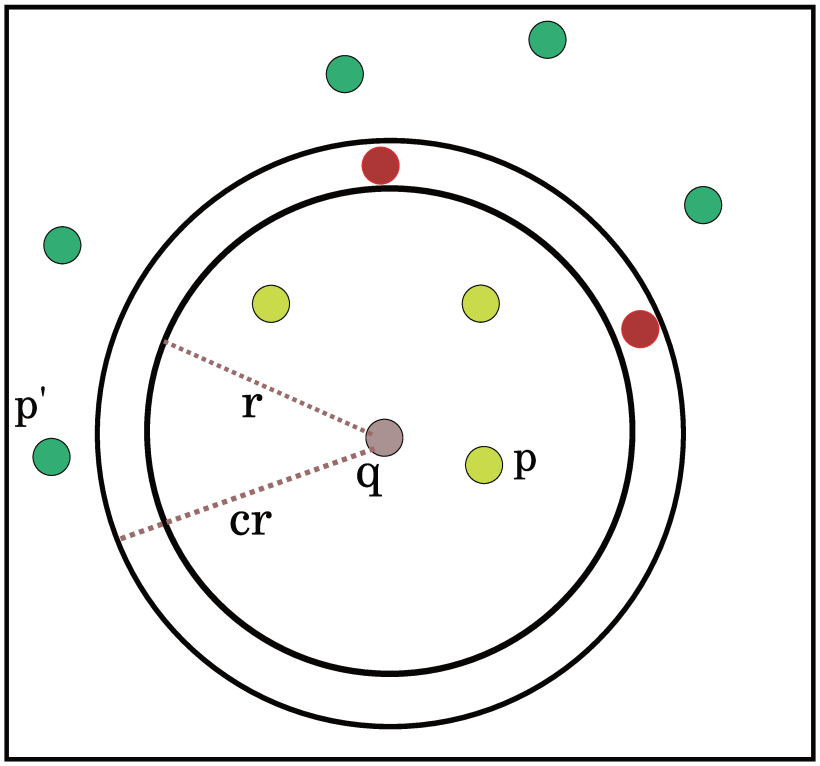


Figure 2.3: LSH and (R, c)-NN: probability of reporting points inside closed ball $B_X(q, r)$ as R-Near of q is high. Probability of reporting points outside closed ball $B_X(q, cr)$ as cr-near neighbor of q is lower. Points in the middle may be wrongly reported as R-Near with some probability

(slightly) distinct formulation of LSH. However, both formulations relied on a definition that related similarities (and distances) between objects and hash probabilities. We will follow the formulation of Indyk and Motwani [1998][2].

Definition 2.11. *Given a distance function $d : X \times X \to \mathbb{R}^+$, a function family $H = \{h : X \to C\}$ is (r, cr, p_1, p_2)-sensitive for a given data set $S \subseteq X$ if, for any points $p, q \in S$, $h \in H$:*

- *If $d(p, q) \leq r$ then $Pr_H[h(q) = h(p)] \geq p_1$ (probability of colliding within the ball of radius r),*
- *If $d(p, q) > cr$ then $Pr_H[h(q) = h(p)] \leq p_2$ (probability of colliding outside the ball of radius cr)*
- *$c > 1$ and $p_1 > p_2$*

A function family G is constructed by concatenating M randomly sampled functions $h_i \in H$, such that each $g_j \in G$ has the following form: $g_j(\mathbf{v}) = (h_1(\mathbf{v}), ..., h_M(\mathbf{v}))$. The use of multiple h_i

[2]It is worth noting that the authors of these articles, Andrei Broder, Moses Charikar and Piotr Indyk, were named recipients of the prestigious Paris Kanellakis Theory And Practice Award in 2012 for their contribution in the development of LSH. *ACM Awards Page: The Paris Kanellakis Theory and Practice Award*, `http://awards.acm.org/kanellakis/`, acessed in June, 7, 2014.

functions reduces the probability of false positives, since two objects will have the same key for g_j only if their value coincide for all h_i component functions. Each object **v** from the input dataset is indexed by hashing it against L hash functions $g_1, \ldots, g_L$. At the *search phase* a query object **q** is hashed using the same L hash functions and the objects stored in the given buckets are used as the candidate set. Then, a ranking is performed among the candidate set according to their distance to the query, and the k closest objects are returned.

LSH works by boosting the locality sensitiveness of the hash functions. As M grows, the probability of a false positive (points that are far away having the same value on a given g_j) drops sharply, but so grows the probability of a false negative (points that are close having different values). But as L grows and we check all hash tables, the probability of false negatives falls, and the probability of false positives grows. LSH theory shows that it is possible to set M and L so to have a small probability of false negatives, with an acceptable number of false positives. This allows the correct points to be found among a small number of candidates, dramatically reducing the number of distance computations needed to answer the queries.

The need to maintain and query L independent hash tables is the main weak point of LSH. In the effort to keep both false positives and false negatives low, there is an "arms race" between M and L, and the technique tends to favor large values for those parameters. The large number of hash tables results in excessive storage overheads. Referential locality also suffers, due to the need to random-access a bucket in each of the large number of tables. More importantly, it becomes unfeasible to replicate the data on so many tables, so each table has to store only pointers to the data. Once the index retrieves a bucket of pointers on one hash table, a cascade of random accesses ensues to retrieve the actual data.

2.3.1 Basic Method in Hamming Space

The basic scheme [Indyk and Motwani, 1998] (Hamming LSH) provided locality-sensitive families for the Hamming distance on Hamming spaces, and the Jacquard distance in spaces of sets.

The original [Indyk and Motwani, 1998] method is limited to Hamming space (bit-vectors of fixed size) using Hamming distance (d_H, which is the number of different bits at corresponding positions, a sum of exclusive-or operation) and point-set space using Jaccard similarities. Equation 2.1 describes the hash functions family for Hamming distance. The idea is to choose one position of the hamming point coordinate as representative of the point.

$$\begin{aligned} H = \{h_i : \{0,1\}^D \rightarrow \quad & \{0,1\} \in \mathbb{Z}\} \\ h_i((b_1, \cdots, b_D)) = \quad & b_i, 1 \leq i \leq D \end{aligned} \tag{2.1}$$

Because the distance between two hamming points is bounded and the number of different hashing function also is bounded, it is easy to see that the probabilities p_1 and p_2 are bounded and obeying the restriction of the LSH definition. Indeed, there are D possible functions h_i for a given Hamming space of dimensionality D and the hamming distance between two points measures the number of times (over the D possible) that the hashing of the two points are supposed to have distinct values.

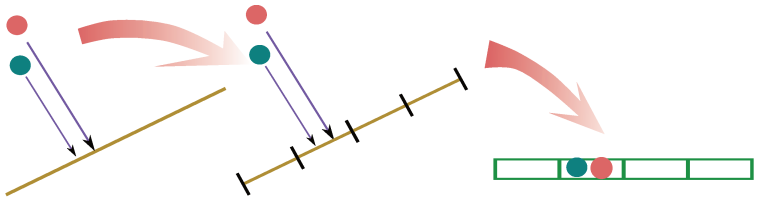


Figure 2.4: E2LSH: projection on a random line and quantization

Thus, the probability of not colliding is given by:

$$P_{h_i \sim H}[h_i(q) \neq h_i(p)] = \frac{d_H(q,p)}{|H|} = \frac{d_H(p,q)}{D}$$

Considering the case of $d_H(q,p) > cr$, we obtain

$$P_{h_i \sim H}[h_i(q) \neq h_i(p)] = \frac{d_H(p,q)}{D} > \frac{cr}{D}$$

$$\Rightarrow P_{h_i \sim H}[h_i(q) = h_i(p)] = 1 - P_{h_i \sim H}[h_i(q) \neq h_i(p)] < 1 - \frac{cr}{D}$$

And if $d_H(q,p) \leq r$,

$$P_{h_i \sim H}[h_i(q) \neq h_i(p)] \leq \frac{r}{D}$$

$$\Rightarrow P_{h_i \sim H}[h_i(q) = h_i(p)] \geq 1 - \frac{r}{D}$$

So it is clear that H is a $(r, cr, 1 - \frac{r}{D}, 1 - \frac{cr}{D})$-sensitive hashing function for Hamming metric space.

2.3.2 Extensions in Vector Spaces

For several years those were the only families available, although extensions for L_1-normed (Manhattan) and L_2-normed (Euclidean) spaces were proposed by embedding those spaces into Hamming spaces [Gionis et al., 1999].

The practical success of LSH, however, came with the E2LSH[3] (Euclidean LSH) [Datar et al., 2004], for L_p-normed space, where a new family of LSH functions was introduced (Figure 2.4 is a graphical representation of this equation):

$$H = \{h_i : \mathbb{R}^D \rightarrow \mathbb{Z}\} \tag{2.2}$$

$$h_i(\mathbf{v}) = \left\lfloor \frac{\mathbf{a}_i.\mathbf{v} + b_i}{w} \right\rfloor \tag{2.3}$$

[3] *LSH Algorithm and Implementation (E2LSH)*, accessed in 22/09/2013. `http://www.mit.edu/~andoni/LSH/`

$\mathbf{a}_i \in \mathbb{R}^D$ is a random vector with each coordinate picked independently from a Gaussian distribution $N(0,1)$, b_i is an offset value sampled from uniform distribution in the range $[0,\ldots,w]$ and w is a fixed scalar that determines quantization width. Applying h_i to a point or object $\mathbf{v}$ corresponds to the composition of a projection to a random line and a quantization operation, given by the quantization width w and the floor operation.

Andoni and Indyk [2006] extends LSH using geometric ball hashing and lattices. This approach achieves a complexity near the lower bound established by Motwani et al. [2006] and O'Donnell et al. [2014] for LSH-based algorithms. Instead of projecting the high-dimensional points to a line, as in Datar et al. [2004] and Indyk and Motwani [1998], it is done a projection to a lower-dimensional space with dimension $k \ll D$ greater than one (D is the original space dimension) and "ball partitioning" quantization on the low-dimensional space. Nevertheless, for practical purposes (fast encoding and decoding) Leech lattices quantizer is used as alternative to "ball partitioning" quantizer. The Leech lattice is a dense lattice in the 24-dimensional space introduced by Leech [1964] for the problem of ball packing.

Query adaptive LSH [Jegou et al., 2008] introduces a dynamic bucket filtering scheme based on the relative position of the hashed (but not quantized) value of the query point to the quantizer cell frontier. Suppose the hashing function $h(x)$ may be seen as the composition of a function $f : M \to \mathbb{R}$, mapping the point to a scalar value, and a quantizer $g : x \to \lfloor x \rfloor$. The cell frontier of $h(x)$ is $\lfloor f(x) \rfloor$ and $\lfloor f(x) \rfloor + 1$, and distance to the center of the cell given by $|\lfloor f(x) \rfloor - f(x) + 1/2|$ can be seen as a relevance criterion for the quality of the bucket. The further from the cell frontier (or closer to the cell center), the better the quality of the bucket. Using this as a relevance criteria, the "best" buckets are selected without using any point-wise distance calculation.

Panigrahy [2006] proposes *entropy based LSH*, an alternative approach to LSH in high-dimensional nearest neighbor search that employs very few hash tables (often just one), and for the search phase, probes multiple randomly "perturbed" buckets around the query bucket in each table, in order to maintain the probabilistic response guarantees. Theoretical analysis support the assertion that this approach renders similar performance to the basic LSH. Although the number of probes is very large, increasing the query costs, the trade-off might still be interesting for some applications, specially in very large-scale databases, since main memory might be a system-wide constraint, while processing time might be available.

Multiprobe LSH [Lv et al., 2007] follows the approach of Panigrahy, that, instead of using the expensive random probes, generates a carefully probing sequence, visiting first the buckets most likely to contain the results. The probing sequence follows a success likelihood estimation, which generates an optimal probing sequence, whose quality can be controlled by setting the number of probes. Joly and Buisson's *a posteriori multi-probe LSH* [Joly and Buisson, 2008] extends that work by turning the likelihoods into probabilities by incorporating a Gaussian prior estimated from training data, into a scheme aptly called *a posteriori LSH*. They employ an "estimated accuracy probability" stop criterion instead of a number of probes.

Other contributions approach the problem of parameter tuning and the dynamic adaptation of the LSH method. *LSH Forest* [Bawa et al., 2005] uses variable-length hashes and a prefix-tree structure for self-tuning of the LSH parameters. Ocsa and Sousa [2010] propose a similar adaptive multilevel

LSH index for dynamic index construction, changing the hash length parameter in each level of the structure. A more recent work by Slaney et al. [2012] focus on the optimization of the parameters using prior knowledge of the distance distribution of the base to achieve a desired quality level of the nearest neighbor method.

2.3.3 Structured or Unstructured Quantization?

In original LSH formulation [Indyk and Motwani, 1998; Gionis et al., 1999; Datar et al., 2004; Andoni and Indyk, 2006], the hashing schemes are based on structured random quantization of the data space and this regularity is useful for the bounds on the hash collision probability. However, the question could be raised whether this structure is really necessary for locality-sensitiveness and if other non-regular partitioning could be applied. This discussion concerns how distinct space or data partitioning be useful in the LSH framework.

Indeed, early proposals, Hamming LSH and E2LSH for example, used exclusively regular quantizers of the space independent with the data. However, *K-means LSH* [Paulevé et al., 2010] presents a comparison between *structured* (random projections, lattice) and *unstructured* (K-means and hierarchical K-means) quantizers in the task of searching high dimensional SIFT descriptors, resulting on the proposal of a new LSH family based on the latter (Equation 3.2). Experimental results indicate that the LSH functions based on unstructured quantizers perform better, as the induced Voronoi partitioning adapts to the data distribution, generating more uniform hash cells population than the structured LSH quantizers. A drawback of this work is the reliance solely on empirical evidence and the lack of theoretical results demonstrating, for example, how the collision probabilities in K-means LSH could offer better results than the previous version based on structured quantization.

Advancing the theoretical analysis in the direction of unstructured quantization, Andoni et al. [2014] developed an adapted version of LSH using two-level data-dependent hashing. Two reference works [Andoni and Indyk, 2006; Andoni, 2009] demonstrate lower bounds for approximate nearest-neighbor search with locality-sensitive hashing (lower bound on ρ, given that the query complexity is $dn^{\rho+o(1)}$); also an LSH family based on lattices and ball partitioning achieving near-optimal performance is presented. Further more, relying on the same ideas, a two-level data-dependent hashing may be constructed and perform (theoretically) even better [Andoni et al., 2014].

Finally, K-means and hierarchical K-means are clustering algorithms for vector spaces, restricting the application of this approach to metric data. In order to overcome this limitation we turn to a clustering and data-dependent partitioning algorithms designed to work in generic metric spaces as *K-medoids clustering* and random Voronoi partitioning.

2.3.4 Extensions in General Metric Spaces

As far as we know, there are only three works that tackles the problem of designing LSH indexing schemes using only distance information, both of them exploiting the idea of Voronoi partitioning of the space.

M-Index *M-Index* [Novak and Batko, 2009] is a Metric Access Method for exact and approximate similarity search based on universal mapping from the original metric space to scalar values in $[0, 1]$. The values of the mapping are affected by the permutation order of a set of reference points and the distance to these points. It uses a broad range of metric filtering when performing the query processing. In a follow-up work [Novak et al., 2010], this indexing scheme is analyzed empirically as a locality-sensitive hashing for general metric spaces.

Permutation-based Hashing Tellez and Chavez [2010] presents a general metric space LSH method using *permutation based indexing*, combining a technique of mapping and hashing. In the permutation index approach the similarity is inferred by the perspective on a group of points called *permutants* (if point p sees the permutants in the same order as point q, so p and q are likely to be close to each other). In a way, this is similar to embedding the data to a proper space for LSH and then applying a built-in (in this proper space) LSH function. Indeed the method consists in two steps: first it creates a permutation index (designed to be compared using Hamming distance); and then it hashes the permutation indexes using Hamming LSH [Indyk and Motwani, 1998].

DFLSH The DFLSH (Distribution Free Locality-Sensitive Hashing) is introduced in a recent paper by Kang and Jung [Kang and Jung, 2012]. The idea is to randomly choose t points from the original dataset (with $n > t$ points) as *centroids* and index the dataset using the nearest centroid as hash key — this construction yields an approximately uniform number of points-per-bucket: $O(n/t)$. Repeating this procedure L times it is possible to generate L hash tables. A theoretical analysis is provided, and with some simplifications, it shows that this approach follows the locality-sensitive property.

2.4 Final Remarks

We presented the fundamental mathematical and algorithmic concepts essential for this work, with emphasis on the definition of metric space and different type of similarity search over metric spaces. Despite the large number of metric indexing methods, we relied on the general techniques and principles in the literature to give a wide description of the intuition behind most of the metric indexing methods. Nevertheless it is still an open challenge to scaling up of the algorithms in terms of size of the dataset and dimensionality. Given the flexibility of permutation-based methods for a possible family of LSH methods, and taking in consideration the idea of data-based quantizer for Euclidean spaces, we will present in next chapter two algorithms for Similarity Search in general metric spaces using a family of LSH functions in metric spaces.

Chapter 3

Towards a Locality-Sensitive Hashing in General Metric Spaces

This chapter introduces our main contribution. We present two methods for Locality-Sensitive Hashing in general metric spaces and a theoretical characterization of the proposed methods as locality-sensitive. In general metric spaces, all structural and local information is encoded in the distance between the points, forcing us to somehow use this in the design of hashing functions. Our practical solution is partitioning the metric space using clustering or simple induced Voronoi diagrams (or Dirichlet Domains) by a subset of points (generalized hyperplane partitioning), assigning numbers to the partitions and using this to build hashing functions. We present VoronoiLSH in Section 3.1 and VoronoiPlex LSH in Section 3.2, using an initial intuitive presentation and then a theoretical discussion of the methods. Section 3.3 introduces the work regarding parallelization of VoronoiLSH using dataflow programming.

3.1 VoronoiLSH

We propose a novel method for locality-sensitive hashing in the metric search framework and compare them with other similar methods in the literature. Our method is rooted on the idea of partitioning the data space with a distance-based clustering algorithm (K-medoids) as an initial quantization step and constructing a hash table with the quantization results. *Voronoi LSH* follows a direct approach taken from [Paulevé et al., 2010]: each partition cell is a hash table bucket, therefore the hashing is the index number of the partition cell.

3.1.1 Basic Intuition

Each hash table of Voronoi LSH employs a hash function induced by a Voronoi diagram over the data space. If the space is known to be Euclidean (or at least coordinate) the Voronoi diagram can use as seeds the centroids learned by a Euclidean clustering algorithm, like K-means (in which case Voronoi LSH coincides with K-means LSH). However, if nothing is known about the space,

points from the dataset must be used as seeds, in order to make as few assumptions as possible about the structure of the space. In the latter case, randomly sampled points can be used (in which case Voronoi LSH coincides with DFLSH). However, it is also possible to try select the seeds by employing a distance-based clustering algorithm, like K-medoids, and using the medoids as seeds. Take note that the Voronoi partitioning is implicit and only the centroids of the partitions and pointers to the points of the dataset should be stored, which is needed in order to avoid an exponential space usage.

Neither K-means nor K-medoids are guaranteed to converge to a global optimum, due to dependencies on the initialization. For Voronoi LSH, however, obtaining *the* best clustering is not a necessity. Moreover, when it is used with several hash tables, the differences between runs of the clustering are employed in its favor, as one of the sources of diversity among the hash tables (Figure 3.1). We further explore the problem of optimizing clustering initialization below.

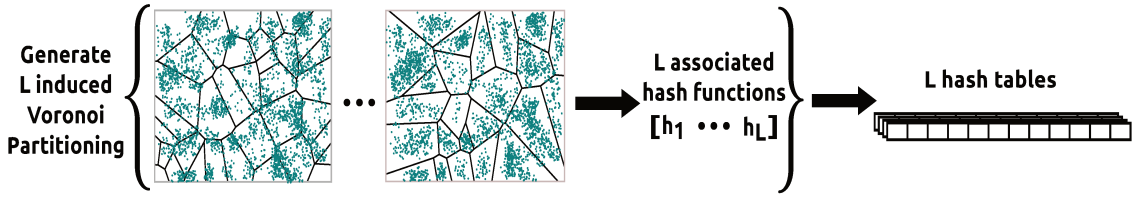


Figure 3.1: Each hash table of Voronoi LSH employs a hash function induced by a Voronoi diagram over the data space. Differences between the diagrams due to the data sample used and the initialization seeds employed allow for diversity among the hash functions.

PAM (Partitioning Around Medoids) [Kaufman and Rousseeuw, 1990] is the basic algorithm for K-medoids clustering. A medoid is defined as the most centrally located object within a cluster. The algorithm initially selects a group of medoids at random or using some heuristics, then iteratively assigns each non-medoid point to the nearest medoid and update the medoids set looking for the optimal set of points that minimize the quadratic sum of distance $f(C)$, with $C = \{c_1, \ldots, c_k\}$ being the set of centroids (Equation 3.1, refer to Definition 2.7 for $d(x, C)$).

$$f(C) = \sum_{x \in X \setminus C} d(x, C)^2 \tag{3.1}$$

PAM features a prohibitive computational cost, since it performs $O(k(n-k)^2)$ distance calculation for each iteration. There are a few methods designed to cope with those complexities, such as CLARA [Kaufman and Rousseeuw, 1990], CLARANS [Ng, 2002], CLATIN [Zhang and Couloigner, 2005], the algorithm by Park and Jun [2009] and FAMES [Paterlini et al., 2011]. Park and Jun [2009] proposes a simple approximate algorithm based on PAM with significant performance improvements. Instead of searching the entire dataset for a new optimal medoid, the method restricts the search for points within the cluster. We choose to apply this method for its simplicity as an initial attempt and baseline formulation, further exploration of other metric space clustering algorithm is performed and results are reported.

3.1.2 Algorithms

We define more formally the hash function used by Voronoi LSH in Equation 3.2. In summary, it computes the index of the Voronoi seed closest to a given input object[1]. In addition, we present the indexing and querying phases of Voronoi LSH, respectively, in Algorithms 1 and 2. Indexing consists in creating L lists with k Voronoi seeds each ($C_i = \{c_{i1}, \ldots, c_{ik}\}, \forall i \in \{1, \ldots, L\}$). When using K-medoids, which is expensive, we suggest the Park and Jun fast K-medoids algorithm (apud [Paulevé et al., 2010]), performing this clustering over a sample of the dataset, and limiting the number of iterations to 30. Then, for each point in the dataset, the index stores a reference in the hash table T_i ($i \in \{1, \ldots, L\}$), using the hashing function defined in Equation 3.2 ($h_{C_i}(x)$). When using K-means, we suggest the K-means++ variation [Ostrovsky et al., 2006; Arthur and Vassilvitskii, 2007]. The seeds can also simply be chosen at random (like in DFLSH). The querying phase is conceptually similar: the same set of L hash functions ($h_{C_i}(x)$) is computed for a query point, and all hash tables are queried to retrieve the references to the candidate answer set. The actual points are retrieved from the dataset using the references, forming a candidate set (shortlist), and the best answers are selected from this shortlist.

In this work, we focus on k-nearest neighbor queries. Therefore, this final step consists of computing the dissimilarity function to all points in the shortlist and selecting the k closest ones.

Definition 3.1. *Given a metric space (X, d) (X is the domain set and d is the distance function), the set of cluster centers $C = \{c_1, \ldots, c_k\} \subset U$ and an object $x \in X$:*

$$\begin{gathered} h_C : U \to \mathbb{N} \\ h_C(x) = \text{argmin}_{i=1,\ldots,k}\{d(x, c_1), \ldots, d(x, c_i), \ldots, d(x, c_k)\} \end{gathered} \tag{3.2}$$

input : Set of points X, number of hash tables L, size of the sample set S and number of Voronoi seeds k
output: list of L index tables $T_1, \ldots, T_L$ populated with all points from X and list of L Voronoi seeds $C_i = \{c_{i1}, \ldots, c_{ik}\}, \forall i \in 1, \ldots, L$

```
for i ← 1 to L do
    Draw sample set S_i from X;
    C_i ← choose k seeds from sample S_i (random, K-means, K-medoids, etc.);
    for x ∈ X do
        T_i[h_{C_i}(x)] ← T_i[h_{C_i}(x)] ∪ {pointer to x};
    end
end
```

Algorithm 1: Voronoi LSH indexing phase: a Voronoi seed list (C_i) is independently selected for each of the L hash tables used. Further, each input data point is stored in the bucket entry ($h_{C_i}(x)$) of each hash table.

[1] Chávez et al. [2001] use to term Compact partition relation to refer to this kind of partitioning of the dataset and demonstrate some lower bounds for algorithms using compact partitions

input : Query point q, index tables $T_1, \ldots, T_L$, L lists of M Voronoi seeds each $C_i = \{c_{i1}, \ldots, c_{ik}\}$, number of nearest neighbors to be retrieved N
output: set of N nearest neighbors $\text{NN}(q, N) = \{n_1, \ldots, n_N\} \subset X$

```
CandidateSet ← ∅ ;
for i ← 1 to L do
    CandidateSet ← CandidateSet ∪ T_i[h_{C_i}(q)] ;
end
NN(q, N) ← {k closest points to q in CandidateSet };
```

Algorithm 2: Voronoi LSH querying phase: the input query point is hashed using same L functions as in indexing phase, and points in colliding bucket in each hash table are used as nearest neighbors candidate set. Finally, N closest points to the query are selected from the candidate set.

Clustering Initialization

K-means and K-medoids clustering results exhibit a strong dependence on the initial cluster selection. Usually those points are selected at random, but there are some proposed heuristics. K-means++ [Ostrovsky et al., 2006; Arthur and Vassilvitskii, 2007] solve the $O(\log k)$ approximate K-means problem[2] simple by carefully designing a good initialization algorithm. That raises the question of how the initialization could affect the final result of the kNN search for the proposed methods. The K-means++ initialization method (Algorithm 3) is based on metric distance information and sampling, thus can be plugged into a K-medoids algorithm without further changes. [Park and Jun, 2009] propose also a special initialization (Algorithm 4) for the fast K-medoids algorithm. We implemented both of those initialization, and the random selection as well and empirically evaluated their impact on the similarity search task.

input : Set of points X and number of cluster centers k
output: list of cluster centers $\{c_1, \ldots, c_k\}$

```
Choose an initial cluster c_1 at random from X;
while total of number of cluster centers < k do
    choose the next center c_i, selecting c_i = x' ∈ X with probability D(x')^2 / Σ_{x∈X} D(x)^2;
end
```

Algorithm 3: k-means++ procedure for initial cluster selection

[2] the exact solution is NP-Hard

input : Set of points $X = \{x_1, \ldots, x_n\}$ and expected number of cluster centers k
output: list of cluster centers $\{c_1, \ldots, c_k\}$
Calculate distance matrix with $\{d_{ij} = d(x_i, x_j)\}_{n \times n}$;
for $j \leftarrow 1$ **to** n **do**
  $v_j \leftarrow \sum_{i=1}^{n} d_{ij} / \sum_{l=1}^{n} d_{il}$;
end
Sort $\{v_1, \ldots, v_n\}$ in ascending order;
Select the k points with smallest v_j value as initial cluster centers;

Algorithm 4: Park and Jun procedure for initial cluster selection

3.1.3 Cost models and Complexity Aspects

We will consider two models for query cost, a more simplistic looking only to distance computation and another one taking into account other costs. We must remember that there is two basic component of the query cost: the cost to find the appropriate bucket (called *internal cost*) and the cost to sequentially process and rank the points in the bucket (called *external cost*). The basic supposition is that the number of points in a bucket can be approximated by its mean when we evaluate the average query cost.

Given a dataset X of n points and k centroids c_i associated to a bucket X_i (which forms a partition of the dataset, implying that $\cup_{i=1}^{K} X_i = \mathrm{X}$ and $\cap_{i=1}^{K} X_i = \emptyset$), the average number of point in a bucket (**n**) is given by:

$$\mathbf{n} = \frac{\sum_{i=1}^{K} |X_i|}{k} = \frac{n}{k}$$

For the fist model let us consider that the distance function evaluation is computationally much more expensive than any other computation in the algorithm, so the complexity is dominated by the number of distance function evaluations. We denominate this cost model the *Range Search Cost Model* because it does not differentiate between range queries and k-nearest neighbor queries in terms of cost. The internal cost is the number of distance computation in the linear scan for the nearest centroid to the query and is fixed on k. The external cost is the number of distance computations for the final ranking. In the average case we will approximate this cost by the average number of points **n** in each bucket. So the final cost is given by:

$$\mathrm{RC}(n, k) = \frac{n}{k} + k$$

Another possibility is to jointly analyze the final ranking cost, which does not depend solely on the number of distance computations. For that matter, we assume that any *basic* operation (mathematical operators,numbers comparisons, memory transfer) has unit cost and that a given distance has a constant cost of $d > 1$. We will denominate this model as *Nearest-Neighbors Cost Model*. The final ranking operation has the complexity of a sorting algorithm, which is $O(nlogn)$,

over the average number of points in each bucket. Taking all into account the final cost is:

$$\text{NNC}(n,k,d) = \frac{n}{k}\log(\frac{n}{k}) + d\frac{n}{k} + dk$$

Optimizing the parameters: VoronoiLSH has basically two parameters, the number of tables L and number of centroids k. For the sake of simplicity we will look only for the parameter k (supposing a fixed L, and given that this parameter has a fixed multiplicative positive impact over the query time complexity and a non-obvious multiplicative positive impact over the quality of the search).

Let us start with Range Search Model; Note that $1 \leq k \leq n$ and that in practical setups $k \ll n$.

$$\frac{\partial \text{RC}(n,k)}{\partial k} = -\frac{n}{k^2} + 1 = 0$$

So the optimal number of centroids k is

$$\Rightarrow k = \sqrt{n}$$

$$\Rightarrow \text{RC}(n) = 2\sqrt{n}$$

For the Nearest-Neighbor Model we proceed in the same way, using the same assumptions:

$$\frac{\partial \text{NNC}(n,k,d)}{\partial k} = -\frac{n}{k^2}[\log(\frac{n}{k}) + 1] - d\frac{n}{k^2} + d = 0$$

$$\Rightarrow \frac{n}{k^2}[\log(\frac{n}{k}) + 1 + d] = d$$

So the optimal number of centroids k is implicit given by

$$k^2 = \frac{n}{d}[\log(\frac{n}{k}) + 1 + d]$$

Knowing that $log(n/k) > 0$ we conclude that

$$k^2 > \frac{n}{d}(d+1)$$

$$\Rightarrow \frac{n^2}{k^2} < \frac{d}{d+1}n$$

$$\Rightarrow \log(\frac{n}{k}) < \frac{1}{2}\log(\frac{d}{d+1}n)$$

$$\Rightarrow \frac{n}{d}(d+1) < k^2 = \frac{n}{d}[\log(\frac{n}{k}) + 1 + d] < \frac{n}{d}[\frac{1}{2}\log(\frac{d}{d+1}n) + 1 + d]$$

$$\Rightarrow \sqrt{n(\frac{1}{d}+1)} < k < \sqrt{n[\frac{1}{2d}\log(\frac{d}{d+1}n) + \frac{1}{d} + 1]}$$

$$\Rightarrow d\sqrt{n(\frac{1}{d}+1)} < dk < d\sqrt{n[\frac{1}{2d}\log(\frac{d}{d+1}n)+\frac{1}{d}+1]}$$

$$\Rightarrow \sqrt{n(d+1)} < dk < \sqrt{n[\frac{1}{2}\log(\frac{d}{d+1}n)+d+1]}$$

$$\Rightarrow 1/\sqrt{n[\frac{1}{2d}\log(\frac{d}{d+1}n)+\frac{1}{d}+1]} < 1/k < 1/\sqrt{n(\frac{1}{d}+1)}$$

$$\Rightarrow \frac{n}{\sqrt{n[\frac{1}{2d}\log(\frac{d}{d+1}n)+\frac{1}{d}+1]}} < n/k < \frac{n}{\sqrt{n(\frac{1}{d}+1)}}$$

$$\Rightarrow \sqrt{\frac{dn}{\frac{1}{2}\log(\frac{d}{d+1}n)+d+1}} < n/k < \sqrt{\frac{dn}{d+1}}$$

Knowing that the optimal k must satisfy

$$kd = \frac{n}{k}\log\frac{n}{k} + d\frac{n}{k} + \frac{n}{k}$$
$$\Rightarrow kd - \frac{n}{k} = \frac{n}{k}\log\frac{n}{k} + d\frac{n}{k}$$

We conclude that the optimal cost $\text{NNC}_{opt}(n,d) = \text{NNC}(n,k,d)$ may be simplified to

$$\text{NNC}_{opt}(n,d) = 2dk + (d-1)\frac{n}{k}$$

Implying that

$$\sqrt{nd}[2\sqrt{d+1}+\sqrt{\frac{(d-1)^2}{\log(\sqrt{\frac{dn}{d+1}})+d+1}}] < \text{NNC}_{opt}(n,d) < \sqrt{nd}[2\sqrt{\log(\sqrt{\frac{dn}{d+1}})+d+1}+\sqrt{\frac{(d-1)^2}{d+1}}]$$

$$\Rightarrow \text{NNC}_{opt}(n,d) = O(\sqrt{nd(\log(\sqrt{n})+d+1)} = O(d\sqrt{n(\log(\sqrt{n})+\frac{1}{d}+1)})$$

In the Nearest-Neighbor cost model the dimensionality information is built-in the distance cost d, for instance, if we consider L_p distance in a D dimensional Euclidean space, the distance cost d may be approximated by dimensionality D. It is interesting to note that, although we did not achieve a logarithm query cost, we obtained a sub-linear cost on the number of points (and linear on the distance cost) which does not hide exponential costs on the dimensionality. Comparing to standard LSH in Euclidean space which solves the (c,r)-approximate nearest-neighbors with $O(dn^{\rho})$ query complexity, where d is the dimensionality (and the approximate cost of evaluating the distance function) and ρ is a fractional power (depending on the approximation factor c) [Indyk and Motwani, 1998; Andoni and Indyk, 2006; Datar et al., 2004], we may conclude that the expected query complexity (under reasonable assumptions) are very close to the Euclidean LSH. However

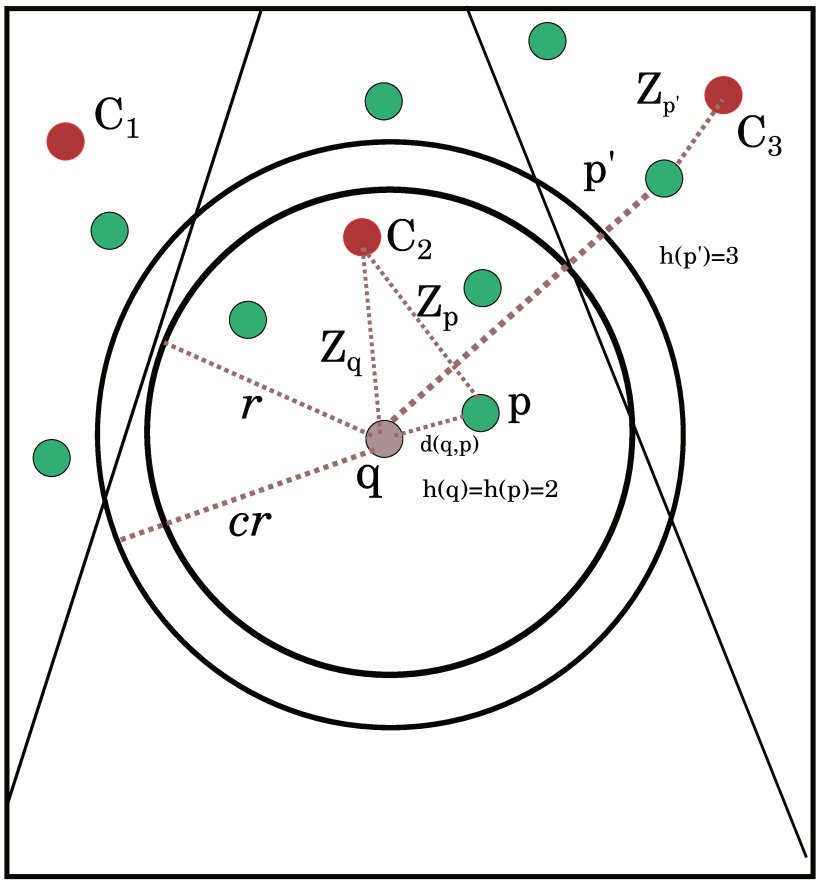


Figure 3.2: Voronoi LSH: a toy example with points $C = \{c_1, c_2, c_3\}$ as centroids, query point p, relevant nearest neighbor q, irrelevant nearest neighbor p', radii of search r and cr and random variables Z_q, Z_p and $Z_{p'}$

the approach can be used to a broader set of distance functions, without the need of embedding the data in Euclidean space, which could add up more complexity and distortions in the method.

Regarding space complexity it is straightforward: the algorithms stored L copies of a hash-table with k entries, each entry with pointers to points of the dataset. The number of pointers is Ln and the number of table entries is Lk, with a total of $L(n + k)$.

3.1.4 Hashing Probabilities Bounds

Now we proceed to analyze the hashing collision probabilities and verifying the locality-sensitiveness of our proposed scheme. We consider the metric space (X, d) and a collection of random points of size k, $C = \{c_1, \ldots, c_k\} \subset X$ inducing a random Voronoi partitioning of the dataset and distinct hash functions. Our hashing scheme consists of attributing an integer code i to each point in the region of the space covered by some point $c_i \in C$ (defined by the subset of points closest to C, $X_i = \{x | d(x, C) = \inf_{c \in C} d(x, c), x \in X\}$). Alternatively, given a point $p \in X$, the notation $\text{NN}_C(q)$ represents the nearest-neighbor of p in the random set C. Putting all together, given the random set C, a point $p \in X$ is associated with the random variable

$Z_p = d(p, \text{NN}_C(p)) = d(p, C)$, and if $\text{NN}_C(p) = c_i$ then $h_C(p) = i$. We want to analyze the probabilities of points being assigned the same hash value (attributed to the same Voronoi region) looking at the distance of the points for different possible hash functions.

Because we are dealing with generic metric spaces without any specific structural information, we shall recall to the tools and language of probability measure spaces. Our probability measure space is (Ω, F, Pr), where $\Omega = \{d(x, \text{NN}_C(x)) | x \in X, C \subset X\} \subseteq [0, M]$ is the sample space, consisting in all possible distances from points in X to subsets of X, F is a Borel σ-algebra over Ω (collection of subset of Ω closed on the operations of complement and disjoint union), which constitutes the event space, and Pr is a probability measure over the σ-algebra, which assigns a probability to each event. As an abuse of notation, we shall use inequalities (or other relations) with the random variables in Ω representing elements of F, for example, $Pr[Z < r]$, $Z \in \Omega$, is the probability measure associated with the event (a set $A \in F$) where the relation is true. Similarly we shall use a fashion of comprehensive set notation with inequalities (or other relations) and random variables as a representation of events (sets in F) where that relation is true, for example, $\{Z < r\}$ represents the event where a random variable Z is less than a fixed value r. Taking both notations in consideration and the same example, $Pr[Z < r]$ and $Pr[\{Z < r\}]$ represents the same probability, which is the measure of the event $\{Z < r\} \in F$.

The analysis will be developed supposing the framework of (R, c)-Nearest neighbors problem and the random variables associated of the distance distribution. We will demonstrate that it is possible to upper and lower bound the hashing probabilities using distance distribution information, and under certain conditions, this may be shown to be locality-sensitive.

Theorem 3.1. *The family of hash function $\{h_C | h_C : X \to \mathbb{Z}^+$, with $C \subset X$ and $|C| = k\}$ over metric space (X, d), mapping a point to the index of the correspondent Voronoi region induced by C, is locality-sensitive for every pairs of points $p, q \in X$ and some fixed scalar $r > 0, c > 3$. In other words:*

- *If $d(p, q) \leq r$ then $Pr[h_C(q) = h_C(p)] \geq p_1 = Pr[|Z_q - Z_p| \geq r]$ (probability of colliding within the ball of radius r),*
- *If $d(p, q) > cr$ then $Pr[h_C(q) = h_C(p)] \leq p_2 = Pr[Z_q + Z_p \geq cr]$ (probability of colliding outside the ball of radius cr)*
- *$p_1 > p_2$, which is equivalent to $Pr[|Z_q - Z_p| \geq r] > Pr[Z_q + Z_p \geq cr]$*

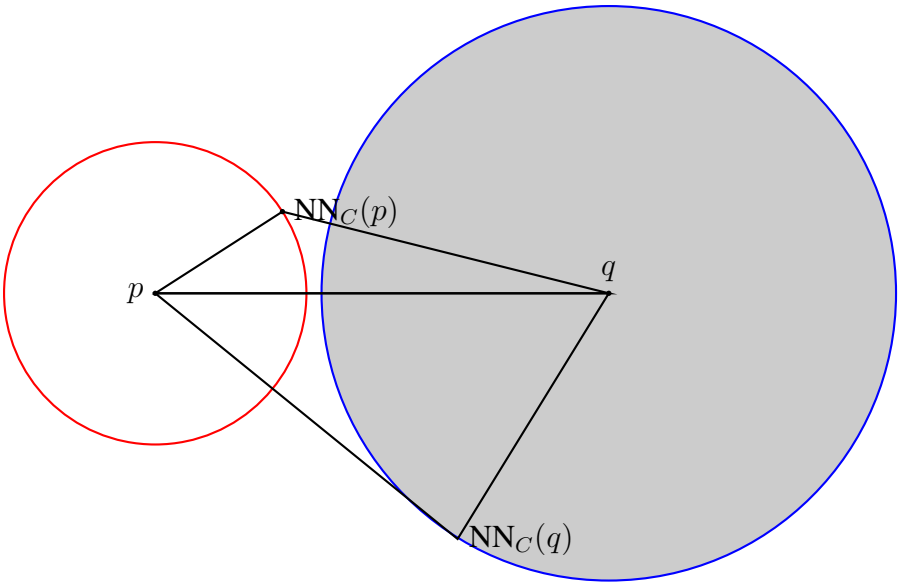


Figure 3.3: Closed Balls centered at point q and p, illustrating the case where events with $Z_q + Z_p \leq d(p,q)$, which implies that $\text{NN}_C(p) \neq \text{NN}_c(q)$

Proof. Let us take two points, p and q, and look at the value of the random variables representing the distance from a point to its nearest neighbor among the set C, Z_p and Z_q, and how their outcomes might affect $h_C(p)$ and $h_C(q)$. We shall analyze the hashing probabilities in the case where $d(p,q) \leq r$, $d(p,q) > cr$ and finally show how those probabilities are related.

Considering the case where $d(p,q) \leq r$, by triangle inequality we have:

$$d(p, \text{NN}_C(q)) \leq d(p,q) + d(q, \text{NN}_C(q)) = d(p,q) + Z_q \leq r + Z_q$$

$$d(q, \text{NN}_C(p)) \leq d(p,q) + d(p, \text{NN}_C(p)) = d(p,q) + Z_p \leq r + Z_p$$

The probabilities of $h_C(p) \neq h_C(q)$ is modeled as:

$$Pr[h_C(p) \neq h_C(q)] = Pr[\{d(q, \text{NN}_C(q)) < d(q, \text{NN}_C(p))\}, d(p, \text{NN}_C(p)) < d(p, \text{NN}_C(q))]$$

$$= Pr[Z_q < d(q, \text{NN}_C(p)), Z_p \leq d(p, \text{NN}_C(q))]$$

$$\leq Pr[Z_q < r + Z_p, Z_p < r + Z_q] = Pr[|Z_q - Z_p| < r]$$

$$\Rightarrow Pr[h_C(p) \neq h_C(q)] \leq Pr[|Z_q - Z_p| < r]$$

Since $Pr[h_C(p) = h_C(q)] = 1 - Pr[h_C(p) \neq h_C(q)]$ we finally obtain:

$$Pr[h_C(p) = h_C(q)] \geq Pr[|Z_q - Z_p| \geq r] = p_1$$

Consider the case where $d(p,q) > cr$. We define the events of interest $\xi_0 = \{Z_q + Z_p < cr\}$, $\xi_1 = \{Z_q + Z_p < d(p,q)\}$ and $\xi_2 = \{Z_q < d(q, \text{NN}_C(p)\} \cap \{Z_p < d(p, \text{NN}_C(q)\}$.

From the definitions and the hypothesis that $cr < d(p,q)$:

$$Z_p + Z_q < cr \Rightarrow Z_p + Z_q < d(p,q)$$

$$\Rightarrow \xi_0 \subseteq \xi_1$$

Applying triangle inequality we obtain in conjunction with $Z_p + Z_q < d(p,q)$:

$$\begin{cases} Z_q + Z_p < d(p,q) \leq d(p, \text{NN}_C(p)) + d(q, \text{NN}_C(p)) = Z_p + d(q, \text{NN}_C(p)) \\ Z_q + Z_p < d(p,q) \leq d(q, \text{NN}_C(q)) + d(p, \text{NN}_C(q)) = Z_q + d(p, \text{NN}_C(q)) \end{cases}$$

$$\Rightarrow \begin{cases} Z_q < d(q, \text{NN}_C(p)) \\ Z_p < d(p, \text{NN}_C(q)) \end{cases}$$

$$\Rightarrow \xi_1 \subseteq \xi_2$$

$$\Rightarrow \xi_0 \subseteq \xi_1 \subseteq \xi_2$$

Knowing that the event ξ_2 represents the situation where $h_C(p) \neq h_C(q)$ we obtain a lower bound for the hashing probabilities:

$$Pr[h_C(p) \neq h_C(q)] = Pr[\xi_2] \geq Pr[\xi_0] = Pr[Z_q + Z_p < cr]$$
$$\Rightarrow Pr[h_C(p) = h_C(q)] \leq Pr[Z_q + Z_p \geq cr] = p_2$$

To show that $p_1 > p_2$ we have only a proof sketch that works under strict conditions. Start with the complement event $\xi_0^C = \{Z_q + Z_p \geq cr\}$. Notice that $cr > r$ and assume that $Z_q < \delta r$ (for some $\delta > 1$), meaning that the range r of search can not be less than a fixed factor of to the closest cluster center (a limiting condition that does not hold in general). Putting all together:

$$Z_q + Z_p \geq cr$$

$$\Rightarrow Z_q + Z_p - 2Z_q \geq cr - 2Z_q$$

Since $Z_q < \delta r$, we know that $-Z_q > -\delta r$

$$\Rightarrow Z_q + Z_p - 2Z_q = Z_p - Z_q \geq (c - 2\delta)r$$

Choosing $c > 2\delta + 1$ we obtain

$$\Rightarrow Z_p - Z_q \geq r$$
$$\Rightarrow \{Z_q + Z_p \geq cr\} \subseteq \{|Z_q - Z_p| \geq r\}$$
$$\Rightarrow p_2 \leq p_1$$

In order to finish the sketch-proof, consider the hypothetical case where $Z_q = r - \epsilon$ and $Z_p = 2\delta r - \epsilon$,

for an arbitrary $\epsilon > 0$ (and maintaining the previous assumptions on δ and c). Taking $Z_p - Z_q$, we obtain $2\delta r - \epsilon - r + \epsilon = (2\delta - 1)r$, meaning that $|Z_q - Z_p| > r$ (Z_p and Z_q are elements of the event $\{|Z_q - Z_p| \geq r\}$). Computing $Z_p + Z_q = 2\delta r - \epsilon + r - \epsilon$, we obtain $(2\delta + 1)r - 2\epsilon$, implying that $Z_q + Z_p < (2\delta + 1)r$; Since $(2\delta + 1) < c$ (an assumption from the previous step of the proof), we obtain $Z_q + Z_p < cr$, and conclude that $\{Z_q + Z_p \geq cr\}$ is a proper subset of $\{|Z_q - Z_p| \geq r\}$ under the conditions we are analyzing.

$$\Rightarrow p_2 < p_1$$

□

This is a generic result depending on some assumptions of the metric structure of the space and a somehow reasonable workload of the (r, c)-ANN, but does not elucidate relationships between the choice of variable (namely k and L) and the probabilistic guarantees of the method. Further development in this line of argument is possible by observing that the distribution of the random variables Z_q and Z_p may be understood as an order statistics over the distribution of distances in the dataset. Indeed if we define the (unordered) sequence of random variable $Z_{q(1)}, \cdots, Z_{q(k)}$ as distances from $q \in X$ to a each point $\{c_1, \cdots, c_k\}$ from a random sample $C \subseteq X$ ($|C| = k$), the random variable Z_q correspond to the order statistics $\min(Z_{q(1)}, \cdots, Z_{q(k)})$. We may compute the cumulative distribution function (cdf) $F_{\min}$ of the minimum order statistics using the hypothesis that the random variables $Z_{q(1)}, \cdots, Z_{q(k)}$ are i.i.d. (a not unrealistic one, given that the sample C is randomly chosen from the original dataset) and are distributed according to a cdf F :

$$F_{\min}(x) = P(Z_q \leq x) = P(\min(Z_{q(1)}, \cdots, Z_{q(k)}) \leq x) = 1 - P(\min(Z_{q(1)}, \cdots, Z_{q(k)}) > x)$$

$$= 1 - P(Z_{q(1)} > x, \cdots, Z_{q(k)} > x) = 1 - \prod_{i=1}^{i=k} P(Z_{q(i)} > x) = 1 - \prod_{i=1}^{i=k} (1 - F(x))$$

$$\Rightarrow F_{\min}(x) = 1 - (1 - F(x))^k$$

The interesting fact about this relation is that somehow, without using the "concatenation" trick (building a composite LSH hash using the concatenation of multiple LSH, implying in the multiplication of the hashing probabilities), we achieved a similar *exponential* probability for the nearest-neighbor distance probability from the distance distribution.

3.2 VoronoiPlex LSH

VoronoiLSH was our first attempt to develop a LSH family for general metric space, seeking good trade-off between effectiveness, efficiency and generalization. We took a very simplistic design, and as we have presented in the theoretical characterization of the method, despite the minimalistic approach (in relation the other methods that directly apply various metric space properties for filtering and pruning) the method achieves competitive average query time performance and can be

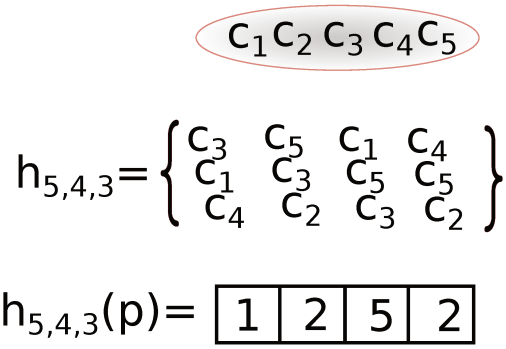


Figure 3.4: VoronoiPlex LSH: a toy example with points $C = \{c_1, c_2, c_3, c_4, c_5\}$ as shared centroids between partitioning, choosing three centroids for each partitioning and concatenating four partitioning in the final hash-value

formally described in the locality-sensitive framework – meaning, there are provable probabilistic bounds for the approximate nearest-neighbors search. However, given that the only parameter available for tuning the metric by constraining the size of the partitions is the number of cluster centers, and that the relation between the number of cluster centers and the method potential of selecting the most relevant nearest-neighbor is not trivial, an extension of the method was developed taking advantage of the locality-sensitiveness of the hashing family and the same fashion of Indyk and Motwani [Indyk and Motwani, 1998] – boosting the probabilities of "good" hashing collisions more than the "bad" ones, and minimizing the number of distance computations necessary for multiples VoronoiLSH.

3.2.1 Basic Intuition

The basic idea is the boost the probability of finding true nearest-neighbors in the same bin more than the probability of finding true nearest-neighbors in distinct bins. This objective may be achieved by the application of multiples random partitioning of the dataset (meaning, in our framework, multiple VoronoiLSH) and hashing a way that only points that share the same nearest-neighbor in all partitioning share the same hash number. We would expect that, under these random perturbations on the partitioning, most of points that share all the same partitions are very similar and that most false positive similar points are being filtered out. An another alternate view of this approach is to see it as a two levels partitioning: the first level is of direct Voronoi partitioning using random points as centroids, this operation is repeated many times; the second level is a unique partitioning that is built by combining all the multiple Voronoi partitioning in a way that points that share the same code in all of them are hashed together.

We have four control parameters in this approach: the number of hash tables (L), the number of centroids for each partitioning (p), the number of centroids shared between partitioning (k), and the number of multiple partitioning that are later put together (w). The number of hash tables have a positive impact over the overall quality of the approximate nearest-neighbor search, however isolated it does not improve the selectivity of the method, increasing the number false positives.

The other three parameters have a non-trivial impact in selectivity and quality of the search that we still need to further investigate. In the following sections, we will present a brief theoretical characterization of this impact. Figure 3.4 presents an example with five shared centroids ($k = 5$), four partitioning concatenated ($w = 4$) and three centroids in each partitioning ($p = 3$).

3.2.2 Algorithms

The algorithms for indexing and kNN searching are essentially the same described in Algorithm 1 and Algorithm 2, with the exception of the hash function. Hence, we will present the algorithm for the construction of the hash function and for the application of the hashing to a point in the metric space. We will assume that functions/methods are first class objects with the possibility of being returned from a method, encapsulating data and computation.

Algorithm 5 performs a multiple selection of index and stores the selected indexes in a data structure that represents the hashing function. The indexes selected are used as sub-samples of a given sample of the dataset, and each group of selected indexes forms a set of centroid points that spans a Voronoi partitioning of the space. Those k partitioning forms a product space of hash-vectors of size k. The intuitive idea is to build up on the VoronoiLSH intuition, using the compact closest center relation as a hashing function, and with a bounded number of distance computation, construct a product space of hash function. The idea is to boost the single hash probabilities with a controlled number of distance of computation.

input : Integer size k of the sample $C = \{c_1, \ldots, c_k\} \subset X$, integer number of distinct partitioning w, and integer number of centroids for each partitioning $p < k$
output: A hash function $h_{k,w,p}$ object with two arrays indicating the random choice of centroids made by the method

```
1  selected ← new binary array of size k;
2  subsample ← new integer multi-array of size w × p;
3  for j ← 1 to w do
4  |   Random sample S = {s_1, ··· , s_p} from {1, ··· , k};
5  |   for i ← 1 to p do
6  |   |   subsample[j, i] ← s_i;
7  |   |   selected[s_i] ← 1;
8  |   end
9  end
10 h_{k,w,p} ← (selected,subsample) ;
```

Algorithm 5: Hash function building

input : Hash function object $h_{k,w,p}$,Sample $C = \{c_1, \ldots, c_k\} \subset X$ ($|C| = k$) and a point $q \in X$
output: Integer value $h_{k,w,p}(q)$

(selected,subsample) $\leftarrow$ retrieved from $h_{k,w,p}$ distances $\leftarrow$ new floating-point array of size k;
for $j \leftarrow 1$ **to** k **do**
  if *selected*$[j] == 1$ **then**
    distances$[j] \leftarrow d(q, c_j)$;
  end
end
hasharray $\leftarrow$ new integer array of size w;
for $i \leftarrow 1$ **to** w **do**
  hasharray$[i] \leftarrow$ element in subsample$[i]$ that minimize distances$[j]$ (varying j) ;
end
$h_{k,w,p}(q) \leftarrow$ hash(hasharray) ;

Algorithm 6: Hash function Application

3.2.3 Theoretical characterization

First, we must note that the hash function build procedure (Algorithm 5) is blind to the actual data. It is a random selection of indexes of the arrays that store the sampled points. So in first sight we can see that it could be precomputed, or at least re-used in various distinct settings of the algorithm (for example, in a distributed memory system, there could be strong re-utilization of the hash functions). The algorithm consist of two "for" loops and a simple random sample in a set of k elements, resulting in a time complexity of $O(w(p + k))$ and a space complexity of $O(k + wp)$ (with size k of the sample $C = \{c_1, \ldots, c_k\} \subset X$, integer number of distinct partitioning w, and integer number of centroids for each partitioning $p < k$).

For the time complexity we will continue with the focus on expensive computation bottlenecks – distance computation. The discussion will be focused on the expected number of positions in the *selected* array that are set to one – $\mathbf{E}_{i=1,\cdots,k}[\text{selected}[i] = 1]$.

Proposition 3.1. *After w rounds of the outer "for" loop in Algorithm 5, the expected number of position of the array selected set to one is* $\mathbf{E}_{i=1,\cdots,k}[\textit{selected}[i] = 1] = k - k(1 - \frac{p}{k})^w$.

Proof. Start with the probability of a specific position being assigned 1 in a single iteration of the for loop. There are $\binom{k}{p}$ possible ways of sampling p values from k. There are $\binom{k-1}{p}$ ways of sampling p values of k excluding some value of the set with size k. So, the number of ways of

sampling p and not assigning selected$[i]$ one of them is $\binom{k-1}{p}$. Putting all together:

$$Pr_{\text{single iteration}}[\text{selected}[i] = 1] = 1 - \frac{\binom{k-1}{p}}{\binom{k}{p}} = \frac{p}{k}$$

After w iteration, the probability of the position i not being assigned to 1 is multiplication of the probabilities of not being assigned at each iteration.

$$Pr_{\text{after } w \text{ iteration}}[\text{selected}[i] \neq 1] = \prod_{n=1}^{w} Pr_{\text{single iteration}}[\text{selected}[i] \neq 1]$$

$$Pr_{\text{after } w \text{ iteration}}[\text{selected}[i] \neq 1] = (1 - \frac{p}{k})^w$$

$$\Rightarrow Pr_{\text{after } w \text{ iteration}}[\text{selected}[i] = 1] = 1 - (1 - \frac{p}{k})^w$$

Using the probability of being assigned after w iterations of the for loop, we calculate the final expectation of the number of elements in the array selected that are assigned to 1.

$$\mathbf{E}_{i=1,\cdots,k}[\text{selected}[i] = 1] = kPr_{\text{after } w \text{ iteration}}[\text{selected}[i] = 1]$$

$$\mathbf{E}_{i=1,\cdots,k}[\text{selected}[i] = 1] = k - k(1 - \frac{p}{k})^w$$

□

This result is important because it helps us understand how the number of distance computation (and the internal cost) of the hashing algorithm is bounded using the parameters that the hash function is build and taking into account the randomness of the building algorithm. So in general the internal cost, looking only for the number of distance computation is $O(k - k\epsilon)$ where this ϵ factor is well related with the effectiveness of the search.

3.3 Parallel VoronoiLSH

The design and adaptation of the original sequential algorithm for a distributed and parallel framework was developed through a collaborative work of the author, Prof. Eduardo Valle, Prof. George Teodoro and Thiago Teixeira. George Teodoro and Thiago Teixeira work is specialized in distributed systems and they developed substantial part of the parallelization, specially the implementation using dataflow programming paradigm and performance test in a distributed cluster. This collaboration resulted in an conference paper accepted at SISAP2014.

The parallelization strategy we employ is based on the dataflow programming paradigm [Arpaci-Dusseau et al., 1999; Beynon et al., 2001; Teodoro et al., 2008]. Dataflow applications are typically

represented as a set of computing *stages*, which are connected to each other using directed *streams*.

Our parallelization decomposes Voronoi LSH into five computing stages organized into two conceptual pipelines, which execute the index building and the search phases of the application. All stages may be replicated in the computing environment to create as many copies as necessary. Additionally, the streams connecting the application stages implement a special type of communication policy referred here as *labeled-stream*. Messages sent through a labeled-stream have an associated label or tag, which provides an affordable scheme to map message tags to specific copies of the receiver stage in a stream. We rely on this communication policy to partition the input dataset and to perform parallel reduction of partial results computed during a query execution. The data communication streams and processes management are built on top of Message Passing Interface (MPI).

The index building phase of the application, which includes the Input Reader (IR), Bucket Index (BI), and Data Points (DP) stages, is responsible for reading input data objects and building the distributed LSH indexes that are managed by the BI and the DP stages. In this phase, the input data objects are read in parallel using multiple IR stage copies and are sent (1) to be stored into the DP stage (message i) and (2) to be indexed by the BI stage (message ii). First, each object read is mapped to a specific DP copy, meaning that there is no replication of input data objects. The mapping of objects to DPs is carried out using the data distribution function *obj_map* (labeled-stream mapping function), which calculates the specific copy of the DP stage that should store an object as it is sent through the stream connecting IR and DP. Further, the pair <object identifier, DP copy in which it is stored> is sent to every BI copy holding buckets into which the object was hashed. The distribution of buckets among BI stage copies is carried out using another mapping function: *bucket_map*, which is calculated based on the bucket value/key. Again, there is no replication of buckets among BIs and each bucket value is stored into a single BI copy. The *obj_map* and *bucket_map* functions used in our implementation are modulo operation based on the number of copies of the receiver in a stream. We plan to evaluate other hashing strategies in the future.

The index construction is very compute-intensive, and involves many distance calculations between the input data objects and the Voronoi seeds. For Euclidean data, we implemented a vectorized code using Intel SSE/AVX intrinsics to take advantage of the wide SIMD (Single Instruction, Multiple Data) instructions of current processors. Preliminary measurements have shown that the use of SIMD instructions speeds-up the index building 8 times.

The search phase of the parallel LSH uses four stages, two of them shared with the index building phase: Query Receiver (QR), Bucket Index (BI), Data Points (DP), and Aggregator (AG). The QR stage reads the query objects and calculates the bucket values into which the query is hashed for the L hash tables. Each bucket value computed for a query is mapped to a BI copy using the *bucket_map* function. The query is then sent to those BI stage copies that store at least one bucket of interest (message iii). Each BI copy that receives a query message visits the buckets of interest, retrieves the identifier of the objects stored on those buckets, aggregates all object identifiers to be sent to the same DP copy (list(obj_id)), and sends a single message to each DP stage that stores at least one of the retrieved objects (message iv). For each message received by a DP copy, it calculates the distance from the query to the objects of interest, selects the k-nearest

neighbor objects to the query, and sends those local NN objects to the AG stage. Finally, the AG stage receives the message containing the DPs local NN objects from all DPs involved in that query computation and performs a reduction operation to compute the global NN objects. The DP copies (message v) use the query_id as a label to the message, guaranteeing that the same AG copy will process all messages related to a specific query. As a consequence, multiple AG copies may be created to execute different queries in parallel. Although we have presented the index building and the search as sequential phases for sake of simplicity, their executions may overlap.

The parallelization approach we have proposed exploits task, pipeline, replicated and intra-stage parallelism. Task parallelism results from concurrent execution that allows indexing and searching phases to overlap, e.g. during an update of the index. Pipeline parallelism occurs as the search stages, for instance, execute different queries in parallel in a pipeline fashion. Replicated parallelism is available in all stages of the application, which may have an arbitrary number of copies. Finally, intra-stage parallelism results of the application's ability to use multiple cores within a stage copy. This parallelism has the advantages of sharing the same memory space among computing cores in a stage copy, and a reduced number of messages exchanged, since a smaller number of state partitions may be used.

3.4 Final remarks

We have demonstrated that using only metric space properties it is possible to design a LSH family of function over general metric space, using a form of compact partition relation (inducing a partitioning of the space in Dirichlet domains). Our mathematical proof is an original contribution and one of the main contribution of this dissertation. However we should see experimentally how this approach is comparable to other similar methods in the literature. We will accomplish this comparison in the next chapter.

Chapter 4

Experiments

In this chapter the experimental data generated during this research is reported and discussed. This is a recollection of at least four distinct groups of experiments performed in two years of research. Since the development of the algorithms, and specially the parameters of the algorithms was done in a cycle of theories, hypothesis and tests, the experimental data reported here is not homogeneous nor along the datasets, neither along the methods. So for the case of Euclidean data, we compare the performance of our proposed methods with the performance of K-Means LSH. When we are dealing with metric data, K-Means LSH can not be directly applied (only with an additional embedding, which is not the intent of this work), so we compare with another version of LSH for metric space (BPI). In the large scale experiment our intent was not to compare with other methods, but evaluate if the effectiveness and efficiency reported in the sequential small scale setup would scale-up in a distributed framework.

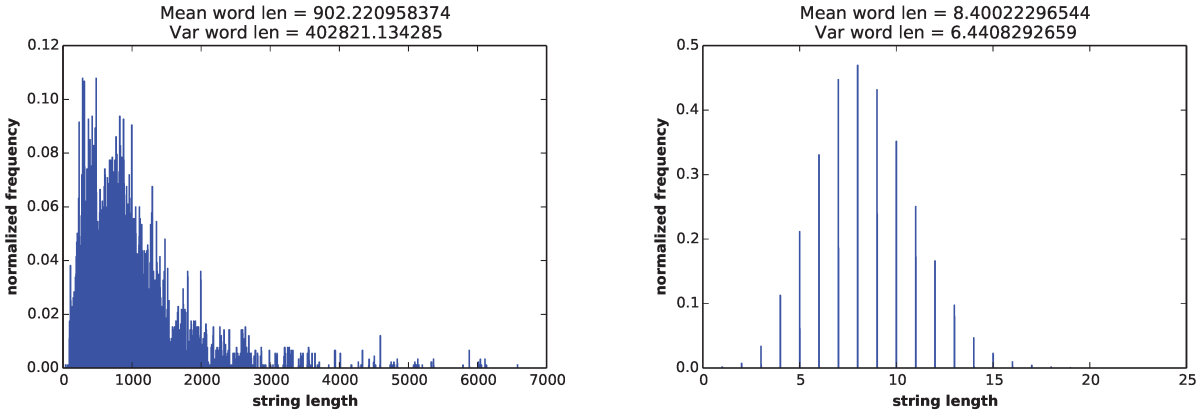
4.1 Datasets

For the experiments regarding Euclidean Spaces and Image Retrieval we resorted to the APM dataset (*Arquivo Público Mineiro* – The Public Archives in Minas Gerais) which was created from the application of various transformation (15 transformation of scale, rotation, etc) to 100 antique photographs of the Public Archives of Minas Gerais. The SIFT descriptors of each transformed image were calculated, resulting in a dataset of 2.871.300 feature vectors (SIFT descriptor is a 128 dimensional vector). The queries dataset is build from the SIFT descriptors of the original images (a total of 263.968 feature vectors). Each query point is equipped with its set of true nearest neighbors – the ground-truth. For these experiments we used 5000 points uniformly sampled from the query dataset and performed a 10-NN search.

In order to evaluate the case of general metric space we resorted to English dictionary of strings and *Listeria* Genes string datasets with Levenshtein distance from SISAP Metric Library [Figueroa et al., 2007]. The English dataset has 69,069 strings, 500 strings are randomly removed from the set to serve as query dataset. In Figure 4.1 there are summarized information and graph regarding string length distribution in those metric datasets. String length distribution affects directly for the

Levenshtein distance calculation (quadratic on the string length).

We used BigAnn [Jegou et al., 2011] dataset for the large scale experiment. The BigAnn contains a thousand million (10^9) 128-dimensional SIFT local feature vectors extracted from images, and 10,000 query feature vectors. Euclidean distance is used for SIFT. We perform kNN searches, with k=5, 10, and 30 for the Dictionary and *Listeria* dataset, and k=10 for the BigAnn dataset.



(a) (b)

Figure 4.1: (a) Listeria Gene Dataset string length distribution (b) English Dataset string length distribution

4.2 Techniques Evaluated

K-means LSH, Voronoi LSH and DFLSH: In order to evaluate the feasibility of Voronoi LSH (using K-Medoids) we compared it with the K-means LSH *et al.* [Paulevé et al., 2010] and the DFLSH [Kang and Jung, 2012]. Although all three methods may be described as Voronoi LSH, but with distinct centroid selection heuristics, the distinction we are making here (between Voronoi, K-Means and DFLSH) serve the purpose of investigating the trade-offs present in the choice of clustering k

We implemented all the methods in the same framework, namely Java and *The Apache Commons™ Mathematics Library*[1]. K-means LSH is used as a baseline method, since its performance and behavior are well studied and presented in the literature and DFLSH is an alternative method for the same problem.

VoronoiLSH and Brief-Permutation Index LSH : In order to allow the comparison with BPI LSH, we followed an experiment protocol as close as possible to the one described in their paper, and used the recall and query runtime reported there. In order to remove the effects of differences in implementation details and execution environment, we normalize both methods runtime by the

[1] *Commons Math: The Apache Commons Mathematics Library*, accessed in 22/09/2013. `http://commons.apache.org/proper/commons-math/`

respective brute-force linear scan runtime. Because the authors report their numbers graphically, we added error bars to indicate the imprecision of *our* reading of their results.

Large-scale experiment : in this case the most important aspect to evaluate is the algorithm speed up in the distributed configuration scaling-up the number of nodes and processors.

4.3 Evaluation Metrics

Given a query point $\mathbf{q}$, we denote $N(\mathbf{q}, k)$ the set of *true* relevant k-nearest neighbors, $A(\mathbf{q})$ the set of candidate nearest neighbors (shortlist) and $R(\mathbf{q}, k)$ the set of k-nearest neighbors returned by the algorithm. We are concerned especially with the relations between the recall and extensiveness metric given the variation in the parametric space of the methods. The *recall* metric is the fraction of total correct answers over the number of true relevant answers (Equation 4.1). The *extensiveness* metric is the fraction of the dataset selected for the shortlist processing (Equation 4.2). The related *selectivity* metric (Equation 4.3) has distinct (and sometimes conflicting) use in the database and image communities.

We employ the *recall* as a metric of quality, and the *extensiveness* as metric of cost for the techniques. The recall is defined as usual for information retrieval, as the fraction of relevant answers that was effectively retrieved. The *extensiveness* metric is the fraction of the dataset selected into the shortlist for linear scan. As the indexes work by selecting a (hopefully small) fraction of the dataset as candidates, and then computing the actual distance to the query for those candidates, the size of the shortlist corresponds to the number of distances computed for that query. The related *selectivity* metric ($selectivity = 1 - extensivity$) has distinct (and sometimes conflicting) use in the database and image retrieval research communities: some authors used it as a synonymous for extensiveness, while others and we use it as a complementary notion (as selectivity grows, extensivity drops). We also employ the *query runtime* as metric of cost. Runtime metrics are difficult to compare across the literature, due to differences in code optimization, programming language, and execution environment, but in controlled experiments like ours they can offer a perspective on the cost of different techniques.

Given a query point $\mathbf{q}$, we denote $N(\mathbf{q}, k)$ the set of *true* relevant k-nearest neighbors, $A(\mathbf{q})$ the set of candidate nearest neighbors (shortlist) and $R(\mathbf{q}, k)$ the set of k-nearest neighbors returned by the algorithm we define the recall, extensivity and selectivity metrics in Equation 4.1, Equation 4.2 and Equation 4.3.

$$recall(q) = \frac{|R(\mathbf{q}, k)|}{|N(\mathbf{q}, k)|} \tag{4.1}$$

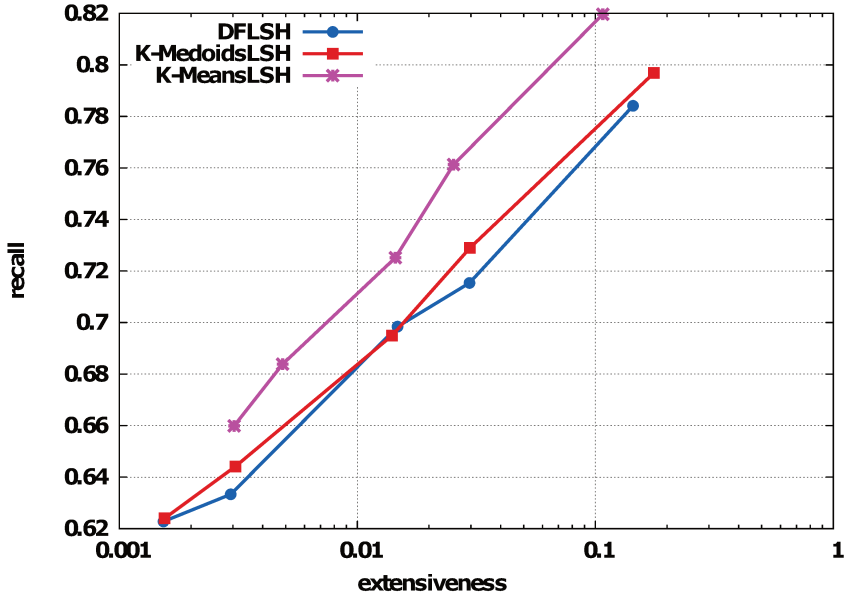
$$extensiveness(q) = \frac{|A(\mathbf{q})|}{n} \tag{4.2}$$

$$selectivity(q) = 1 - \frac{|A(\mathbf{q})|}{n} \tag{4.3}$$

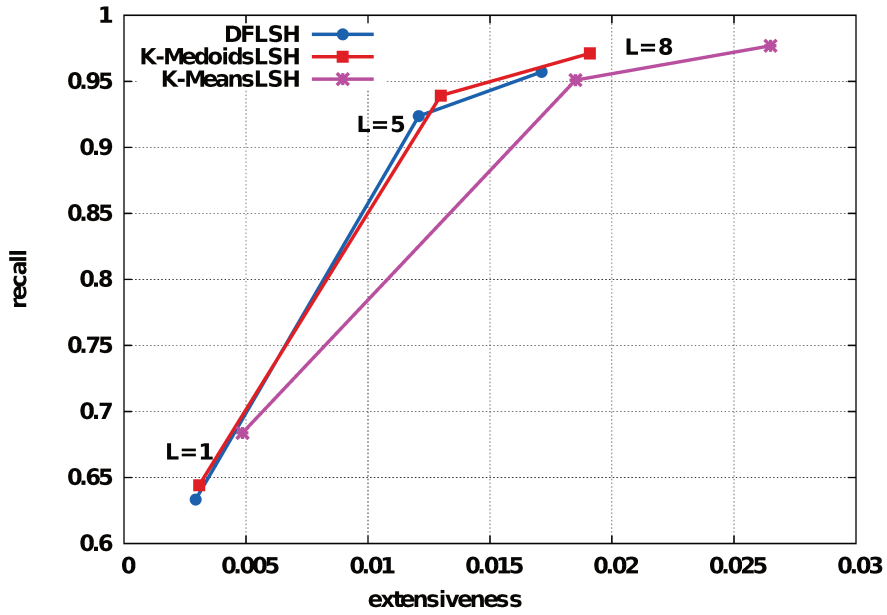
4.4 Results

In the next section we will present our experimental results separated by datasets and methods. In the Section 4.4.1 we will present the results using the APM dataset (Euclidean) which permits us a comparison with K-Means LSH, since K-Means algorithm depends and uses specific coordinate information that are not always available in generic metric spaces. In Section 4.4.2 the metric dataset of words in English is used. Since the same dataset is used for reporting the results of Brief-Permutation Index LSH, we use it for comparison of this alternative formulation of LSH in metric spaces. It was shown in Figure 4.1 that the English dictionary is more easily processed because of the limited and low length of the words, in the Listeria dataset the situation is much more complicated because of the large variability and average lengths of words. We report results of DFLSH and VoronoiPlex-LSH in this settings in Section 4.4.3. Finally in Section 4.4.4 the results of the distributed large scale experiment using BigANN dataset (One billion of points of 124-dimensional euclidean points) are reported.

4.4.1 APM Dataset: comparison with K-Means LSH

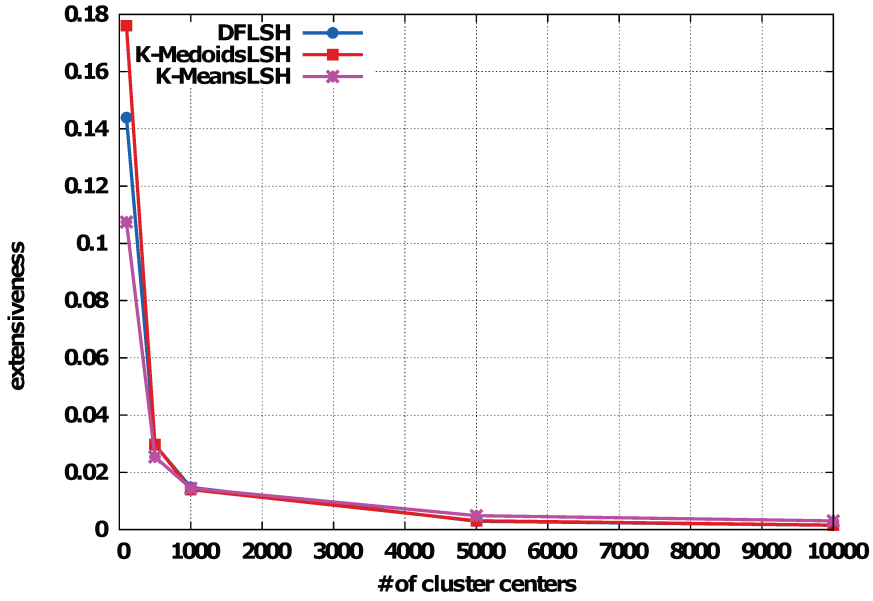


(a) Recall x Extensiveness (log scale)

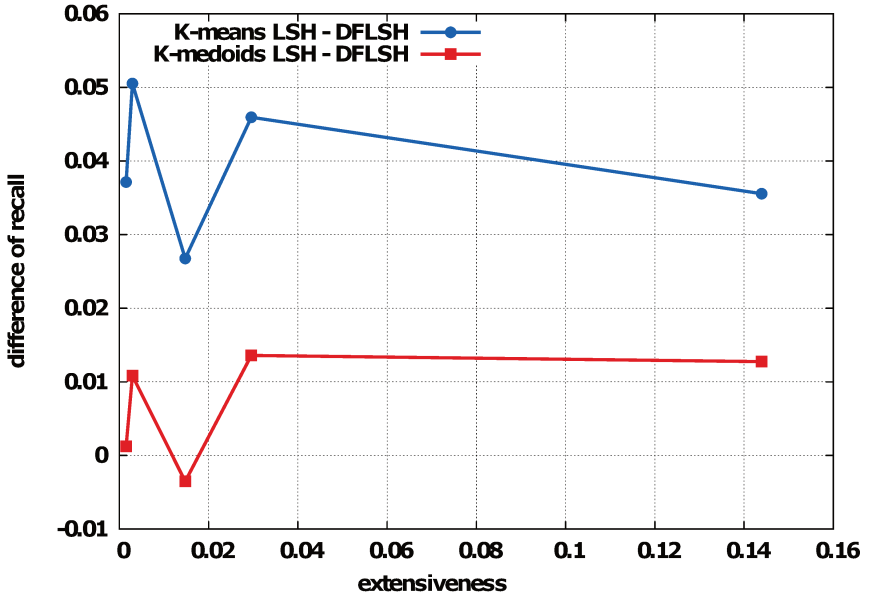


(b) Recall x Number of hash functions L, Extensiveness (for 5000 cluster centers)

Figure 4.2: APM Dataset: Comparison of VoronoiLSH with three centroids selections strategy K-means, K-medoids and Random for 10-NN



(a) Extensiveness x Number of cluster centers



(b) Difference of Recall (Using DFLSH as a baseline) x Extensiveness

Figure 4.3: APM Dataset: Comparison of the impact of the number of cluster center in the extensivity metric and the correlation between extensivity and the recall for K-means LSH, Voronoi LSH and DFLSH (10-NN)

Figure 4.2a presents results relating recall with extensiveness. Notice that high extensiveness means that a large part of the dataset is being processed in the final sequential ranking – that is not a desirable point of operation, since the main performance bottleneck on the query processing time is located at that stage. For an extensiveness as low as 0.3%, both methods' recall metric is approximately 65% and with 1% extensiveness the recall grows up to a 80%. These results could be

improved by using more than one hash function (Figure 4.2b) . Clearly K-means LSH presents the best result on the recall metric, however it is important to notice that both DFLSH and Voronoi LSH are not exploiting any vector coordinate properties, relying solely on distance information, and are therefore more general.

There seems to be a trade-off between number of cluster centers, number of hash functions and selectivity. In Figure 4.2b we see a comparison of recall and extensiveness for a varying number of hash functions. We observe results indicating that in some setups Voronoi LSH and DFLSH may offer a better trade-off between quality of the response and selectivity than K-means LSH. It is not clear what specific feature of those methods may trigger this improvement, but nonetheless it is a very interesting result and may offer some precious intuition on the design of metric indexing methods and choice of parameters.

Although K-means LSH attained the best results with a single hash table, the performance of Voronoi LSH and DFLSH is comparatively better as the number of hash tables used increase (See Figure 4.2b). As shown, in some setups Voronoi LSH and DFLSH may offer a better trade-off between recall and selectivity than K-means LSH. For instance, for a recall of about 95% (L=5) Voronoi LSH and K-means have extensiveness of nearly 0.013 and 0.018, respectively. In other words, our method has to evaluate about 28% less elements in the final ranking phase. A similar level of comparative performance is attained when with a recall of about 97% (L=8). In comparison to DFLSH, our method attained better performance in all setups. The extensiveness results are very important because the final ranking phase represents one of the bottlenecks of searching algorithms.

There is a strong linear correlation between query time and selectivity. Fitting this to a linear model we obtain a very significant and well-adjusted linear model ($R^2 = 0.9985$, p-value $< 2.2 \times 10^{-}15$), with a linear coefficient of 17756.205 for the extensiveness and an intercept coefficient of -2.274. Other noticeable strong correlation appears between selectivity and number of cluster centers (Figure 4.3a). Theoretically it is possible to show that, given an approximate uniform population of points in the hash buckets, the selectivity for a k number of cluster centers is $O(k^{-1})$. The plot shows that the experimental data follows a power law curve. As presented, the extensiveness of Voronoi LSH and DFLSH are higher than that of K-means LSH for small number of Voronoi seeds. However, as the number of Voronoi seeds increase, the extensiveness of Voronoi LSH and DFLSH reduce quickly and these methods outperform K-means LSH for a number of Voronoi seeds higher than 1,000. It is important to highlight that small extensiveness are desirable and methods all methods would attain better performance with Voronoi seed sets with more than 1,000 elements. Therefore, the configuration in which Voronoi LSH and DFLSH are superior correspond to the configuration used in practice.

Using DFLSH as a baseline, we plot the difference on the recall from K-means LSH and Voronoi LSH to DFLSH (Figure 4.3b). The recall difference can be up to +0.05 for K-means LSH and +0.015 for Voronoi LSH. The trend on the differences is sustained along the curves for different values of extensiveness. There is a clear indication that Voronoi LSH is a viable approach with equivalent performance to K-means LSH and DFLSH.

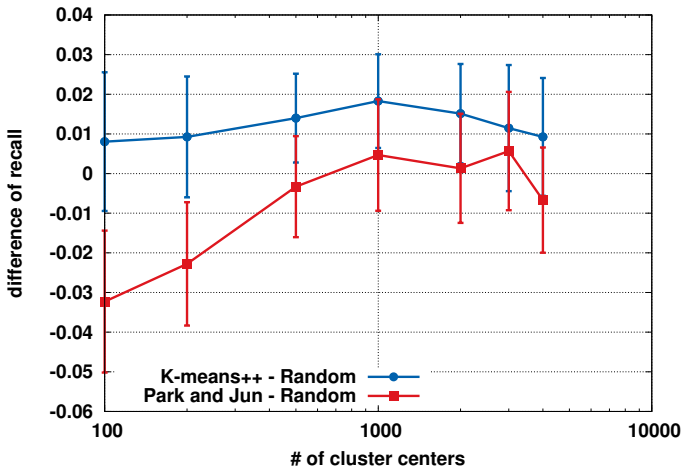


Figure 4.4: Effect of the initialization procedure for the K-medoids clustering in the quality of nearest-neighbors search

Initialization effect: First of all it is important to notice that the initial segment of the curves in Figure 4.4 is not much informative - those points corresponds to a number of clusters up to 100, implying in an almost brute-force search over the dataset (notice the explosion on the extensiveness in Figure 4.3a). Figure 4.4 depicts the difference on the recall metric for the two initialization algorithms, using the random selection as a baseline. The K-means++ initialization affects positively the results with average gain of 1%. On the other hand, the Park and Jun initialization does not present considerable gain over the random baseline (in fact, the average gain is negative). Our analysis indicates that the K-means++ initialization can contribute to better results in the recall metric, while the Park and Jun method presents a null effect (or negative). This graph is averaged over 16 runs of the algorithm and the error bars depicts the standard deviation.

4.4.2 Dictionary Dataset: Comparison of Voronoi LSH and BPI-LSH

We compared our Voronoi LSH with Brief Permutation Indexing (BPI) LSH [Tellez and Chavez, 2010] in a sequential (non-distributed) environment. We implemented our Voronoi LSH on Java with The Apache Commons Mathematics Library[2]. Also included is Distribution-Free LSH (DFLSH) [Kang and Jung, 2012], which we evaluate as a specific configuration of our implementation of Voronoi LSH with the seeds of the Voronoi diagram chosen at random.

[2] *Commons Math: The Apache Commons Mathematics Library*. `http://commons.apache.org/proper/commons-math/`

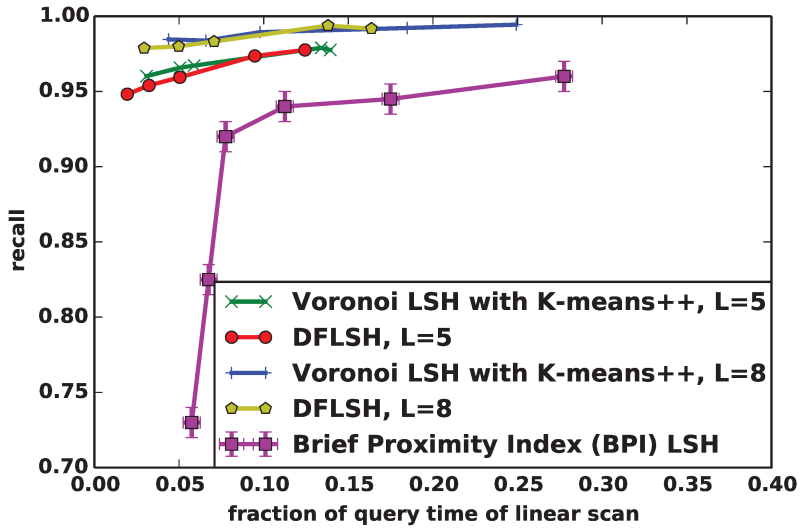


Figure 4.5: Voronoi LSH recall–cost compromises are competitive with those of BPI (error bars are the imprecision of *our interpretation* of BPI original numbers). To make the results commensurable, time is reported as a fraction of brute-force linear scan.

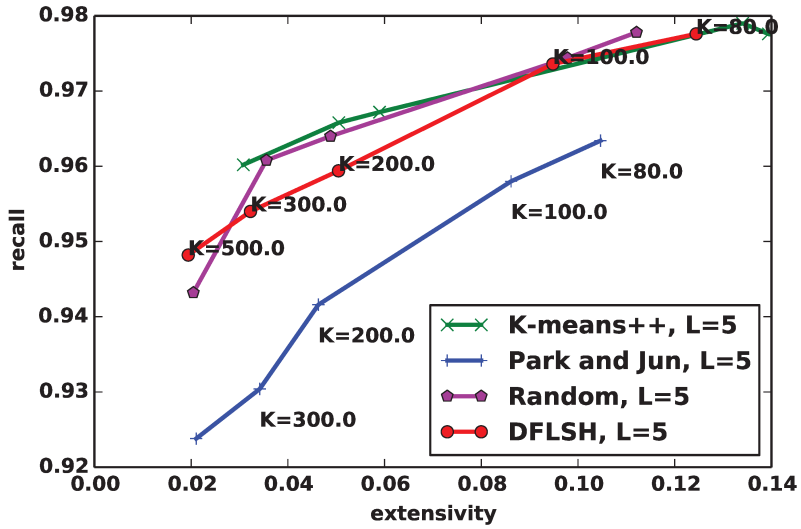


Figure 4.6: Different choices for the seeds of Voronoi LSH: K-medoids with different initializations (K-means++, Park & Jun, random), and using random seeds (DFLSH). Experiments on the English dictionary dataset.

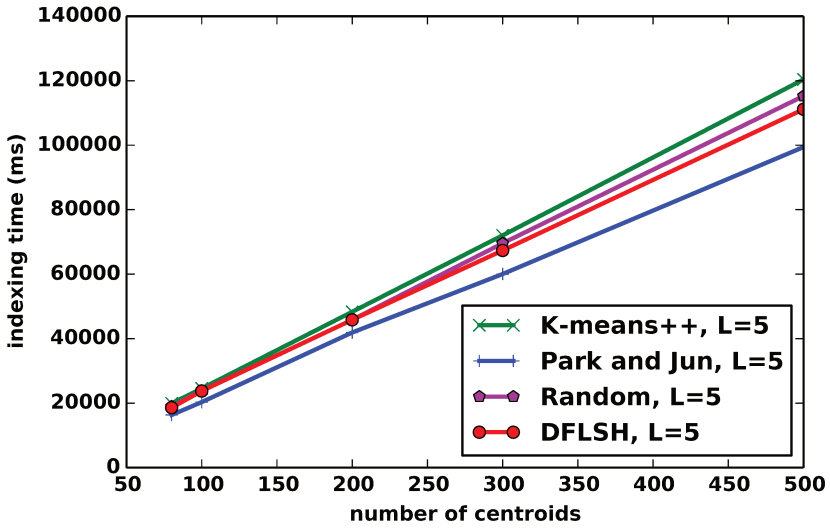


Figure 4.7: Indexing time (ms) varying with the number of centroids. Experiments on the English dictionary dataset.

The experiments used the English Dictionary dataset. Figure 4.5 and Figure 4.6 summarizes the results. We compare the impact of the Clustering algorithm initialization to the search quality of the K-medoids nearest neighbor results. For the sake of this analysis, the initialization strategy proposed in K-means++ and in the work of "Park and Jun" and the naive random initialization are considered. Figure 4.6 presents the recall and query time for varying number of centroids. As shown, the K-means++ initialization and the random initialization performance are not very distinguishable. On the other hand, the initialization of Park and Jun is clearly poorer than the others.

In Figure 4.5 the comparison with BPI-LSH is presented using varying number of centroids for Voronoi LSH (80,100,200,300 and 500, as is reported in Figure 4.6) and scaling the average query time of the method with the average time of linear scan. Under this conditions Voronoi LSH approach offers better trade-off than BPI-LSH, achieving better recall for the same scaled time measure. Besides, it is worth noting that the "knee" of the recall curve is more smooth for Voronoi LSH than for BPI-LSH, or at least is closer to zero, which is a great advantage, since it indicates that as the time factor is decreased, the method keeps its good recall performance longer (in the parametric space) indicating robustness of the method.

4.4.3 Listeria Dataset: DFLSH and VoronoiPlex LSH

As long as it is in our knowledge, there is no comparable reported experimental data for the Approximate Nearest Neighbor problem using the *Listeria* dataset. However since this is a more challenging dataset for strings then the English dictionary, for its average long string sequence, we wanted to test the performance of our methods in this more difficult conditions. Hence, our objective was to advance and understand the empirical performance of the proposed methods in a setup where the distance functions are much harder to calculate.

Figure 4.8 presents the performance of the VoronoiLSH using random samples as centroids over the Listeria gene dataset for 5-NN search. It is possible to see that analyzing less than 1% of the dataset the method achieves more than 85% of recall and, and by a trade-off with memory ($L = 3$ hash table) and processing roughly 1% of the dataset it achieves 94% of recall. It seems very promising to achieve such high rates of recall processing so little of the dataset. Figure 4.9 report similar results using VoronoiPlex LSH, varying the key length w. It is possible to see the obvious recall gain by the use of multiple hash tables, but it is not clear yet how the different parameters of this method relate with each other to achieve a certain recall result.

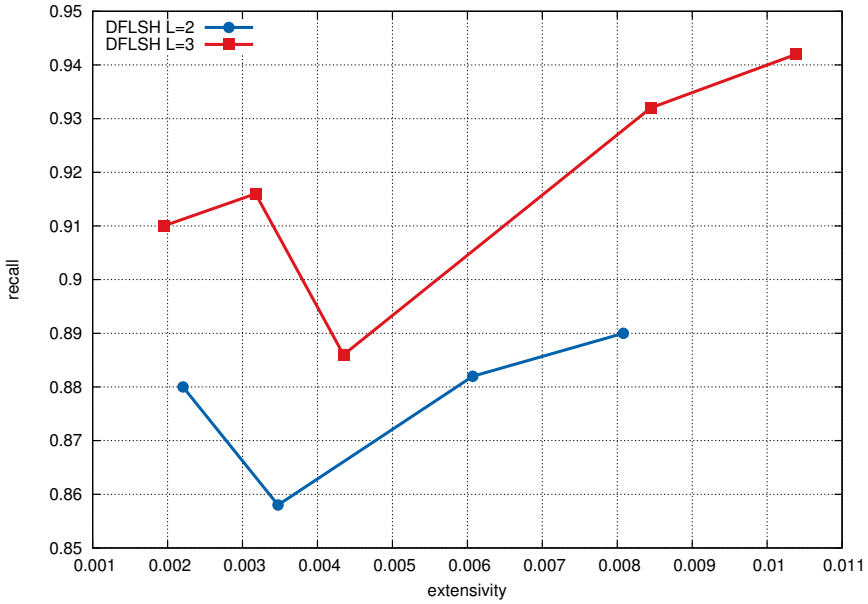


Figure 4.8: Recall metric of VoronoiLSH using random centroids (DFLSH) for the *Listeria* gene dataset using L=2 and L=3 hash-tables and varying the number of centroids (obtaining varying extensivity)

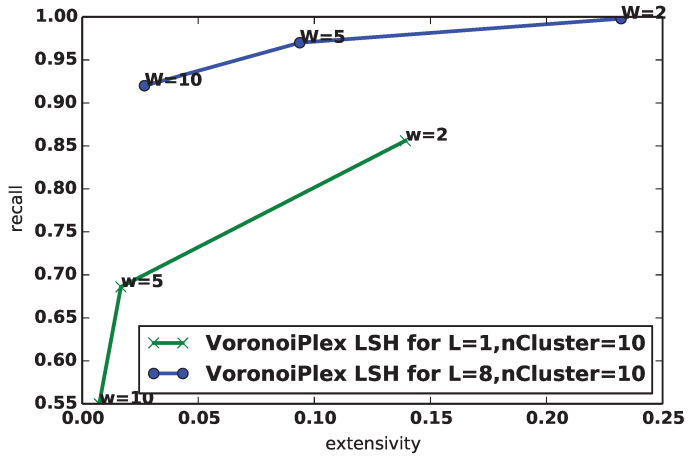


Figure 4.9: Recall metric of VoronoiPlex LSH using random centroids (10 centroids selected from a 4000 point sample set) for the *Listeria* gene dataset using L=1 and L=8 hash-tables and varying the size w of the key-length

4.4.4 BigANN Dataset: Large-Scale Distributed Voronoi LSH

The large-scale evaluation used a distributed-memory machine with 70 nodes interconnected through a FDR Infiniband switch. Each computation node was equipped with a dual-socket Intel E5 2.60 GHz Sandy Bridge processor with a total of 16 CPU cores, 32 GB of DDR3 RAM, running Linux OS kernel version 2.6.32. Parallel Voronoi LSH was implemented on C++ and MPI.

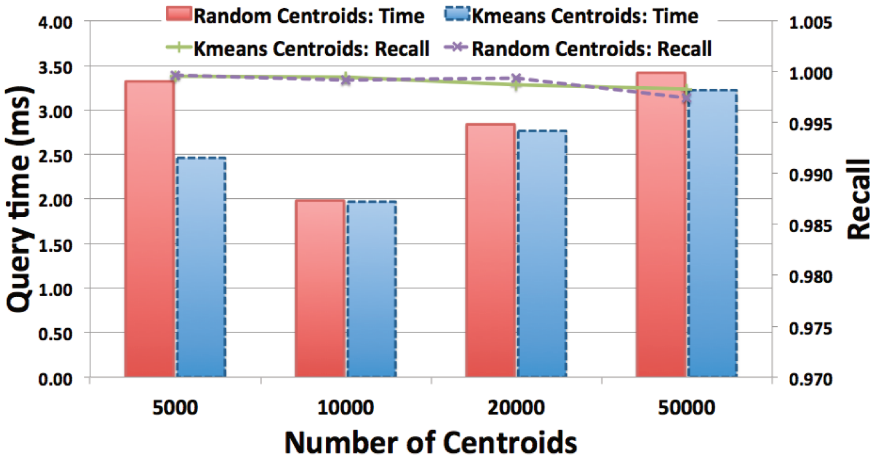


Figure 4.10: Random seeds vs. K-means++ centroids on Parallel Voronoi LSH (BigAnn). Well-chosen seeds have little effect on recall, but occasionally lower query times.

The Figure 4.10 shows the query phase execution time and recall of parallel Voronoi LSH using BigANN dataset with Voronoi seeds chosen at random, and by using K-means++ centroids. Although the difference in recall is negligible, using K-means++ centroids in general resulted in

better query times than using random seeds. The best execution time in both cases was with 10,000 Voronoi seeds. For minimizing the distance computations, there is an expected compromise between too few seeds (cheap hash functions, large shortlists due to more points in buckets) and too many (costly hash functions, smaller shortlists). The theoretical sweet spot is obtained with $\sqrt{n}$ seeds, where n is the dataset size (around 30,000 for BigAnn). However, in our tests, the empirical best value was always much smaller than that, favoring, thus, the retrieval of fewer buckets with many points in them, instead of many buckets with fewer points. That suggests that the cost of accessing the data is overcoming the cost of computing the distances.

Our parallelism efficiency analysis employs a scale-up (weak scaling) experiment in which the reference dataset and the number of computing cores used increase proportionally. A scale-up evaluation was selected because we expect to obtain an abundant volume of data for indexing, which would only fit in a distributed system. For each CPU core allocated to the BI stage 4 CPU cores are assigned to the DP stage, resulting in a ratio of computing cores by BI:DP of 1:4. A single core is used for the AG stage.

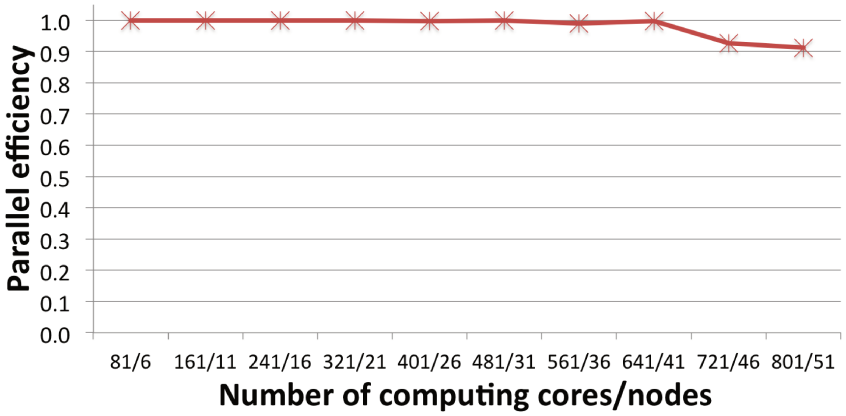


Figure 4.11: Efficiency of Parallel Voronoi LSH parallelization as the number of nodes used and the reference dataset size increase proportionally. The results show that the parallelism scheme scales-up well even for a very large dataset, with modest overhead.

The efficiency of the parallel Voronoi LSH is presented in Figure 4.11. As the number of CPU cores and nodes used increase, the application achieves a very good parallel efficiency of 0.9 when 801 computing cores are used (10 nodes for BI and 40 nodes for DP), indicating very modest parallelism overheads. The high efficiency attained is a result of (i) the application asynchronous design that decouples communication from computation tasks and of (ii) the intra-stage parallelization that allows for a single multi-threaded copy of the DP stage to be instantiated per computing node. As a consequence, a smaller number of partitions of the reference dataset are created, which reduces the number of messages exchanged by the parallel version (using 51 nodes) in more than 6$\times$ as compared to an application version that instantiates a single process per CPU computing core.

Chapter 5

Conclusion

Efficient large-scale similarity search is a crucial operation for Content-based Multimedia Information Retrieval (CMIR) systems. But because those systems employ high-dimensional feature vectors, or other complex representations in metric spaces, providing fast similarity search for them has been a persistent research challenge. LSH, a very successful family of methods, has been advanced as a solution to the problem, but it is available only for a few distance functions.

Voronoi diagrams has been long established as one of the top method for solving nearest-neighbor queries in two dimensional Euclidean spaces. However, difficult to generalize due to the necessity of an exponential amount of space to maintain the polygons that represent the Voronoi region. Nevertheless there has been a variety of methods in high-dimensional Euclidean spaces using K-Means (or other Vector Quantization methods) for Nearest-Neighbor queries, and after some analysis, it is possible to see the k-means partitioning and Vector Quantization as a form of Voronoi partitioning of the data, but dropping the need to store the partition polygons, maintaining only a central element as a representative of the partition. We pursue this same idea for general metric space and use it to address the limitation of possible distance and space for hashing, by extending LSH to general metric spaces, using a Voronoi diagram as basis for a LSH family of functions using "central" points to represent the partitioning. We have proposed a method that follows a more directly the application of this general idea of partitioning (VoronoiLSH) and another one that builds up on this idea in order to create a more selective hash (VoronoiPlex LSH). The central theoretical development is a proof that VoronoiLSH is (r, cr, p_1, p_2)-locality sensitive hashing family under some limited assumptions and that the average performance of the method is sub-linear. The extension of the analysis for VoronoiPlex is natural and is similar to the concatenation "trick" used on the original formulation of LSH, nevertheless a follow-up work should be done in order to be obtained a more complete theoretical characterization of VoronoiPlex. In the algorithm design, to avoid repetitive distance calculations, we added a structure maintained the index of the points selected for a group of hash functions and also presented results characterizing the expected number of points shared by the group of hash functions – it remains open to investigation how this additional structure can affect the final quality.

Our experiments show our partitioning strategy to index the data works well both for metric and

for Euclidean data. The experiments do not show any clear advantage in learning the seeds of the Voronoi diagram by clustering: a random choice seems to work just as well. Further experimental evaluation and theoretical analysis should be pursued in order to confirm or discard the advantages of learning the centroids. If learning does not play a key role in quality of the search, it will also be an important hint for scalability, since clustering is expensive. On the other hand, clustering might in some cases affect *query times*, which is surprising. This seems to be due to a more uniform partition of the data, since random seeds tend to create an unbalanced distribution of the dataset on the buckets of the hash tables. The large-scale experiments show that our proposed parallelization has very modest overhead and scales-up well even for a very large collection.

5.1 Open questions and future work

This work is certainly not exhaustive and there is many opened doors to pursue. One of the more central problems is related to the probabilistic bounds of the hashing methods. The question is how can we integrate our analysis, with other in the similarity metric space framework that uses concentration of measures [Pestov, 2000, 2008; Volnyansky and Pestov, 2009] and unveil the role of intrinsic dimensionality in the case of hashing-based methods for metric indexing, following, for example, interesting relationship between intrinsic dimensionality and discriminability [Houle, 2013]. Another possible fruitful direction would be to follow recent developments in the area of Ptolemaic Indexing, which assumes a more specific inequality axiom (more specific compared to the triangle inequality), and analyze how the proposed methods would work under those different assumptions and design possible adaptations of the method for this different setup.

5.2 Concluding Remarks

This work present two randomized hashing methods designed to approach the problem of similarity search in general metric spaces: VoronoiLSH and VoronoiPlex LSH. Those methods are natural extension to the metric space framework of the established Locality-Sensitive Hashing designed for Hamming, Vector (Angular) and Euclidean Spaces. We offered theoretical characterization of the methods using limiting bounds on the hashing collision probabilities, and average case performance of the search algorithms. Experimental validation and performance comparison with other methods in the literature were presented and the results are good.

We developed, in a collaboration work, a parallelization of the VoronoiLSH algorithms and performed weak scaling experiments on the a thousand million high dimensional points dataset and conclude that the parallel algorithm presented very modest parallelization overhead.

All evidences suggest that using metric partitioning techniques for designing hashing functions for metric space is an effective technique and should be further explored and developed.

We reported to the community two of these results: an initial article presenting VoronoiLSH in a national event (SBBD 2013) and an article describing the parallelization of VoronoiLSH to be presented in an international event (SISAP 2014):

- Eliezer Silva, Thiago Teixeira, George Teodoro, and Eduardo Valle. Large-scale distributed locality-sensitive hashing for general metric data. In *Similarity Search and Applications - Proceedings of the 7th International Conference on Similarity Search and Applications*. Springer, October 2014

- Eliezer Silva and Eduardo Valle. K-medoids lsh: a new locality sensitive hashing in general metric space. In *Proceedings of the 28th Brazilian Symposium on Databases*, pages 127–132, Brazil, 2013. SBC. URL `http://sbbd2013.cin.ufpe.br/Proceedings/artigos/sbbd_shp_22.html`

Bibliography

Charu C. Aggarwal, Alexander Hinneburg, and DanielA. Keim. On the surprising behavior of distance metrics in high dimensional space. In Jan Van den Bussche and Victor Vianu, editors, *Database Theory — ICDT 2001*, volume 1973 of *Lecture Notes in Computer Science*, pages 420–434. Springer Berlin Heidelberg, 2001. ISBN 978-3-540-41456-8. doi: 10.1007/3-540-44503-X_27. URL http://dx.doi.org/10.1007/3-540-44503-X_27.

Giuseppe Amato and Pasquale Savino. Approximate similarity search in metric spaces using inverted files. In *Proceedings of the 3rd International Conference on Scalable Information Systems*, InfoScale '08, pages 28:1–28:10, ICST, Brussels, Belgium, Belgium, 2008. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering). ISBN 978-963-9799-28-8. URL http://dl.acm.org/citation.cfm?id=1459693.1459731.

Alexandr Andoni. *Nearest neighbor search: the old, the new, and the impossible*. PhD thesis, Massachusetts Institute of Technology, 2009.

Alexandr Andoni and Piotr Indyk. Near-Optimal Hashing Algorithms for Approximate Nearest Neighbor in High Dimensions. In *focs*, pages 459–468. Ieee, 2006. ISBN 0-7695-2720-5. doi: 10.1109/FOCS.2006.49. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4031381.

Alexandr Andoni, Piotr Indyk, Huy L. Nguyen, and Ilya Razenshteyn. Beyond locality-sensitive hashing. In Chandra Chekuri, editor, *SODA*, pages 1018–1028. SIAM, 2014. ISBN 978-1-61197-338-9, 978-1-61197-340-2.

Remzi H. Arpaci-Dusseau, Eric Anderson, Noah Treuhaft, David E. Culler, Joseph M. Hellerstein, David A. Patterson, and Katherine Yelick. Cluster I/O with River: Making the Fast Case Common. In *IOPADS '99: Input/Output for Parallel and Distributed Systems*, Atlanta, GA, May 1999.

David Arthur and Sergei Vassilvitskii. k-means++: the advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, SODA '07, pages 1027–1035, Philadelphia, PA, USA, 2007. Society for Industrial and Applied Mathematics. ISBN 978-0-898716-24-5. URL http://dl.acm.org/citation.cfm?id=1283383.1283494.

Michal Batko. *Scalable and Distributed Similarity Search*. PhD thesis, Masarykova univerzita, Fakulta informatiky, 2006. URL <http://is.muni.cz/th/2907/fi_d/>.

Mayank Bawa, Tyson Condie, and Prasanna Ganesan. LSH forest. In *Proceedings of the 14th international conference on World Wide Web - WWW '05*, page 651, New York, New York, USA, 2005. ACM Press. ISBN 1595930469. doi: 10.1145/1060745.1060840. URL http://dl.acm.org/citation.cfm?id=1060840http://portal.acm.org/citation.cfm?doid=1060745.1060840.

Richard Ernest Bellman. *Adaptive control processes: a guided tour*, volume 4 of *'Rand Corporation. Research studies*. Princeton University Press, 1961.

Kevin S. Beyer, Jonathan Goldstein, Raghu Ramakrishnan, and Uri Shaft. When is "nearest neighbor" meaningful? In *Proceedings of the 7th International Conference on Database Theory*, ICDT '99, pages 217–235, London, UK, UK, 1999. Springer-Verlag. ISBN 3-540-65452-6. URL http://dl.acm.org/citation.cfm?id=645503.656271.

Michael D. Beynon, Tahsin Kurc, Umit Catalyurek, Chialin Chang, Alan Sussman, and Joel Saltz. Distributed processing of very large datasets with DataCutter. *Parallel Comput.*, 27(11): 1457–1478, 2001. ISSN 0167-8191. doi: http://dx.doi.org/10.1016/S0167-8191(01)00099-0.

J. Bourgain. On lipschitz embedding of finite metric spaces in hilbert space. *Israel Journal of Mathematics*, 52(1-2):46–52, 1985. ISSN 0021-2172. doi: 10.1007/BF02776078. URL http://dx.doi.org/10.1007/BF02776078.

Andrei Z Broder, Moses Charikar, Alan M Frieze, and Michael Mitzenmacher. Min-wise independent permutations. *Journal of Computer and System Sciences*, 60(3):630–659, 2000.

W. A. Burkhard and R. M. Keller. Some approaches to best-match file searching. *Commun. ACM*, 16(4):230–236, April 1973. ISSN 0001-0782. doi: 10.1145/362003.362025. URL http://doi.acm.org/10.1145/362003.362025.

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

Edgar Chávez and Gonzalo Navarro. A compact space decomposition for effective metric indexing. *Pattern Recognition Letters*, 26(9):1363–1376, 2005.

Edgar Chávez, Gonzalo Navarro, Ricardo Baeza-Yates, and José Luis Marroquín. Searching in metric spaces. *ACM Computing Surveys*, 33(3):273–321, September 2001. ISSN 03600300. doi: 10.1145/502807.502808. URL http://portal.acm.org/citation.cfm?doid=502807.502808.

Edgar Chávez, Karina Figueroa, and Gonzalo Navarro. Proximity searching in high dimensional spaces with a proximity preserving order. In Alexander Gelbukh, Álvaro de Albornoz, and Hugo Terashima-Marín, editors, *MICAI 2005: Advances in Artificial Intelligence*, volume 3789 of *Lecture Notes in Computer Science*, pages 405–414. Springer Berlin Heidelberg, 2005. ISBN

978-3-540-29896-0. doi: 10.1007/11579427_41. URL http://dx.doi.org/10.1007/11579427_41.

Paolo Ciaccia, Marco Patella, and Pavel Zezula. M-tree: An efficient access method for similarity search in metric spaces. In *Proceedings of the 23rd International Conference on Very Large Data Bases*, VLDB '97, pages 426–435, San Francisco, CA, USA, 1997. Morgan Kaufmann Publishers Inc. ISBN 1-55860-470-7. URL http://dl.acm.org/citation.cfm?id=645923.671005.

Kenneth L Clarkson. Nearest-Neighbor Searching and Metric Space Dimensions. In Gregory Shakhnarovich, Trevor Darrell, and Piotr Indyk, editors, *Nearest-Neighbor Methods in Learning and Vision: Theory and Practice (Neural Information Processing)*, Advances in Neural Information Processing Systems. The MIT Press, 2006. ISBN 026219547X. URL http://books.google.com.br/books?id=5FIZAQAAIAAJ.

Mayur Datar, Nicole Immorlica, Piotr Indyk, and Vahab S. Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. In *Proceedings of the twentieth annual symposium on Computational geometry - SCG '04*, page 253, New York, New York, USA, 2004. ACM Press. ISBN 1581138857. doi: 10.1145/997817.997857. URL http://portal.acm.org/citation.cfm?doid=997817.997857.

Ritendra Datta, Dhiraj Joshi, Jia Li, and James Z. Wang. Image retrieval. *ACM Computing Surveys*, 40(2):1–60, April 2008. ISSN 03600300. doi: 10.1145/1348246.1348248. URL http://portal.acm.org/citation.cfm?doid=1348246.1348248.

M Deza and M Laurent. Geometry of cuts and metrics. *Algorithms and Combinatorics*, 15:1–587, 1997.

Michel Marie Deza and Elena Deza. *Encyclopedia of distances*. Springer, 2009.

David L Donoho et al. High-dimensional data analysis: The curses and blessings of dimensionality. *AMS Math Challenges Lecture*, pages 1–32, 2000.

K. Figueroa, G. Navarro, and E. Chávez. Metric spaces library, 2007. Available at http://www.sisap.org/Metric_Space_Library.html .

A Flexer and D. Schnitzer. Can shared nearest neighbors reduce hubness in high-dimensional spaces? In *Data Mining Workshops (ICDMW), 2013 IEEE 13th International Conference on*, pages 460–467, Dec 2013. doi: 10.1109/ICDMW.2013.101.

Kimmo Fredriksson. Engineering efficient metric indexes. *Pattern Recognition Letters*, 28(1): 75–84, 2007.

Aristides Gionis, Piotr Indyk, and Rajeev Motwani. Similarity search in high dimensions via hashing. In *Proceedings of the 25th International Conference on Very Large Data Bases*, VLDB

'99, pages 518–529, San Francisco, CA, USA, 1999. Morgan Kaufmann Publishers Inc. ISBN 1-55860-615-7. URL http://dl.acm.org/citation.cfm?id=645925.671516.

Edgar Chavez Gonzalez, Karina Figueroa, and Gonzalo Navarro. Effective proximity retrieval by ordering permutations. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 30(9): 1647–1658, 2008.

Magnus Lie Hetland. The basic principles of metric indexing. In *Swarm intelligence for multi-objective problems in data mining*, pages 199–232. Springer Berlin Heidelberg, 2009.

Gisli R. Hjaltason and Hanan Samet. Index-driven similarity search in metric spaces (survey article). *ACM Trans. Database Syst.*, 28(4):517–580, December 2003. ISSN 0362-5915. doi: 10.1145/958942.958948. URL http://doi.acm.org/10.1145/958942.958948.

Michael E Houle. Dimensionality, discriminability, density and distance distributions. In *Data Mining Workshops (ICDMW), 2013 IEEE 13th International Conference on*, pages 468–473. IEEE, 2013.

Piotr Indyk and Rajeev Motwani. Approximate nearest neighbors. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing - STOC '98*, pages 604–613, New York, New York, USA, 1998. ACM Press. ISBN 0897919629. doi: 10.1145/276698.276876. URL http://dl.acm.org/citation.cfm?id=276876http://portal.acm.org/citation.cfm?doid=276698.276876.

Herve Jegou, Laurent Amsaleg, Cordelia Schmid, and Patrick Gros. Query adaptative locality sensitive hashing. In *2008 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 825–828. IEEE, March 2008. ISBN 978-1-4244-1483-3. doi: 10.1109/ICASSP.2008.4517737. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4517737.

Herve Jegou, Romain Tavenard, Matthijs Douze, and Laurent Amsaleg. Searching in one billion vectors: Re-rank with source coding. In *ICASSP*, pages 861–864. IEEE, 2011. ISBN 978-1-4577-0539-7.

Alexis Joly and Olivier Buisson. A posteriori multi-probe locality sensitive hashing. In *Proceedings of the 16th ACM International Conference on Multimedia*, MM '08, pages 209–218, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-303-7. doi: 10.1145/1459359.1459388. URL http://doi.acm.org/10.1145/1459359.1459388.

Byungkon Kang and Kyomin Jung. Robust and Efficient Locality Sensitive Hashing for Nearest Neighbor Search in Large Data Sets. In *NIPS Workshop on Big Learning (BigLearn)*, pages 1–8, Lake Tahoe, Nevada, 2012. URL http://biglearn.org/2012/files/papers/biglearning2012_submission_5.pdf.

Leonard Kaufman and Peter J. Rousseeuw. *Finding Groups in Data: An Introduction to Cluster Analysis*. Wiley-Interscience, New York, USA, 9th edition, March 1990. ISBN 0471878766. URL http://www.amazon.com/exec/obidos/redirect?tag=citeulike07-20&path=ASIN/0471878766.

John Leech. Some sphere packings in higher space. *Canadian Journal of Mathematics*, 16: 657–682, January 1964. ISSN 1496-4279. doi: 10.4153/CJM-1964-065-1. URL http://www.cms.math.ca/10.4153/CJM-1964-065-1.

David G. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision*, 60(2):91–110, November 2004. ISSN 0920-5691. doi: 10.1023/B:VISI.0000029664.99615.94. URL http://www.springerlink.com/openurl.asp?id=doi:10.1023/B:VISI.0000029664.99615.94.

Qin Lv, William Josephson, Zhe Wang, Moses Charikar, and Kai Li. Multi-probe LSH: efficient indexing for high-dimensional similarity search. In *Proceedings of the 33rd international conference on Very large data bases*, VLDB '07, pages 950–961. VLDB Endowment, 2007. ISBN 978-1-59593-649-3. URL http://dl.acm.org/citation.cfm?id=1325851.1325958.

Jiří Matoušek. *Lectures on discrete geometry*, volume 108. Springer New York, 2002.

Jiří Matoušek. On the distortion required for embedding finite metric spaces into normed spaces. *Israel Journal of Mathematics*, 93(1):333–344, 1996. ISSN 0021-2172. doi: 10.1007/BF02761110. URL http://dx.doi.org/10.1007/BF02761110.

Rajeev Motwani, Assaf Naor, and Rina Panigrahy. Lower bounds on locality sensitive hashing. In *Proceedings of the twenty-second annual symposium on Computational geometry - SCG '06*, page 154, New York, New York, USA, 2006. ACM Press. ISBN 1595933409. doi: 10.1145/1137856.1137881. URL http://portal.acm.org/citation.cfm?doid=1137856.1137881.

RT Ng. CLARANS: a method for clustering objects for spatial data mining. *IEEE Transactions on Knowledge and Data Engineering*, 14(5):1003–1016, September 2002. ISSN 1041-4347. doi: 10.1109/TKDE.2002.1033770. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1033770http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1033770.

David Novak and Michal Batko. Metric Index: An Efficient and Scalable Solution for Similarity Search. In *2009 Second International Workshop on Similarity Search and Applications*, pages 65–73. IEEE Computer Society, August 2009. ISBN 978-0-7695-3765-8. doi: 10.1109/SISAP.2009.26. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5272384.

David Novak, Martin Kyselak, and Pavel Zezula. On locality-sensitive indexing in generic metric spaces. In *Proceedings of the Third International Conference on SImilarity Search and APplications - SISAP '10*, pages 59–66, New York, New York, USA, 2010. ACM Press. ISBN 9781450304207. doi: 10.1145/1862344.1862354. URL http://portal.acm.org/citation.cfm?doid=1862344.1862354.

Alexander Ocsa and Elaine P M De Sousa. An Adaptive Multi-level Hashing Structure for Fast Approximate Similarity Search. *Journal of Information and Data Management*, 1(3):359–374, 2010.

Ryan O'Donnell, Yi Wu, and Yuan Zhou. Optimal lower bounds for locality-sensitive hashing (except when q is tiny). *ACM Trans. Comput. Theory*, 6(1):5:1–5:13, March 2014. ISSN 1942-3454. doi: 10.1145/2578221. URL http://doi.acm.org/10.1145/2578221.

Rafail Ostrovsky, Yuval Rabani, Leonard Schulman, and Chaitanya Swamy. The Effectiveness of Lloyd-Type Methods for the k-Means Problem. In *focs*, pages 165–176. IEEE, December 2006. ISBN 0-7695-2720-5. doi: 10.1109/FOCS.2006.75. URL http://dl.acm.org/citation.cfm?doid=2395116.2395117http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4031353.

Rina Panigrahy. Entropy based nearest neighbor search in high dimensions. In *Proceedings of the seventeenth annual ACM-SIAM symposium on Discrete algorithm*, SODA '06, pages 1186–1195, New York, NY, USA, 2006. ACM. ISBN 0-89871-605-5. doi: 10.1145/1109557.1109688. URL http://doi.acm.org/10.1145/1109557.1109688.

Hae-Sang Park and Chi-Hyuck Jun. A simple and fast algorithm for K-medoids clustering. *Expert Systems with Applications*, 36(2):3336–3341, 2009. ISSN 09574174. doi: 10.1016/j.eswa.2008.01.039. URL http://linkinghub.elsevier.com/retrieve/pii/S095741740800081X.

Marco Patella and Paolo Ciaccia. Approximate similarity search: A multi-faceted problem. *Journal of Discrete Algorithms*, 7(1):36 – 48, 2009. ISSN 1570-8667. doi: http://dx.doi.org/10.1016/j.jda.2008.09.014. URL http://www.sciencedirect.com/science/article/pii/S1570866708000762. Selected papers from the 1st International Workshop on Similarity Search and Applications (SISAP) 1st International Workshop on Similarity Search and Applications (SISAP).

Adriano Arantes Paterlini, Mario A Nascimento, and Caetano Traina Junior. Using Pivots to Speed-Up k-Medoids Clustering. *Journal of Information and Data Management*, 2(2):221–236, June 2011. URL http://seer.lcc.ufmg.br/index.php/jidm/article/view/99.

Loïc Paulevé, Hervé Jégou, and Laurent Amsaleg. Locality sensitive hashing: A comparison of hash function types and querying mechanisms. *Pattern Recognition Letters*, 31(11):1348–1358, August 2010. ISSN 01678655. doi: 10.1016/j.patrec.2010.04.004. URL http://linkinghub.elsevier.com/retrieve/pii/S0167865510001169.

Vladimir Pestov. On the geometry of similarity search: Dimensionality curse and concentration of measure. *Information Processing Letters*, 73(1–2):47 – 51, 2000. ISSN 0020-0190. doi: http://dx.doi.org/10.1016/S0020-0190(99)00156-8. URL `http://www.sciencedirect.com/science/article/pii/S0020019099001568`.

Vladimir Pestov. An axiomatic approach to intrinsic dimension of a dataset. *Neural Networks*, 21 (2):204–213, 2008.

Miloš Radovanović, Alexandros Nanopoulos, and Mirjana Ivanović. Nearest neighbors in high-dimensional data: The emergence and influence of hubs. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pages 865–872, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-516-1. doi: 10.1145/1553374.1553485. URL `http://doi.acm.org/10.1145/1553374.1553485`.

Miloš Radovanović, Alexandros Nanopoulos, and Mirjana Ivanović. Hubs in space: Popular nearest neighbors in high-dimensional data. *J. Mach. Learn. Res.*, 11:2487–2531, December 2010. ISSN 1532-4435. URL `http://dl.acm.org/citation.cfm?id=1756006.1953015`.

Satish Rao. Small distortion and volume preserving embeddings for planar and euclidean metrics. In *Proceedings of the fifteenth annual symposium on Computational geometry*, pages 300–306. ACM, 1999.

Hanan Samet. *Foundations of Multidimensional and Metric Data Structures*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2005. ISBN 0123694469.

Uri Shaft and Raghu Ramakrishnan. Theory of nearest neighbors indexability. *ACM Trans. Database Syst.*, 31(3):814–838, September 2006. ISSN 0362-5915. doi: 10.1145/1166074.1166077. URL `http://doi.acm.org/10.1145/1166074.1166077`.

Eliezer Silva and Eduardo Valle. K-medoids lsh: a new locality sensitive hashing in general metric space. In *Proceedings of the 28th Brazilian Symposium on Databases*, pages 127–132, Brazil, 2013. SBC. URL `http://sbbd2013.cin.ufpe.br/Proceedings/artigos/sbbd_shp_22.html`.

Eliezer Silva, Thiago Teixeira, George Teodoro, and Eduardo Valle. Large-scale distributed locality-sensitive hashing for general metric data. In *Similarity Search and Applications - Proceedings of the 7th International Conference on Similarity Search and Applications*. Springer, October 2014.

Tomáš Skopal. Where are you heading, metric access methods?: a provocative survey. In *Proceedings of the Third International Conference on SImilarity Search and APplications*, SISAP '10, pages 13–21, New York, NY, USA, 2010. ACM. ISBN 978-1-4503-0420-7. doi: 10.1145/1862344.1862347. URL `http://doi.acm.org/10.1145/1862344.1862347`.

Malcolm Slaney, Yury Lifshits, and Junfeng He. Optimal Parameters for Locality-Sensitive Hashing. *Proceedings of the IEEE*, 100(9):2604–2623, 2012.

Eric Sadit Tellez and Edgar Chavez. On locality sensitive hashing in metric spaces. In *Proceedings of the Third International Conference on SImilarity Search and APplications*, SISAP '10, pages 67–74, New York, NY, USA, 2010. ACM. ISBN 978-1-4503-0420-7. doi: 10.1145/1862344.1862355. URL http://doi.acm.org/10.1145/1862344.1862355.

George Teodoro, Daniel Fireman, Dorgival Guedes, Wagner Meira Jr., and Renato Ferreira. Achieving multi-level parallelism in the filter-labeled stream programming model. *Parallel Processing, International Conference on*, 0:287–294, 2008. ISSN 0190-3918. doi: http://doi.ieeecomputersociety.org/10.1109/ICPP.2008.72.

A.J.M. Traina, A. Traina, C. Faloutsos, and B. Seeger. Fast indexing and visualization of metric data sets using slim-trees. *Knowledge and Data Engineering, IEEE Transactions on*, 14(2):244–260, 2002. ISSN 1041-4347. doi: 10.1109/69.991715.

Caetano Traina, Jr., Agma J. M. Traina, Bernhard Seeger, and Christos Faloutsos. Slim-trees: High performance metric trees minimizing overlap between nodes. In *Proceedings of the 7th International Conference on Extending Database Technology: Advances in Database Technology*, EDBT '00, pages 51–65, London, UK, UK, 2000. Springer-Verlag. ISBN 3-540-67227-3. URL http://dl.acm.org/citation.cfm?id=645339.650146.

Jeffrey K. Uhlmann. Satisfying general proximity / similarity queries with metric trees. *Information Processing Letters*, 40(4):175 – 179, 1991. ISSN 0020-0190. doi: http://dx.doi.org/10.1016/0020-0190(91)90074-R. URL http://www.sciencedirect.com/science/article/pii/002001909190074R.

Ilya Volnyansky and Vladimir Pestov. Curse of dimensionality in pivot based indexes. In *Proceedings of the 2009 Second International Workshop on Similarity Search and Applications*, pages 39–46. IEEE Computer Society, 2009.

Peter N. Yianilos. Data structures and algorithms for nearest neighbor search in general metric spaces. In *Proceedings of the fourth annual ACM-SIAM Symposium on Discrete algorithms*, SODA '93, pages 311–321, Philadelphia, PA, USA, 1993. Society for Industrial and Applied Mathematics. ISBN 0-89871-313-7. URL http://dl.acm.org/citation.cfm?id=313559.313789.

Pavel Zezula. Future trends in similarity searching. In *Proceedings of the 5th international conference on Similarity Search and Applications*, SISAP'12, pages 8–24, Berlin, Heidelberg, 2012. Springer-Verlag. ISBN 978-3-642-32152-8. doi: 10.1007/978-3-642-32153-5_2. URL http://dx.doi.org/10.1007/978-3-642-32153-5_2.

Pavel Zezula, Giuseppe Amato, Vlastislav Dohnal, and Michal Batko. *Similarity Search - The Metric Space Approach*, volume 32 of *Advances in Database Systems*. Kluwer Academic Publishers, Boston, 2006. ISBN 0-387-29146-6. doi: 10.1007/0-387-29151-2. URL http://www.springerlink.com/index/10.1007/0-387-29151-2.

Qiaoping Zhang and Isabelle Couloigner. A new and efficient k-medoid algorithm for spatial clustering. In Osvaldo Gervasi, MarinaL. Gavrilova, Vipin Kumar, Antonio Laganà, HeowPueh Lee, Youngsong Mun, David Taniar, and ChihJengKenneth Tan, editors, *Computational Science and Its Applications (ICCSA)*, volume 3482 of *Lecture Notes in Computer Science*, pages 181–189. Springer Berlin Heidelberg, Singapore, 2005. ISBN 978-3-540-25862-9. doi: 10.1007/11424857_20. URL http://dx.doi.org/10.1007/11424857_20.

Appendix A

Implementation

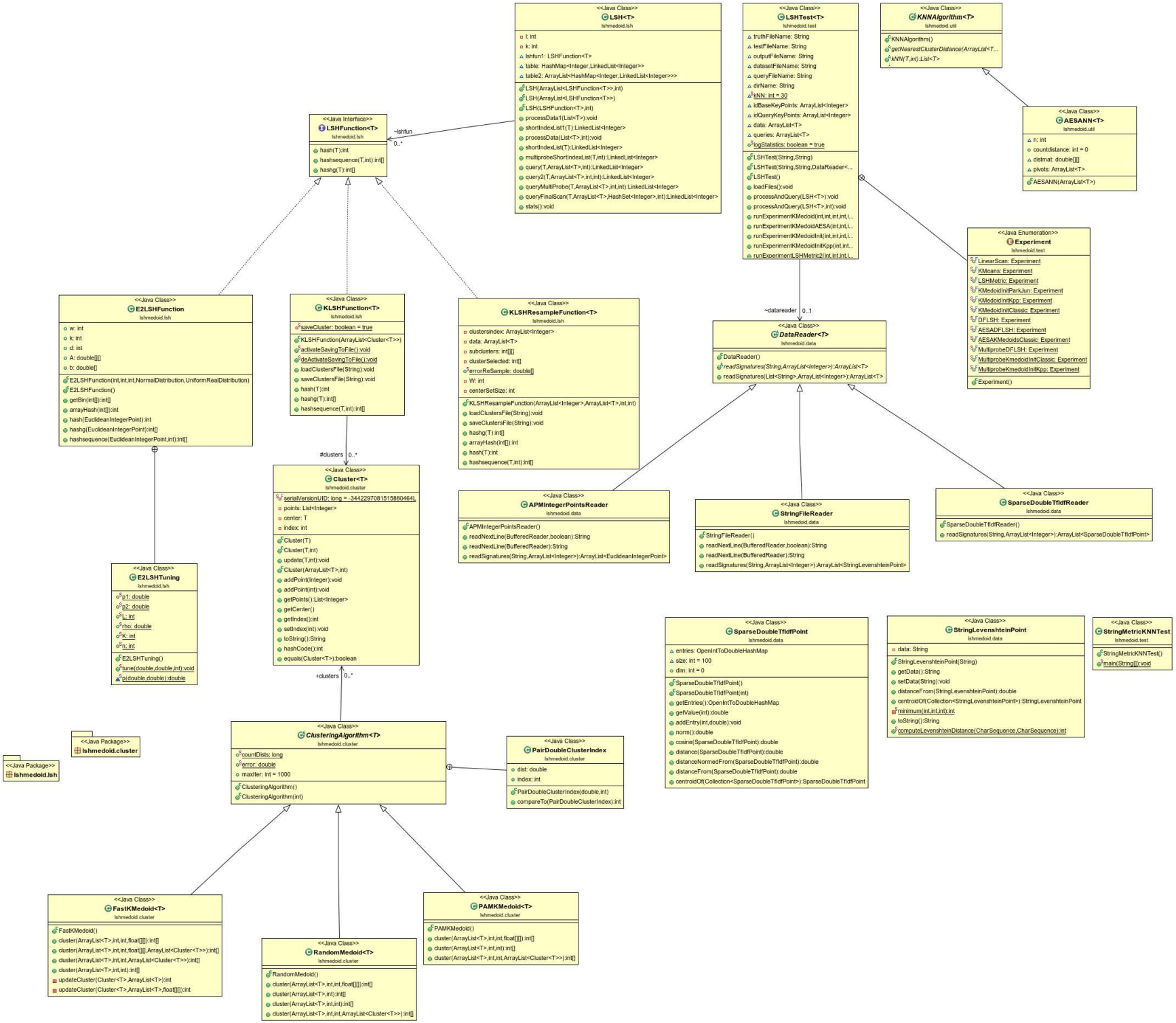


Figure A.1: Class diagram of the system