Università degli Studi di Bologna

Dipartimento di Elettronica Informatica e Sistemistica

Dottorato di Ricerca in Ingegneria Elettronica,
Informatica e delle Telecomunicazioni

XIX Ciclo

Models and Techniques for Approximate Similarity Search in Large Databases

Tesi di:
Dott. Ing. Alessandro Linari

Coordinatore:
Chiar.mo Prof. Ing. Paolo Bassi

Relatori:
Chiar.mo Prof. Ing. Paolo Ciaccia

Chiar.mo Prof. Ing. Marco Patella

Settore Scientifico Disciplinare ING-INF/05

Contents

Chapter 1

Introduction

This thesis investigates the problem of the processing of approximate similarity queries in very large data bases. We assume that a data base is split in multiple, independent regions which can be accessed separately.

An informal definition of a *data region* is:

Definition 1.1 (Data region)**.** *A* data region *is a small subset of a data collection which can be conveniently stored, searched an retrieved, independently from the rest of data set.*

All modern database management systems (DBMS) are organized as a set of regions. Multimedia DBMS's, for example, organize their data in *pages* that are stored in secondary memory and distributed data bases such as peer-to-peer (P2P) systems rely on storing each region at physically different locations.

With approximate similarity search we refer to those situation where one can trade-off the quality of the result for the speed of evaluation. The idea is that the user does not need to receive as a result the data object(s) that best match the query; in certain cases the full computation of the query consumes too many resources or it takes too much time (is it feasible to contact all the peers of a P2P network?). In other circumstances it might be not needed (if you search a photo of the *Colosseum*, does it make sense to define the best-among-all pictures?) and so on.

1.1 Motivation

Despite their differences, the vast majority of modern data base systems are organized into regions which contain data objects and which are accessed independently when the query processing takes place. The two main issues are:

Efficiency, i.e., the amount of resources that are needed in order to answer the query.

effectiveness, i.e., the accuracy of the results that are retrieved by the system.

When a system performs an approximate search it means that there is a constrain consisting in the amount of resources that can be used to find the answer (limited number of steps or bounded period of time). In our scenario, then, we aim at developing tools and techniques for processing similarity queries which optimize the use of these resources by minimizing the number of regions accessed to find a result and, at the same time, maximizing its quality.

A standard technique used to reduce query costs is to provide each region with a set of statistics. This information can also be exploited at query time to guide the search in a more effective way in order to reduce the costs and to increase the quality of results. Our solution, then, should address:

- The formal definition of data region, in order to model a variety of different problems;
- Their statistical characterization, to be exploited at query time to distinguish among relevant and non-relevant regions;
- The existence of optimal scheduling strategies for the problem of similarity search.

1.2 Summary of Contributions

The contribution of this thesis is the following:

- We present a theoretical framework for the characterization of the problem of approximate similarity search in large data bases. Our framework does not depend on the specific implementation and is applied both to a centralized and to a distributed scenario. We are particularly interested to the relevant case of databases which can be embedded in a metric space.
- We interpret a database as a *network of data regions* and distinguish among: completely connected networks, hierarchical networks and partially connected networks.
- We propose a cost model for approximate similarity queries and derive optimal-on-the-average schedules in completely connected networks. We conduct extensive experiments with three different data set to validate our theoretical results.
- We study the problem of similarity search in partially connected networks and propose a novel P2P architecture called Metric Overlay Network (MON), which exploits a metric distance between peers to build a network where regions are connected to their k nearest neighbors.

- We apply the notion of MON to a problem of Information Retrieval in P2P networks, where peers are represented as Language Models (LM) and the metric distance is based on the Karhunen-Loeve divergence.

- We further apply the notion of MON to a problem of security: we show how it is possible to achieve anonymous and censorship resistant query routing in a P2P network which is embedded in a metric space. Based on the metric properties of the system, we give probabilistic guarantees on the level of censorship resistance that can be achieved by the system.

1.3 Thesis Outline

The thesis is organized as follows:

- In Chapter 2 we formally define the problem of similarity search and we characterize a data set as a collection of data regions. We statistically represent each data region through the distance distribution of the nearest neighbor from a random query and present the algorithm for exact and approximate similarity search.

- In Chapter 3 we present a cost model for approximate 1-NN queries which applies to completely connected networks. We further show how, given the cost model, we are able to derive optimal-on-the-average scheduling strategies. We validate our theoretical framework by conducting extensive experiments on three real data sets and we show that, within some extent, also the similarity search in hierarchical networks benefit from our studies.

- In Chapter 4 we investigate the issue of query processing in partially connected networks. We introduce the concept of Metric Overlay Network(MON). Peers in a MON self-organize in order to build an infrastructure which facilitates query routing by exploiting the metric properties of the network. We present three different routing strategies for queries in a MON and compare them through experiments on a real data set.

- In Chapter 5 we present the conclusions of our work and a few open problems that still need to be investigated.

- Appendix A shows an efficient technique to compress a LM by representing it through a histogram and by further compressing the histogram by using a set of Bloom-filters.

- Appendix B presents *Clouds*, a P2P system in which the communications can be made anonymous and censorship resistant by exploiting the metric properties of the network.

Chapter 2

Similarity Search in Large Data Bases

In this chapter We introduce our notion of approximate similarity search in metric spaces and present a formalization for the concepts of data regions. At the end of the chapter we review some related work in the field of similarity search.

2.1 Similarity Search

The standard way to decide which objects o best match a query q is to utilize a *similarity function* $sim(q, o)$, which assigns a numerical score to each pair (q, o). The scores corresponding to different objects are then compared to rank the objects with respect to the query and a similarity search retrieves the top-ranked objects with respect to the search criterion.

In general, the result to a similarity query is not a single object, rather it is a list $\mathcal{R}$ of objects, chosen according to some given criterion. We define the problem of exact similarity search as follows:

Definition 2.1 (Exact similarity search)**.** *Let $\mathcal{X}$ be the (possibly infinite) set of all query points, $\mathcal{D}$ the set of objects in the data set, $sim : \mathcal{X} \times \mathcal{D} \rightarrow \mathbb{R}_0^+$ a similarity function between queries and objects and $crit(s, \mathcal{R}) \in \{0, 1\}$ a similarity criterion, where s is a similarity score and $\mathcal{R}$ the current result list. Then, we define a problem of similarity search over the data set $\mathcal{D}$ to be the quadruple $\langle \mathcal{X}, \mathcal{D}, sim, crit \rangle$.*

The similarity criterion $crit(s, \mathcal{R})$ in Definition 2.1 returns 1 if the object whose similarity value is s should be inserted in the result $\mathcal{R}$, 0 otherwise. Following are the three classes of queries, and their similarity criterion, which has been considered in this work.

Definition 2.2 (Range query). *A range query is defined as* $range(q, r)$*, where* $q \in \mathcal{X}$ *is the query point and* $r \in \mathbb{R}_0^+$ *is the* query radius. *The similarity criterion* $crit(s, r)$*, where* $s = sim(q, o)$ *is defined as:* $crit(s, r) = 1$ iff $s \leq r$.

Definition 2.3 (k nearest neighbor query). *A* k *nearest neighbor query is defined as* $knn(q, k)$*, where* $q \in \mathcal{X}$ *is the query point and* $k \in \mathbb{N}$ *is the cardinality of the result list. A* k*-NN query selects* k *objects* $o_1, \ldots, o_k$ *and keeps them ordered such that* $sim(q, o_1) \geq \ldots \geq sim(q, o_k) \geq sim(q, o)$*. The similarity criterion* $crit(s, \mathcal{R})$ *is defined as:* $crit(s, r) = 1$ iff $s \geq sim(q, o_k)$.

Definition 2.4 (Bounded k nearest neighbor query). *A bounded* k *nearest neighbor query is defined as* $bound(q, k, r)$*, where* $q \in \mathcal{X}$ *is the query point,* $k \in \mathbb{N}$ *is the maximum cardinality of the result-set and* r *is the radius of the query. A bounded* k*-NN query selects* $k' \leq k$ *objects in the intersection of* $range(q, r) \cap knn(q, k)$ *and* $crit(s, , r, \mathcal{R})$ *is defined as:* $crit(s, r, \mathcal{R}) = 1$ iff $s \geq sim(q, o_k) \wedge s \leq r$.

For ease of simplicity in the following we will refer to a query with its query-point q and we will do otherwise only when needed.

2.2 Approximate Similarity Search

The complexity of processing similarity queries in very large data bases is nowadays well recognized and has lead in recent years to the development of new *approximate* search paradigms, where one can trade-off the quality of the result for the speed of evaluation. The adoption of these paradigms is is even more justified by the observation that a similarity function is itself an approximation of the user's criteria.

An approximate similarity search returns a set of objects $\widetilde{\mathcal{R}} = \{\widetilde{o}_i\}, o_i \in \mathcal{D}$ which, in terms of similarity with respect to the query, are not better than those returned by a similarity search as it is defined by Definition 2.1. More formally we characterize it as follows:

Definition 2.5 (Approximate similarity search). *A problem of similarity search over the data set* $\mathcal{D}$ *is defined by the tuple* $\langle \mathcal{X}, \mathcal{D}, sim, crit, err \rangle$*, where* $\mathcal{X}, \mathcal{D}, sim, crit$ *have the usual meaning and* $err : \{\mathcal{R}\} \times \{\widetilde{\mathcal{R}}\} \rightarrow \mathbb{R}_0^+$ *is an* error function *whose value is proportional to the difference between the* approximate *and exact result* $\widetilde{\mathcal{R}}$ *and* $\mathcal{R}$.

A general definition for the error function *err* does not exist because it depends on the type of query that has been processed and on the problem that has to be solved. In the following paragraph we present a short discussion on the problem of defining a suitable error function and present a few examples.

2.2.1 Error Metrics

If we consider a simple 1-NN query, an typical error measure is given by

$$Err = \frac{sim(q, \widetilde{\mathcal{R}})}{sim(q, \mathcal{R})} - 1 \tag{2.1}$$

where $sim(q, \widetilde{\mathcal{R}})$ is the similarity between the query and the object in the approximate result, while $sim(q, \mathcal{R})$ is the similarity between the query and the object in the exact result. Notice that in Equation 2.1 it is subtracted the value 1 such that, in the case of exact search, an error of 0 is returned.

Extending the previous Equation to the case of k-NN queries is not obvious, because there are k objects in a result and, even if it is straightforward to define the i-th error as $Err_i = \frac{sim(q,\widetilde{\mathcal{R}}_i)}{sim(q,\mathcal{R}_i)} - 1$, however it is not clear how these values should be combined together. Reasonable definitions of error for a k-NN similarity search include (see [CP02] for more details):

$$Err = \max_{i=1}^{k} Err_i = \max_{i=1}^{k} \left\{ \frac{sim(q, \widetilde{\mathcal{R}}_i)}{sim(q, \mathcal{R}_i)} - 1 \right\} \qquad \text{(Maximum Error)} \tag{2.2}$$

$$Err = \frac{1}{k} \cdot \sum_{i=1}^{k} Err_i = \frac{1}{k} \cdot \sum_{i=1}^{k} \left[\frac{sim(q, \widetilde{\mathcal{R}}_i)}{sim(q, \mathcal{R}_i)} - 1 \right] \qquad \text{(Average Error)} \tag{2.3}$$

$$Err = Err_k = \frac{sim(q, \widetilde{\mathcal{R}}_k)}{sim(q, \mathcal{R}_k)} - 1 \qquad (k\text{-th Error}) \tag{2.4}$$

In the case of range search, the error measure should take into account the cardinality of the result set rather than the similarity between the returned objects and the query. The derived error measure is shown in Equation 2.5:

$$Err = 1 - \frac{|\widetilde{\mathcal{R}}|}{|\mathcal{R}|} \tag{2.5}$$

where the cardinality of the approximate and exact result sets are denoted as $|\widetilde{\mathcal{R}}|$ and $|\mathcal{R}|$, respectively (notice that $|\widetilde{\mathcal{R}}| \leq |\mathcal{R}|$ holds).

2.2.2 On-line Similarity Search

On-line processing of similarity queries aims at delivering as early as possible good approximate results, $\widetilde{\mathcal{R}}$, which get progressively refined over time. Eventually, if enough resources are available, the exact result is obtained, i.e., $\widetilde{\mathcal{R}}$ converges to $\mathcal{R}$. A fundamental aspect of on-line processing concerns user interaction, i.e., how the user can control query execution [Hel97]. We consider the following control policies:

Cost: The user can limit the amount of resources that the algorithm can use, e.g., by requiring that no more that a certain number of regions are read at every update [CCV02, BN04]

Quality: The user can set a quality threshold θ on similarity values in order to stop the execution as soon as a result is deemed to be "good enough".

In general, then, a control policy based on limiting the resources consumption stops the search has soon as the query has explored a maximum number of regions. A control policy on the *expected* quality of the result, on the other hand, continues processing the query until it has found an approximate result $\widetilde{\mathcal{R}}$ whose similarity with the query is high enough.

2.3 Data Regions

Nowadays databases typically occupy several gigabytes of storage and secondary memory is necessary to store all the data. When a query is processed, the naïve solution consists in the sequential scan of the data base to search for the objects that match the query. A better solution consists in organizing the data set in *regions* which can be stored, queried and retrieved independently and to access them through an index structure.

2.3.1 Creation

Data regions are created based on a *split rule*, i.e.,, a criterion by which the objects in the data set are grouped to form the regions. The split rule can be defined over the dataset and, in this case, it is said to be *user-driven*, because it is dictated by external constraints which are defined directly by the user or by some other "human" factor. An example is a data base where regions contain all the objects physically stored at the same location, because they belong to a company or because they can be accessed from a specific web site.

If the split rule is *data-driven*, then it is guided by considerations which apply to the space $\mathcal{X}$ and is typically controlled by the underlying system as a way to optimize the management of certain operations, such as query processing. For example, to optimize nearest neighbor queries the data set should be clustered based on the similarity function, such that a query can access only the relevant regions.

Definition 2.6 (User-driven split rule)**.** *A* user-driven *split rule defines the optimal set* $\{\mathcal{D}_1, \mathcal{D}_2, \ldots\}$ *of (possibly overlapping) regions over the data set* $\mathcal{D}$ *as the one which minimizes the cost function* $f_{\mathcal{D}}(\mathcal{D}, \{\mathcal{D}_i\}, K)$*, where* K *is a set of user-defined parameters.*

In Definition 2.6, the parameter K strongly depends on the problem at hand. In a distributed data base in which the data is stored across different sites a primary goal is to avoid to relocate the existing storage. This is, of course, not always convenient from the query processing point of view: assume user u_A at site s_A frequently accesses the data stored at site s_B, it would be preferable to move the data to s_A to avoid overloading the communication network and to guarantee a higher response time.

A data-driven split rule can be defined in terms of the *self-similarity measure* $m_{sim}(q, \mathcal{D}_i)$ of a region $\mathcal{D}_i$ with respect to a query q, which measures the discrepancy in the values of similarity $sim(q, o)$ that are computed between q and all objects in $o \in \mathcal{D}_i$. For example, m_{sim} can be computed as $1 - var[sim(q, \mathbf{o})]$, where $var[\cdot]$ is the variance of the random variable $sim(q, \mathbf{o})$ and $\mathbf{o}$ is a random object drawn from $\mathcal{D}_i$.

Definition 2.7 (Data-driven split rule)**.** *A* data-driven *split rule defines the optimal set* $\{\mathcal{D}_1, \mathcal{D}_2, \ldots\}$ *of (possibly overlapping) regions over the data set* $\mathcal{D}$ *as the one which jointly maximizes the self-similarities* $m_{sim}(\mathbf{q}, \mathcal{D}_1), m_{sim}(\mathbf{q}, \mathcal{D}_2), \ldots$ *for all regions* $\mathcal{D}_i$ *with respect to the random query* $\mathbf{q}$*.*

Given a set of regions $\{\mathcal{D}_1, \mathcal{D}_2, \ldots\}$, then, we need to define a criterion by which the $m_{sim}(\mathbf{q}, \mathcal{D}_1), m_{sim}(\mathbf{q}, \mathcal{D}_2), \ldots$ are jointly maximized. Notice that we can exploit similar arguments to those that were used on the definition of the error metrics in Section 2.2.1.

A data-driven split rule which exploits Definition 2.7 defines a set of highly clustered regions, i.e.,, which contain objects having approximately the same similarity with respect to the random query. In the literature several techniques have been presented that propose optimal split rules when the data set can be embedded in a metric space [CP98]. In many real situations, however, an optimal strategy cannot be always adopted: (i) a data-driven split rule may not be feasible, because of a constraint in the physical localization of the data set, or (ii) it might be too expensive to be performed in practice, because the data set is too big or the scenario is too dynamic (insertions and deletion are too frequent).

2.3.2 Characterization

Data regions are a powerful tool to describe a data set and can be characterized at different levels, depending on the context where they are used. In general, a data region has a *physical representation*, a *region descriptor* and a *region statistics*.

A physical representation of a region is a list of the objects that are contained in it. In this thesis we refer to $\mathcal{D}_i$ as the physical representation of the i-th region of the data set.

The region descriptor $\mathcal{X}_i$, also referred to as its *geometrical representation*, gives an *exact* information about the content of the region and depends on the specific scenario.

Assume to have a relational database which contains the table STUDENTS whose physical representation is $\mathcal{D}_{stud}$: an example of a region descriptor for $\mathcal{D}_{stud}$ would be $\mathcal{X}_{stud} = \langle name, elems, l_1, u_1, \ldots, l_n, u_n \rangle$, where *name* is the name of the table, *elems* the number of elements and l_i, u_i refer, respectively, to the lower and upper bounds on the value of the i-th attribute.

The statistical characterization of a region $\mathcal{D}_i$, denoted as $stats(\mathcal{D}_i)$, summarizes the content of the region according to a representation scheme that depends on the specific scenario but also on other user-defined variables. There are three key elements:

Representation: This is the information that is chosen to effectively represent the region. Given the table STUDENTS, one is probably interested in storing the distribution of values over a few selected attributes.

Building: This is concerned with the technique according to which the information is collected. Typically, this might imply a pre-processing over the whole data set, sampling an already existing data base [GIS03] or even a passive acquisition process collecting the answers to the user queries [AGP99].

Compression: Given a specific representation, one usually needs to compress it in order to not let the statistics grow too much. A typical compression technique consists in using histograms to represent distributions or functions in general [Poo97]. If a set of elements needs to be represented, a popular technique is to use Bloom filters [Blo70].

2.3.3 Space Topology

During query processing it is not always possible to have complete knowledge about the topology of the space that is being explored, i.e.,, given the fact that objects are organized into regions, the query processor might not know all of them and might need to "explore" the space to discover new regions.

This is the case, for example, of a query formulated by a peer in a network: the peer, in fact, knows only its neighbors but there is a high probability that the query is better answered by some other peers in the network. The query, then, needs to navigate the "web" that can be built on top of this concept of neighborhood to find a good result. The same problem can arise in a centralized scenario as well, which is typically characterized by so many regions that they can hardly be maintained main memory. The typical solution consists in organizing them in a hierarchy, such that the regions at the higher levels are big enough to contain those at the lower level. The query, then, starts from the root of the structure and, at each steps, can descend into one or the other sub-tree [Gut84, CPZ97].

More precisely, we a *neighborhood* as follows:

Definition 2.8. *A region* $\mathcal{D}_* \in \{\mathcal{D}_1, \mathcal{D}_2, \ldots\}$ *is in the* neighborhood *of region* $\mathcal{D}_i$ *if there exists a* unidirectional link *from* $\mathcal{D}_i$ *to* $\mathcal{D}_*$. .

Depending on the size of the neighborhood of the regions in the space and on the resulting topology, we can distinguish among three different types of networks, which will be analyzed more in detail in the remaining chapters of the thesis:

Completely connected network Each region is neighbor of each other region, thus generating a completely connected network. This topology represents an ideal case, because it is equivalent to a situation of "complete knowledge" of the search space or, in a more traditional terminology, to a system where all regions' descriptors are in main memory. Though this might seem a far too simplifying scenario, it allowed us to derive important theoretical results;

Hierarchical network Each region has a variable number of neighbors (also called *child* regions) which contain objects drawn from their parent's data space. It is said, then, that a region "contains" its neighbors. Notice that the containment property is usually only virtual, because objects are not replicated and can only be found at the lower levels of the hierarchy (in the leaf regions). More precisely, there exists a *root* region $\mathcal{D}_0$ which is equivalent to the whole data set $\mathcal{D}$. The root has $|\mathcal{N}(\mathcal{D}_0)|$ neighbors such that $\mathcal{D}_0 = \bigcup_i \mathcal{D}_i$ and so on in a recursive fashion until the *leaf* regions are small enough to match the desired size. This scenario is typical of a centralized environment and of certain distributed systems (e.g.,, the DNS system) because of its efficiency.

Flat network Each region has a variable number of neighbors, but without any particular dependency between them. We will show a scenario in which each region chooses its neighbors according to some given criteria, however, it is still a very difficult scenario, because query processing can only rely on a very limited knowledge of the network topology. This paradigm is typically adopted by peer-to-peer environments, because of its flexibility to a fast changing environment.

2.4 Scheduling Strategies

The purpose of any technique for solving similarity queries is to speed up the evaluation by discarding irrelevant parts of the search space. Typically, this is done by exploring first the regions that are more promising with respect to the query, while trying to prove that

Algorithm 1 Algorithm for exact similarity search

Require: Problem $\langle \mathcal{X}, \mathcal{D}, sim, crit \rangle$, query $q \in \mathcal{X}$
Ensure: List $\mathcal{R}$ of results in $\mathcal{D}$

```
1: PQ ← [D_0, MaxSim(q, D_0)]
2: L_V ← ∅, R ← ∅, r ← ∞
3: while PQ ≠ ∅ do
4:     [D_i, MaxSim(q, D_i)] ← Dequeue(PQ)
5:     Insert(D_i, L_V)
6:     if MaxSim(q, D_i) > r then
7:         Explore(D_i)
8:         for all objects o_j in D_i do
9:             if crit(sim(q, o_j), R) = True then Insert(o_j, R) and update r
10:        for all regions D_n neighbor of D_i do
11:            if MaxSim(q, D_n) > r and ¬Present(D_n, L_V) then
12:                Enqueue(PQ, MaxSim(q, D_n))
13: return R
```

no better result can be found in the other regions. In this section we present a very general framework for similarity queries which does not depend on the specific implementation.

2.4.1 Exact Scheduling

We assume that there exists a function $\textsc{MaxSim}(q, \mathcal{D}_i)$ which associates a region $\mathcal{D}_i$ its *similarity upper-bound*, i.e.,, the maximum value of similarity which can be achieved by any object in $\mathcal{D}_i$. MaxSim typically refers to the geometrical representation of the region $\mathcal{D}_i$: if such a representation is not present, then a value of MaxSim cannot be derived for the regions in the space and all regions are equal to the query processor until it explores them. This means that the search becomes random. If, on the other hand, a MaxSim function is defined, it induces an ordering of the data regions and Algorithm 1 is optimal for the problem of exact similarity search.

The algorithm operates a best-first search over the regions that the query can reach, keeping them ordered in a priority queue PQ according to their *maximum similarity* from the query point. The list $\mathcal{L}_\mathcal{V}$ contains the regions already visited, to avoid that a query reaches the same region through different paths. The result $\mathcal{R}$ contains the set of objects that match the criterion *crit* and the *current similarity threshold* r contains the value of similarity below which no objects will be inserted in the result. If the query is a k-NN query, $\mathcal{R}$ contains the best-so-far k points, kept ordered by decreasing values of similarity with respect to q, and r is equal to the k-th distance value in $\mathcal{R}$. If the query is a range

query, $\mathcal{R}$ contains all objects whose similarity is above the threshold without the need to keep them ordered, and r is fixed to the threshold value of the query. The idea is that a region $\mathcal{D}_i$ is worth exploring only if the result can benefit from it, which means that $\textsc{MaxSim}(q, \mathcal{D}_i) > r$ must hold. In any other case $\mathcal{D}_i$ can be safely removed from `PQ`. We refer to this as *geometric pruning* [RKV95, CPZ97, HS03] because it only exploits the geometrical information about regions.

The optimality of Algorithm 1 was proved for the case of metric spaces [BBKK97, HS03], however it straightforwardly extends to the general problem of similarity search and follows from the observation that if $\textsc{MaxSim}(q, \mathcal{D}_i) < \textsc{MaxSim}(q, \mathcal{D}_j)$, then accessing $\mathcal{D}_j$ first can never lead to prune $\mathcal{D}_i$ without giving up the correctness of the result [BBKK97]. Also notice that as soon as a region extracted from `PQ` is pruned (line 3), then the algorithm can be immediately stopped, since all other regions in `PQ` will be pruned as well.

2.4.2 On-line Scheduling

In many real situation it is required to return fast, even if imprecise, results and eventually to continue processing the query in the background. The major issue that needs to be addressed for implementing an effective on-line algorithm concerns the criterion according to which regions are scheduled. In Section 2.2.2 we have introduced two control policies, namely a *cost-* and a *quality-* based policy. The key observation here is that, although visiting regions according to decreasing values of MaxSim guarantees that the exact result is determined with a minimum number of regions accesses [BBKK97], there is no proof that this extends to the case when the user is willing to accept an approximate result.

Intuitively, a scheduling policy Π aims at discovering as soon as possible objects which are similar to q. Ordering regions based on MaxSim, while it gives bounds on the maximum expected similarity, might be not enough precise to characterize the *expected similarity* that can be achieved by visiting that specific region. The idea, then, is that, given the geometric and statistical description of a data region $\mathcal{D}_i$, one can derive several indicators $\Psi_A, \Psi_B, \ldots$ [1], each assigning to $\mathcal{D}_i$ a value, $\Psi_A(q, \mathcal{D}_i), \Psi_B(q, \mathcal{D}_i), \ldots$, that depends on the specific query q and which has a correlation with the probability that $\mathcal{D}_i$ contains "good" objects with respect to Π. From now on, we will assume that a scheduling policy Π_A, based on indicator Ψ_A, given two regions $\mathcal{D}_i$ and $\mathcal{D}_j$, chooses to explore $\mathcal{D}_i$ first, if $\Psi_A(q, \mathcal{D}_i) \leq \Psi_A(q, \mathcal{D}_j)$.

All these considerations are integrated in the on-line Algorithm 2 for similarity search, which, with respect to Algorithm 1, shows the following modifications:

[1] In general, we write $\Psi_j(q, \mathcal{D}_{\mathcal{D}_i})$ to indicate that the value of Ψ_j for the query q depends on either the geometrical description of $\mathcal{D}_i$ or on its statistics $stats(\mathcal{D}_i)$.

Algorithm 2 Algorithm for on-line similarity search

Require: Problem $\langle \mathcal{X}, \mathcal{D}, sim, crit \rangle$, query $q \in \mathcal{X}$, scheduling policy Π
State: current approximate results $\widetilde{\mathcal{R}}$, current search radius r, current `PQ`
Ensure: List $\widetilde{\mathcal{R}}$ of approximate results in $\mathcal{D}$

```
1: PQ ← [D_0, MaxSim(q, D_0)]
2: L_V ← ∅, R ← ∅, r ← ∞
3: while ¬PQ ≠ ∅ ∨ Suspend(q, R̃) do
4:     [D_i, stats(D_i)] ← Dequeue(PQ, Π)
5:     Insert(D_i, L_V)
6:     if MaxSim(q, D_i) > r then
7:         Explore(D_i)
8:         for all objects o_j in D_i do
9:             if crit(sim(q, o_j), R) = True then Insert(o_j, R) and update r
10:        for all regions D_n neighbor of D_i do
11:            if MaxSim(q, D_n) > r and ¬Present(D_n, L_V) then
12:                Enqueue(PQ, [q, D_n], Π)
13: return R̃
```

1. The `Suspend`$(q, \widetilde{\mathcal{R}})$ Boolean method at line 3 checks whether the current termination condition is satisfied, i.e., `Suspend` encapsulates the specific control policy enforced at that moment (e.g., $\widetilde{\mathcal{R}}$ guarantees the desired quality or the maximum execution cost has been reached).

2. The scheduling policy Π is passed as argument to the `Dequeue` and `Enqueue` methods (lines 4 and 12, respectively), which manage the priority queue.

3. The value of the indicator $\Psi\left(q, \mathcal{D}_i\right)$ for each region $\mathcal{D}_i$ is computed by the schedule within the priority queue.

4. The priority queue, the current approximate result, and the search radius are maintained as an internal state that allows the execution to be correctly resumed at the point it left.

An interesting property of Algorithm 2 is that it can be seen as a generalization of the problem of exact similarity search. In particular, Algorithm 1 can be simply obtained by letting $\Psi_{\Pi}\left(q, \mathcal{D}_i\right) \equiv \textsc{MaxSim}(q, \mathcal{D}_i)$.

2.5 Similarity search in Metric Spaces

In this section we introduce the notion of similarity search in metric spaces, which is based on the property that there exists a function $f_m : \mathcal{D} \rightarrow \mathcal{X}$ which maps objects of the data set into points of the space $\mathcal{X}$, such that the co-domain of $\mathcal{D}$ is a subset of $\mathcal{X}$ (i.e., $f_m(\mathcal{D}) \subseteq \mathcal{X}$). The following definitions are very important because the problem of similarity search assumes a well-defined geometrical meaning.

Assume to have a (possibly infinite) set $\mathcal{X}$ and a function d defined as $d : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}^+$. The pair $(\mathcal{X}, d)$ is a *metric space* and d is a *metric distance* if for all $x, y, z \in \mathcal{X}$ the following properties hold:

$$d(x, x) = 0 \qquad \text{(Reflexivity)} \tag{2.6}$$

$$d(x, y) = d(y, x) \qquad \text{(Symmetry)} \tag{2.7}$$

$$d(x, y) + d(y, z) \geq d(x, z) \qquad \text{(Triangle inequality)} \tag{2.8}$$

A problem of similarity search for the data set $\mathcal{D}$ is defined in the metric space $(\mathcal{X}, d)$ if, for any two objects $o_i, o_j \in \mathcal{D}$ whose projections in the space $\mathcal{X}$ are $f_m(o_i)$ and $f_m(o_j)$, then it holds $sim(o_i, o_j) \propto d(f_m(o_i), f_m(o_j))^{-1}$.

According to Property 2.6, then, the highest similarity a point x can achieve in the space is with itself, which corresponds to a distance of 0; Property 2.7 states the the similarity between any two points x and y is reciprocal, i.e., it does not depend on the viewpoint. Finally, the third property is the triangle inequality. The importance of this property, that will be fully exploited in the rest of the thesis to reduce the size of the search space, is easily explained through an example. Given a query q and two objects x, y assume we know the distances $d(q, x) = a$ and $d(x, y) = b$, then we can infer that $d(q, y) \geq |a - b|$. This means that we might know in advance that y is not suitable to be inserted in the result if $|a - b|$ is outside the current search radius (if we are processing a range query) or if we have already found k objects whose distance is lower (with a k-NN query).

Typically, a data region $\mathcal{D}$ in a metric space is defined in terms of a *ball region* $\mathcal{B}_c(r)$ where c is the *center* of the region and r is its *radius* and delimits the area in which an object must be contained in order to be part of that region. In a metric space, the concept of coordinates cannot be defined, thus only the relative distance among objects can be used. This is the reason why it is more difficult to define complex shapes for data regions which, even if they can be tuned to be more more efficient, are also more costly to maintain (see discussion in [CNBYM01]).

Finally, consider the algorithms for exact and on-line search presented in Section 2.4. In general, since the problem of maximizing the similarity between two objects in a

metric space is equivalent to minimize their distance, the function $\textsc{MaxSim}(q, \mathcal{D}_i)$ can be substituted by its metric counterpart $\textsc{MinDist}(q, \mathcal{X}_i)$, which returns the minimum distance between a query q and any point $x \in \mathcal{X}_i$, where $\mathcal{X}_i$ is the representation of $\mathcal{D}_i$ in $\mathcal{X}$. Algorithms 1 and 2, then, aim at minimizing the distance rather than maximizing the similarity.

2.6 Related Work

In this section we review some previous work on similarity search in metric spaces that is relevant to our scenario. In particular, we distinguish between *data structures* for similarity search and approximate similarity search techniques.

2.6.1 General Data Structures

In the literature the number of data structures for similarity search in metric space is relatively low, due to the difficulty of defining efficient and effective tools an techniques for such a complex scenario. Here we only review a few data structures and refer to [CNBYM01] for a more complete survey.

One of the first data structures that were proposed in the literature is the Vantage Point Tree [Yia93] (VPT). The VPT is a binary tree based on *vantage points* which partition the space according to the associated radius as follows: pick a vantage point p_0 of radius r_0, then all objects o from the data set such that $d(p_0, o) \leq r_0$ are in the left subtree, the others in the right subtree. The process is repeated until each node as only one object.

The Geometric Near-Neighbor Access Tree [Bri95] (GNAT) is a static data structure for main memory. At the first step, m *split points* are chosen such that any data object is associated to the closer split point in the space. Those regions which have a too high number of data objects are recursively partitioned, until a suitable number of objects is contained in each region. The result is an unbalances hierarchy with non-overlapping regions. The bigger problem with GNAT is how the split points should be chosen. The basic idea presented in [Bri95] works as follows: pick the first point at random and the second as far as possible from the first one, pick the third as far as possible from the first two and so on.

The M-Tree [CPZ97] operates is data structure that can be used with the secondary memory. It can be seen as a generalization of the R-tree [Gut84], but with the major difference that the data regions are ball region and, due to the absence of ccoordinates an object bbelongsto those regions whose distance from the center is lower than their

radius.In general, an M-tree is built bottom-up: the space is split in a certain number of regions, such that each region is not bigger than a memory page (thus, they can be stored as an atomic element). At the higher level, the space is split again in bigger regions which contain the lower level regions rather than data objects. This way we end up with a single region which represents the whole space.

2.6.2 Approximate Similarity Search

In [GR00] Goldstein and Ramakrishnan introduce the Probabilistic Sphere Tree (P-Sphere), a completely connected network of overlapping data regions which also accept replication at the data level. This data structure allows one to perform an approximate nearest neighbor search as follows: given a query q, the region $\mathcal{B}_c(r)$ whose center c is closer to q is searched first. Let the local nearest neighbor be p, than if the ball region $\mathcal{B}_q(d\,(q,p))$ is completely contained in the explored region, p is the *provably correct* nearest neighbor and can be returned to the user, otherwise it is *not provably correct* and P-Sphere cannot retrieve the correct result. The claim in [GR00] is that the fraction of the times that this happens (i.e., of not finding a provably correct result) can be tuned at construction time by making the regions large enough.

A more flexible technique to approximate similarity search in large data bases is the PAC approach [CP00a], which constitutes the starting point of this thesis. The idea is to let the user specify a pair (ϵ, δ), where ϵ is the maximum error allowed (cf. Section 2.2.1) and δ is a confidence value which represents the probability that the error will really be lower than ϵ. A data set is described through the (global) distribution $G\,(r)$ of the distance of the nearest neighbor from a random query. Given a query q and a confidence value δ, then, with probability δ, the distance $d_q\,(NN)$ between the query and its nearest neighbor will be not lower than $r_\delta = G^{-1}(\delta)$. Then, the stop can be stopped as soon as an object at a distance lower than r_δ is found (or $(1+\epsilon)r_\delta$ if an error ϵ is allowed).

Chapter 3

Completely Connected Networks

Traditional strategies for supporting similarity queries in complex database systems have been demonstrated to be optimal with respect to the problem of finding the "best" answer. However, they experience high execution costs, due to the intrinsic difficulty of the problem. In this chapter we focus on the ideal case of a *completely connected network*, i.e.,we assume that each region has all other regions as neighbors, such that, at each step, the most promising region with respect to the scheduling policy can be accessed.

In Section 3.1 we characterize our scenario in terms of the classification presented in the previous chapter. In Section 3.2 we present a general cost model for on-line similarity queries which exploits information on the distribution of the distance of a query from its nearest neighbor. This distribution can be estimated in several ways and we present four of them in Section 3.3. In Section 3.4 we introduce the notion of *optimal-on-the-average* schedules for the problem of approximate similarity search, an approach which allows one to strongly reduce execution costs while achieving probabilistic guarantees on the quality of the result and we present a relevant approximation of such a schedule which exploits information given by simple indicators (Section 3.5. Finally we present the experimental results in Section 3.6 to show the effectiveness of our method of analysis.

3.1 The Scenario

Our scenario consists of a completely connected network which is an instance of the framework presented in Chapter 2. Its main characteristics can be summarized as follows:

- Our theoretical model is developed for 1-NN bounded queries, however we evaluate it with k-NN and bounded k-NN queries;
- Refer to a quality-based control policy and let the threshold be θ. This is equivalent to a bounded query of radius θ, i.e., the search is stopped whenever the approximate

result $\widetilde{\mathcal{R}}$ contains an object o such that $d(q, o) \leq \theta$;

- The network is built adopting a data-driven split rule, which clusters data objects based on their distance. Our cost model, however, does not depend on the distribution of the data inside each region, therefore it applies to any split-rule.

- A data region is equivalent to a ball region $\mathcal{B}_c(r) \subseteq \mathcal{X}$, of center c and radius r, because, as we have anticipated in Section 2.5, this is the most natural way to describe data regions in a metric space.

3.2 Probabilistic Models for On-line Queries

In this section, we investigate the problem of defining a model for on-line nearest neighbor queries which are executed according to a cost-based control policy (cf. Section 2.2.2). This model will be used to predict the cost of a query which aims at finding a result whose distance from the query point is lower than a user-defined threshold θ and will be further exploited to derive optimal on the average scheduling policies. Finally, at the end of the section, (cf. Paragraph 3.2.2 we will present how an analogous model can be defined to predict the behavior of on-line queries which adopt a quality-based control policy.

A nearest neighbor query can be stopped as soon as it finds a data region which contains an object "sufficiently" close to the query point, i.e., such that $d_{\mathcal{D}_i}(q) = \min_{o_j \in \mathcal{D}_i} \{d(q, o_j)\}$ is lower than θ. A data region $\mathcal{D}_i$, then, can be characterized in terms of the probability $\Pr\{d_{\mathcal{D}_i}(q) \leq \theta\}$ that $\mathcal{D}_i$ stores a solution for the *random* query q.

We start by considering the case of 1-NN search and 2-levels trees, since this is the scenario more amenable to be formally characterized. Then, the index tree consists of a root and of M leaf regions, and a schedule Π can thus be viewed as a permutation of the set $\{1, \ldots, M\}$:

$$\Pi = (\Pi_1, \Pi_2, \ldots, \Pi_i, \ldots, \Pi_M) \tag{3.1}$$

where $\mathcal{D}_{\Pi_i}$ is the leaf that schedule Π will fetch at step i. Clearly, the position of each region $\mathcal{D}_i$ in the schedule depends on the query and on the $stats(\mathcal{D}_i)$ statistics of the region.

3.2.1 Cost-based Queries

The cost of a query executed according to a cost-based control policy is measured as the number of regions that need to be accessed in order to satisfy the query criteria. A query is a pair (q, θ), where q is the query point and θ is the (fixed) search radius. The upper-bound to the cost of this query is equal to the number of regions in the system,

because it might happen that either the solution is found only in the last region visited or a solution is not found at all. In the following we have made the assumption that at least one solution to the query exists in the data set.

Let $p_{stop}(c, \theta; \Pi) = \Pr\{\min_{i \leq c}\{d_{\Pi_i}(\mathbf{q})\} \leq \theta\}$ be the probability that the search algorithm, using schedule Π, will find, in no more than c steps, an object whose distance from the random query $\mathbf{q}$ is not higher than θ, i.e.,

$$\begin{aligned} p_{stop}(c, \theta; \Pi) &= \Pr\left\{\min_{i \leq c}\{d_{\Pi_i}(\mathbf{q})\} \leq \theta\right\} \\ &= 1 - \prod_{i=1}^{c} \Pr\{d_{\Pi_i}(\mathbf{q}) > \theta\} = 1 - \prod_{i=1}^{c} (1 - \Pr\{d_{\Pi_i}(\mathbf{q}) \leq \theta\}) \end{aligned} \tag{3.2}$$

where $1 - \Pr\{d_{\Pi_i}(\mathbf{q}) \leq \theta\}$ in Equation 3.2 represents the event that the i-th visited region does not contain a solution to the query. Then, the expected cost $Cost[\mathbf{q}, \Pi; \theta]$ of schedule Π, is

$$\begin{aligned} Cost[\mathbf{q}, \Pi; \theta] &= \sum_{c=1}^{M} c \cdot [p_{stop}(c, \theta; \Pi) - p_{stop}(c-1, \theta; \Pi)] \\ &= \sum_{c=1}^{M} c \cdot p_{stop}(c, \theta; \Pi) - \sum_{c=0}^{M-1} (c+1) \cdot p_{stop}(c, \theta; \Pi) \\ &= \sum_{c=1}^{M} c \cdot p_{stop}(c, \theta; \Pi) - \sum_{c=0}^{M-1} c \cdot p_{stop}(c, \theta; \Pi) - \sum_{c=0}^{M-1} p_{stop}(c, \theta; \Pi) \\ &= M \cdot p_{stop}(M, \theta; \Pi) - \sum_{c=1}^{M-1} c \cdot p_{stop}(c, \theta; \Pi) = M - \sum_{c=1}^{M-1} c \cdot p_{stop}(c, \theta; \Pi) \end{aligned} \tag{3.3}$$

where $p_{stop}(c, \theta; \Pi) - p_{stop}(c-1, \theta; \Pi)$ be the probability that the object is found in *exactly* c steps. In Equation 3.3 the first two passages are due to arithmetic manipulation of the indices in the respective sums. In the third line, the first two sums erase each other, leaving the term $M \cdot p_{stop}(M, \theta; \Pi)$, which is equal to M, because we have assumed that the query always find a solution in the data set (thus, $p_{stop}(M, \theta; \Pi) = 1$).

Practical hints on how probability $\Pr\{d_{\Pi_i}(\mathbf{q}) \leq \theta\}$ should be computed are given in Section 3.3, while in Section 3.6 we present experimental results on the comparison between predicted and effective costs of different schedules.

Cost Model and the Problem of Geometrical Pruning

The problem with the cost model presented in the previous section is that it does not take geometrical pruning into consideration. In a real setting, since we consider bounded nearest neighbor queries, all regions $\mathcal{D}_i$ for which $\textsc{MinDist}(q, \mathcal{D}_i) > \theta$ are removed from the

search space, independently from the probability $\Pr\{d_{\mathcal{D}_i}(\mathbf{q}) \leq \theta\}$. Geometrical pruning is not taken into consideration in the model proposed by Equation 3.3. To understand this point consider the following Example:

Example 3.1. *Assume that, at step c^*, the stopping probability as defined by Equation 3.2 is $p_{stop}(c^*, \theta; \Pi)$ and that, at step c^*+1 is chosen a region Π_{c^*+1} which lower-bound is grater than θ (i.e., $\textsc{MinDist}(q, \Pi_{c^*+1}) > \theta$). In general, $p_{stop}(c^*+1, \theta; \Pi) \geq p_{stop}(c^*, \theta; \Pi)$, even if, in this specific example, there is no chance of finding an object at a distance lower than θ in Π_{c^*+1}. The effect is that the estimated cost increases (cf. Equation 3.3).*

The reason why this happens is that the stopping probability $p_{stop}(c, \theta; \Pi)$ does not consider the region's geometry. We will show in Section 3.6.1 that this results in a sensible overestimation of the final cost of a schedule Π. A solution consists in conditioning the stopping probability to the value of MinDist of the region chosen at step c. Equation 3.2, then, can be rewritten as:

$$\begin{aligned} p'_{stop}(c, \theta; \Pi) &= \Pr\left\{ \min_{i \leq c} \{d_{\Pi_i}(\mathbf{q})\} \leq \theta \mid \textsc{MinDist}(\mathbf{q}, \Pi_i) \right\} \\ &= \begin{cases} p_{stop}(c, \theta; \Pi), & \text{if } \textsc{MinDist}(\mathbf{q}, \Pi_i) \leq \theta \\ 0, & \text{otherwise} \end{cases} \end{aligned} \tag{3.4}$$

3.2.2 Quality-based Queries

In this setion we briefly introduce the model that estimates the *expected quality* that a schedule Π achieves if the user specifies a threshold on the number of regions accessed rather then on the quality. In other words, given the maximum number of region c, we want to estimate

$$\begin{aligned} E\left[\min_{i \leq c} d_{\Pi_i}(\mathbf{q})\right] &= \int_0^{d^+} \theta \frac{dp_{stop}(c, \theta; \Pi)}{d\theta} d\theta = |\theta p_{stop}(c, \theta; \Pi)|_0^{d^+} - \int_0^{d^+} p_{stop}(c, \theta; \Pi)\, d\theta \\ &= d^+ - \int_0^{d^+} p_{stop}(c, \theta; \Pi)\, d\theta \end{aligned} \tag{3.5}$$

where d^+ is the maximum possible distance value and the equation is interpreted as follows: we compute the expected value of the continuous variable θ as the integral over the domain $[0, d^+]$, weighted by the probability $\frac{dp_{stop}(c,\theta;\Pi)}{d\theta}$ that an object at a distance θ from the query is found in c steps.

3.3 Regions Characterization

To compute Equations 3.3 and 3.5 we need to estimate the stopping probability $p_{stop}(c, \theta; \Pi)$ with respect to a random query $\mathbf{q}$. This means knowing the *joint distance distribution of*

the NN local to each region in the data set $G_{1,\ldots,M}(r_1,\ldots,r_M)$, computed with respect to $\mathbf{q}$. Then,

$$\Pr\{d_{\mathcal{D}_i}(q) \leq \theta\} = G_{1,\ldots,M}\left(d_{\mathcal{D}_i}(q) \leq \theta | \{d_{\mathcal{D}_j}(\mathbf{q})\}, \forall j \neq i\right) \tag{3.6}$$

which is clearly too hard to compute (and to store in practice), because it requires the knowledge of a far too complex joint distribution.

In Section 3.3.2 we propose four different approximations of Equation 3.6 which are based on two simple observations:

- It is not realistic to assume that the values $\{d_{\mathcal{D}_j}(\mathbf{q})\}, \forall j \neq i$ are known, because it would require to have already explored all other regions to be able to collect such knowledge;
- On the other hand, it is realistic to assume that the geometric descriptors of the data regions are available, because typically they are known before actually visiting the region.

3.3.1 Regions Indicators

In this section we study in detail the concept of *indicators*, that have been already introduced in Section 2.4 to choose a scheduling strategy.

An indicator $\Psi(q, \mathcal{D}_i)$ is an *observation of a data region* $\mathcal{D}_i$ made from q's point of view, which can be used as a *representative* for $\mathcal{D}_i$ to characterize Equation 3.6. We use this term because $\Psi(q, \mathcal{D}_i)$ indicate the position of region $\mathcal{D}_i$ with respect to q.

In general, given a point (the query q) and a ball (the region $\mathcal{D}_i$) in a metric space, the most common and intuitive indicators of the relative position of q and $\mathcal{D}_i$ are the following:

$\Psi(q, \mathcal{D}_i) = \mathbf{MinDist}(q, \mathcal{D}_i)$ It is the lower-bound on the distance between the query and the region. Given the region's descriptor $\mathcal{B}_{c_i}(r_i)$, it is computed as $\textsc{MinDist}(q, \mathcal{D}_i) = \max\{d(q, c_i) - r_i, 0\}$, i.e., as the distance between the query and the center of the region minus its radius.

$\Psi(q, \mathcal{D}_i) = \mathbf{MinMaxDist}(q, \mathcal{D}_i)$ It is the upper-bound on the distance between the query and the region. It is computed as $\textsc{MinMaxDist}(q, \mathcal{D}_i) = d(q, c_i) + r_i$.

$\Psi(q, \mathcal{D}_i) = \mathbf{MaxDist}(q, \mathcal{D}_i)$ It is the maximum distance between the query and any other object in the region. The center of a region is often a real object, therefore $\textsc{MaxDist}(q, \mathcal{D}_i) = d(q, c_i)$. If this is not the case $\textsc{MaxDist}(q, \mathcal{D}_i) = \textsc{MinMaxDist}(q, \mathcal{D}_i)$.

Other indicators can be derived, such as the average between the MinDist and MinMaxDist (or MinDist and MaxDist), however, in the rest of the dissertation we will concentrate on these three, because they are the most relevant.

3.3.2 Nearest Neighbor Distance Distribution

In this section we show how the distribution of the distance of the nearest neighbor $\Pr\{d_{\mathcal{D}_i}(q) \leq \theta\}$ given in Equation 3.6 can be simplified in order to be efficiently implemented.

In general, the idea behind the introduction of indicators is that they are observable characteristics of a data region, which do not require to effectively visit it to be collected. Thus, they can be used to substitute the events $\{d_{\mathcal{D}_j}(\mathbf{q})\}$ in Equation 3.6 to obtain the approximation

$$G_{1,\ldots,M}\left(\theta|\left(d_{\mathcal{D}_1}(\mathbf{q}),\ldots,d_{\mathcal{D}_M}(\mathbf{q})\right)\right) \approx G_{1,\ldots,M}\left(\theta|\left(\psi_1,\ldots,\Psi_M\right)\right) \tag{3.7}$$

$$\approx G_i\left(\theta|\psi_i\right) \tag{3.8}$$

where $\psi_i = \Psi(\mathbf{q}, \mathcal{D}_i)$. Notice that Equation 3.7 is still too complex to be maintained by each region, thus we can further approximate it by assuming *independence among regions* to obtain Equation 3.8. $G_i(\theta|\psi_i)$ is the distance distribution of the NN in the region $\mathcal{D}_i$ with respect to the random query $\mathbf{q}$, conditioned to the observed value ψ_i of the indicator Ψ.

We start by distinguishing among two classes of distance distributions, the *query-independent* and the *query-dependent* ones.

Definition 3.1 (Query-independent distance distribution)**.** *Let q_1 and q_2 be two queries for which $\psi_1 = \Psi(q_1, \mathcal{D}_i)$ and $\psi_2 = \Psi(q_2, \mathcal{D}_i)$. Then a distance distribution is* query-independent *if $\Pr\{d_{\mathcal{D}_i}(q) \leq \theta|\psi_1\} = \Pr\{d_{\mathcal{D}_i}(q) \leq \theta|\psi_2\}$ for any ψ_1, ψ_2.*

Definition 3.2 (Query-dependent distance distribution)**.** *Let q_1 and q_2 be two queries for which $\psi_1 = \Psi(q_1, \mathcal{D}_i)$ and $\psi_2 = \Psi(q_2, \mathcal{D}_i)$. Then a distance distribution is* query-dependent *if $\Pr\{d_{\mathcal{D}_i}(q) \leq \theta|\psi_1\} = \Pr\{d_{\mathcal{D}_i}(q) \leq \theta|\psi_2\}$ iff $\psi_1 = \psi_2$.*

The main difference between the previous classes is that the query-independent distribution does not depend on the relative position of the query with respect to the data region. In other words, Equation 3.8 is not conditioned to the value ψ_i, which means that all queries *see* the same probability $G_i(\theta|\psi_i)$ for region $\mathcal{D}_i$ (or queries are statistically identical from the regions' perspective).

A distance distribution can be further classified as *novelty-based* or *simple*. Let $\overline{d_i}(q) = \{d_{\mathcal{D}_i}(q) > \theta\}$ be the boolean event that a query of radius θ does not find a solution in

region $\mathcal{D}_i$, a novelty-based distribution aims at balancing the hypothesis of Independence among regions introduced by Equation 3.8. This is done by introducing the evidence that at step i "all previously visited regions have not solved the query". The novelty-based distributions are defined as follows:

Definition 3.3 (Novelty-based distance distribution)**.** *The* query-independent novelty-based distance distribution *is defined as*

$$\Pr\{d_{\mathcal{D}_i}(q) \leq \theta\} \approx G_{\Pi_1,\ldots,\Pi_M}\left(d_{\mathcal{D}_i}(q) \leq \theta \mid \overline{d_1}(q), \ldots, \overline{d_{i-1}}(q)\right) \tag{3.9}$$

and the query-dependent novelty-based distance distribution *is defined as*

$$\Pr\{d_{\mathcal{D}_i}(q) \leq \theta\} \approx G_{\Pi_1,\ldots,\Pi_M}\left(d_{\mathcal{D}_i}(q) \leq \theta \mid \psi_i,\ \overline{d_1}(q), \ldots, \overline{d_{i-1}}(q)\right) \tag{3.10}$$

Definition 3.4 (Simple distance distribution)**.** *The* query-independent simple distance distribution *is defined as*

$$\Pr\{d_{\mathcal{D}_i}(q) \leq \theta\} \approx G_{\Pi_i}(d_{\mathcal{D}_i}(q) \leq \theta) = G_{\Pi_i}(\theta) \tag{3.11}$$

and the query-dependent simple distance distribution *is defined as*

$$\Pr\{d_{\mathcal{D}_i}(q) \leq \theta\} \approx G_{\Pi_i}(d_{\mathcal{D}_i}(q) \leq \theta|\Psi(q, \mathcal{D}_i)) = G_{\Pi_i}(\theta|\psi_i) \tag{3.12}$$

3.4 Optimal-on-the-Average Schedules

In this section we consider schedules that, although not necessarily optimal for a query, are *optimal on the average*. A schedule Π that is either cost- or quality-optimal on the average has the property that no other schedule Π' performs better than it when a random query is considered. Considering the control policies in Section 2.2.2 and assuming that the cost is evaluated in terms of number of region accesses, there are two possible scenarios to consider:

1. Given a threshold value θ, minimize the average cost, and
2. given a cost limit c, minimize the average approximate NN distance.

Optimal on the average schedules require one to compute the probability $p_{stop}(c, \theta; \Pi)$ defined by Equation 3.2

Theorem 3.1 (Optimal on the average schedule)**.** *The optimal schedule is incrementally obtained by choosing at each step j the region that, among the unread regions $\mathcal{D}_i$, maximizes the quantity* $\Pr\{d_{\Pi_i}(\mathbf{q}) \leq \theta\}$.

Proof. When a distance threshold θ is given, the expected cost for processing a random query $\mathbf{q}$ with schedule Π can be expressed, according to Equations 3.3 and 3.2, as:

$$Cost\left[\mathbf{q},\Pi;\theta\right] = M - \sum_{c=1}^{M-1} c \cdot p_{stop}\left(c,\theta;\Pi\right) = M - \sum_{c=1}^{M-1}\left(1 - \prod_{i=1}^{c}\left(1 - \Pr\left\{d_{\Pi_i}\left(\mathbf{q}\right) \leq \theta\right\}\right)\right)$$
$$= 1 + \sum_{c=1}^{M-1}\prod_{i=1}^{c}\left(1 - \Pr\left\{d_{\Pi_i}\left(\mathbf{q}\right) \leq \theta\right\}\right) \tag{3.13}$$

which amounts to choose, at each step j, the region maximizing $\Pr\left\{d_{\Pi_i}\left(\mathbf{q}\right) \leq \theta\right\}$ (similar arguments apply to Equation 3.5, when one wants to minimize the approximate distance of the NN). □

3.4.1 Query-Independent Schedules

From Theorem 3.1, a query-independent optimal schedule maximizes, at each step, the probability defined by Equation 3.9 (novelty-based schedule) or by Equation 3.11 (simple schedule).

These schedules do not depend on the query point q and, for a given θ, the regions are always explored according to the same (static) ordering. From a practical point of view, however, it is an interesting, though simple, case study that has the advantage that can be implemented by simply adding each region a link which points to the following step in the chain. The first region accessed has the highest probability of solving the query or, from a probabilistic point of view, it can solve, in one single step, more queries than any other region (on average). The second region, if we adopt a novelty-based schedule, is the one which solves the highest number of queries *that did not find a solution in the first step.*

This schedule is an elegant solution to those scenarios that requires the system to return a fast, even if approximate, response.

3.4.2 Query-Dependent Schedules

A query-dependent optimal schedule maximizes, at each step, the probability defined by Equation 3.10 (novelty-based schedule) or by Equation 3.12 (simple schedule).

In order to implement a query-dependent simple schedule based on the indicator Ψ, one has to discretize the indicator domain $[\psi_{min}, \psi_{max}]$ and store, for each region, an histogram $hist_{\psi_i}(\theta)$ for each value ψ_i. Given a page size of approximately $8KB$ (cf. [CPZ97]), 10 possible values of Ψ and 20 different values of θ (and assuming $4B$ for each cell of the histograms), the space overhead is $800B$, i.e., 10% of the page size, which is still affordable.

Implementing a query-dependent novelty-based schedule is more complex because Equation 3.10 basically relies on exploiting the joint distribution $G_{1,\ldots,M}(\theta)$.

Our implementation of the schedule is based on a sample set of queries which are used to compute the probability $G_i\left(\theta \mid \psi_i,\ \overline{d_1}(q), \ldots, \overline{d_{i-1}}(q)\right)$ at query time. Given a discretization of the indicator's domain, each region $\mathcal{D}_i$ stores, for each possible value ψ_j of the indicator, the queries from the sample set for which $\Psi(q, \mathcal{D}_i) = \psi_j$ holds. These queries are partitioned in two sets and in the first set there are those that can be solved exploring the region, while in the other those that can not be solved. The probability $G_j(\theta|\psi_j)$ can be estimated as $\frac{|solved|}{|solved|+|unsolved|}$. To take novelty into account, each time the schedule chooses a region $\mathcal{D}_i$, the queries from the sample set that are solved by $\mathcal{D}_i$ has to be removed from all other regions' solved lists.

3.5 Schedules Based on Indicators

In general, maintaining a set of statistics $stats(\mathcal{D}_i)$ for a region $\mathcal{D}_i$ may be too expensive, because of a limited storage space or because they cannot be compute efficiently. In this section we investigate alternative scheduling strategies that do not explicitly rely on statistical descriptions stored at each data region, but only assume that it is possible to conduct a preliminary off-line study on the data set.

In the remainder of the section we presents formal results that show that for regions whose data objects are uniformly distributed within them, if we consider the geometrical indicators presented in Section 3.3.1, then the *lower* is the value of the indicator for a given data region, the *higher* is the probability that $d_{\mathcal{D}_i}(q) \leq \theta$.

The direct consequence is that a query-dependent schedule, which maximizes the probability $\Pr\{d_{\mathcal{D}_i}(q) \leq \theta\}$, can be approximated by one which minimizes the value of $\Psi(q, \mathcal{D}_{\mathcal{D}_i})$.

These results are generalized to any data distribution in a region by taking into account the entropy of the event which models the value of an indicator with respect to a random query. These results are presented in Section 3.5.2.

3.5.1 The Case of Uniform Distribution

Consider the optimal query-dependent schedules which maximize, at each step i, the probability $G_i(\theta|\psi_i)$ (eventually conditioned to the previously explored regions). In this formula the event that a region $\mathcal{D}_i$ satisfies $d_{\mathcal{D}_i}(q) \leq \theta$ is correlated to the event of observing, for the same region, an indicator Ψ whose value is equal to ψ_i.

In general, when we have two events A and B and the conditioned probability $P(A|B)$,

if A and B are highly correlated, then whenever B is verified it is very likely that also A will be verified. Under these conditions, a first-order approximation of the conditional probability $P(A|B)$ would be $P(A|B) \approx P(B)$.

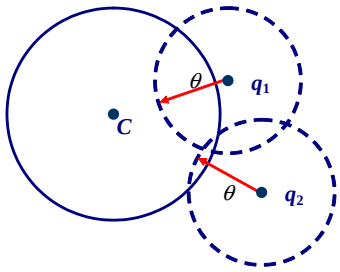


Figure 3.1: Data region and uniform distribution of objects.

Theorem 3.2. *Let $\mathcal{B}_C(r_C)$ be a ball region in a metric space whose data is distributed uniformly within the region and let q_1 and q_2 be two queries of radius θ, such that $lb_1 = \textsc{MinDist}(q_1, \mathcal{D}_C)$ and $lb_2 = \textsc{MinDist}(q_2, \mathcal{D}_C)$. If $lb_1 \leq lb_2$, then $\Pr\{d_{\mathcal{D}_C}(q_1)\,|lb_1\} \geq \Pr\{d_{\mathcal{D}_C}(q_2)\,|lb_2\}$.*

Proof. The proof derives from simple geometrical considerations: assume q_1 and q_2 are two random queries such that $lb_1 = lb_2$, then they would share the same *viewpoint* on $\mathcal{D}_C$ (and eventually the same distance from the NN) disregarding their effective position in the space. Due to the metric properties of the space, It naturally follows that if one of the two queries lies farther from the region, then it is also farther from the nearest neighbor. □

Notice that analogous considerations are valid for the other indicators presented in Section 3.3.1 (MinMaxDist and MaxDist). From Theorem 3.2 we derive the following corollary, which gives us indications on how to schedule data regions based on indicators:

Corollary 3.1. *Let $\mathcal{D}_1$ and $\mathcal{D}_2$ be two regions with uniform data distribution and let $\psi_1 = \Psi(\mathbf{q}, \mathcal{D}_1)$ and $\psi_2 = \Psi(\mathbf{q}, \mathcal{D}_2)$ be the only available information about $\mathcal{D}_1$ and $\mathcal{D}_2$ with respect to a random query $\mathbf{q}$. If $\psi_1 \leq \psi_2$, then $\Pr\{d_1(\mathbf{q}) \leq \theta\} \geq \Pr\{d_2(\mathbf{q}) \leq \theta\}$.*

Proof. Without any other information, we can assume that the radii r_1 and r_2 of $\mathcal{D}_1$ and $\mathcal{D}_2$, respectively, are equal. The proof, then follows from Theorem 3.2. □

3.5.2 Entropy-based Analysis

In Section 3.5 we have assumed a uniform distribution of the objects in the data regions. In this section we generalize the previous results and present a technique which, without relying on any particular data distribution, constitutes a power tool to estimate the

correlation existing between two events: the value assumed by an indicator Ψ in a data region with respect to a random query and the probability that the region has a solution for that query.

We study the *entropy associated to an indicator* Ψ, i.e., the relationship between the value $\Psi\left(q, \mathcal{D}_i\right)$ computed with respect a random region $\mathcal{D}_i$ and the probability that "the threshold is reached exploring $\mathcal{D}_i$". More precisely, adopting an information-theoretic approach, we consider how much information an indicator Ψ can provide us about the probability of finding an object at distance lower than θ from q.

This analysis should be propedeutical to choosing, among all the available indicators, the one that maximizes such information. In a more sophisticated scenario, one might even partition the set of all possible queries to derive the *optimal indicator* for each specific partition. This further characterization is, however, out-of-the-scope of this work.

The following analysis relating an indicator with NN information assumes that we will not incur substantial errors if we try to understand the relationship between the values of an indicator Ψ and the global NN distribution, $G(\cdot)$, rather than the distributions of the M regions. Consider the probabilistic event $S\left(\theta\right) =$ "$d_{\mathcal{D}_i}\left(q\right) \leq \theta$ for a random region $\mathcal{D}_i$", its entropy is defined as:

$$H(S\left(\theta\right)) = -G\left(\theta\right)\log_2 G\left(\theta\right) - \left(1 - G\left(\theta\right)\right)\log_2(1 - G\left(\theta\right)) = \mathcal{H}(G(\theta)) \tag{3.14}$$

where $\mathcal{H}(p)$ is the entropy of a Bernoulli variable with probability of success p. In other words, $H(S\left(\theta\right))$ represents the uncertainty of the event "the termination threshold is reached" for a given value of θ. The use of an indicator Ψ should allow us to reduce such uncertainty. If Ψ can assume m distinct values, $\psi_1, \ldots, \psi_m$, one can compute the *conditional entropy* of success assuming Ψ, i.e.,:

$$H(S\left(\theta\right)|\Psi) = \sum_{i=1}^{m} \Pr(\psi_i) H(S\left(\theta\right)|\psi_i) = \sum_{i=1}^{m} \Pr(\psi_i)\mathcal{H}(G\left(\theta|\psi_i\right)) \tag{3.15}$$

$H(S\left(\theta\right)|\Psi)$ quantifies the amount of uncertainty of the event "the termination threshold is reached" for a given value of θ if we know, at each trial, the value of the Ψ indicator (it has to be noted that $H(S\left(\theta\right)|\Psi) \leq H(S\left(\theta\right))$). The *mutual information* $I(S\left(\theta\right), \Psi)$ represents the information about the termination event contained in the Ψ indicator, and is defined as:

$$I(S\left(\theta\right), \Psi) = H(S\left(\theta\right))H(S\left(\theta\right)|\Psi) \tag{3.16}$$

The higher the value of $I(S\left(\theta\right), \Psi)$ for a given indicator, the better the estimation of the success probability given by the Ψ indicator, and the sooner we are expected to reach the termination threshold for a query if we schedule regions according to values of $G_i\left(\theta|\psi_i\right)$.

This suggests that the indicator that maximizes $I(S(\theta), \Psi)$ for a given data set is the one that provides the optimal schedule.

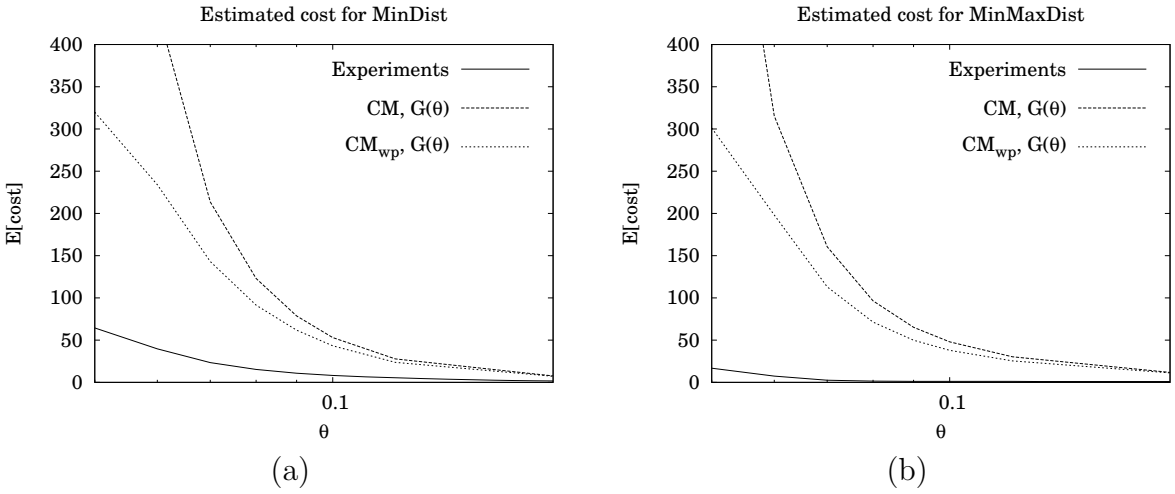
3.6 Experimental Evaluation

In this Section we present the experimental evaluation of the techniques proposed in the previous sections. Results included are all obtained from the following *real* data sets:

`Corel:` This data set contains 68,040 32-dimensional color histograms extracted from a collection of `Corel` images. The value for each dimension in the histogram represents the density of the relative color in the entire image. We downloaded the data set from the UCI Knowledge Discovery Site (`http://kdd.ics.uci.edu/`).

`Airphoto:` This data set consists of 274,424 60-dimensional vectors. Each vector contains texture information extracted from a tile of size 64×64 that is part of a large aerial photograph (there are 40 air-photos in the data set). Each tile is analyzed by means of 30 Gabor filters, and for each filter the mean and the standard deviation of the output are stored in the feature vector. This data set was given to us by B.S. Manjunath.

`EEG:` This data set contains 2,824,483 64-dimensional vectors, where each component represents a measurement obtained from an EEG electrode placed on a subject's scalp. Again, the data set was downloaded from the UCI Knowledge Discovery Site.



(a) (b)

Figure 3.2: Estimated cost based on $G_i(\theta)$ for (a) MinDist and (b) MinMaxDist

3.6.1 Cost Models

In the following figures we show the prediction capabilities of the cost model presented in Section 3.2.1. We denote with $\mathcal{CM}$ the cost model which uses the stopping probability defined by Equation 3.2, while with $\mathcal{CM}_{wp}$ the one defined in Equation 3.4, which is conditioned to the value of the MinDist of the selected region.

In Section 3.3.2 we have introduced four different classes of distance distributions with respect to a random query $\mathbf{q}$. In Figure 3.2 we show:

- The comparison between the original cost model $\mathcal{CM}$ and the cost model $\mathcal{CM}_{wp}$;
- The comparison between the experiments and the predictions based on a query-independent simple distance distribution.

It is worth noting that $\mathcal{CM}$ overestimates the experimental costs, both in the case of the MinDist and the MinMaxDist indicators (Figures 3.2(a) and 3.2(b), respectively). Even with a cost model which performs the pruning of the search space, the predictions are not accurate. The problem is that the cost model is not able to differentiate among different queries, thus showing poor performances, as it is certified by this figures.

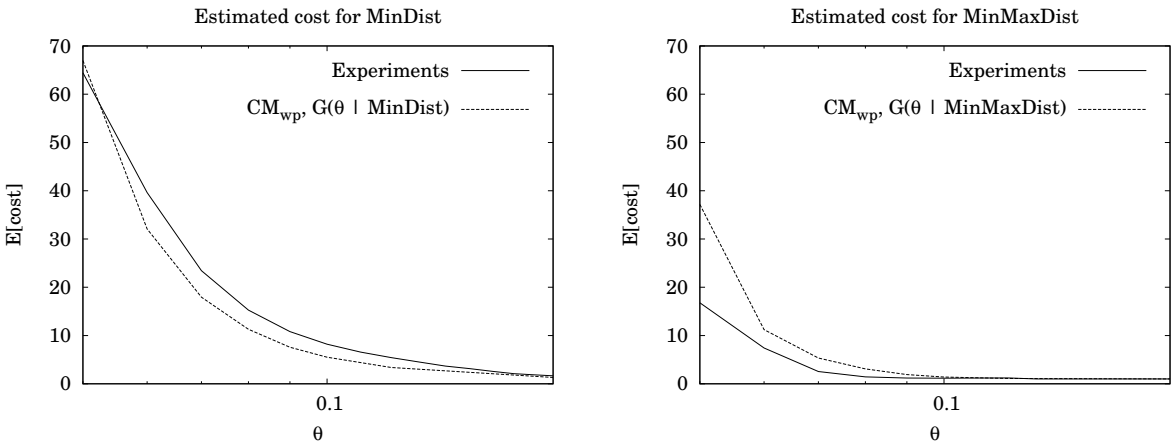
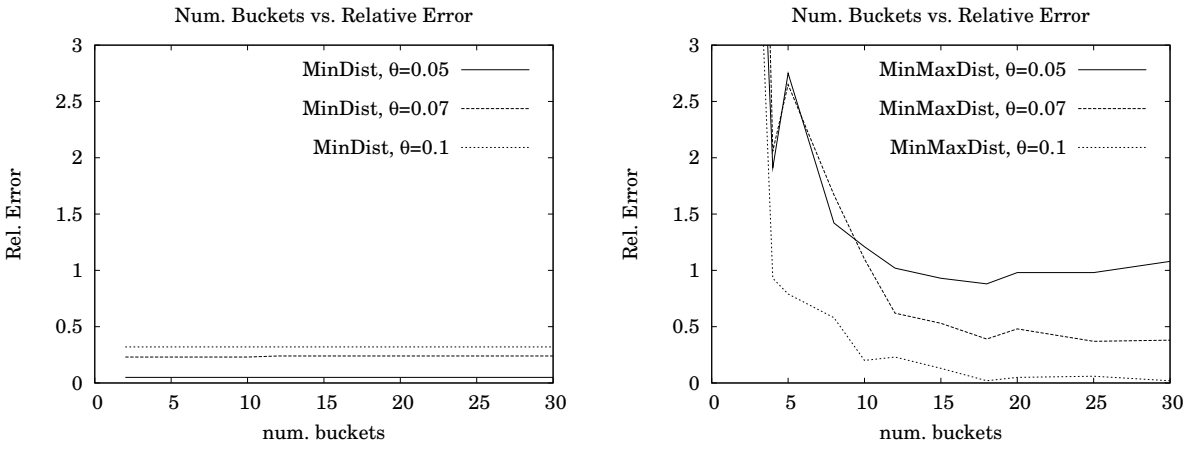


Figure 3.3: Estimated cost based on $G_i(\theta|\psi_i)$ for (a) MinDist and (b) MinMaxDist

In Figures 3.3(a) and 3.3(b) a query-dependent simple distance distribution is used by the model and the predictions are more accurate that in the previous case.

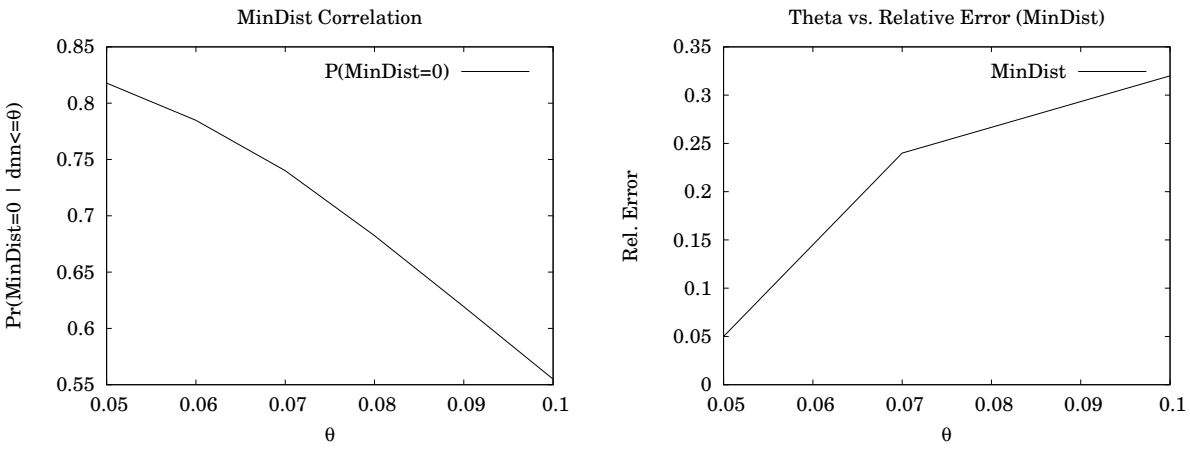
In Figure 3.4, it is shown the relative error on the predictions $\frac{|cost_{est}-cost_{real}|}{cost_{real}}$, i.e., the difference between the estimated cost $cost_{est}$ and the real cost $cost_{real}$ divided by the real cost, for the schedules MinDist and MinMaxDist. In particular, it is shown how the error depends on the number of buckets in which $G_i(\theta|\psi_i)$ is partitioned for a fixed



(a) (b)

Figure 3.4: Relative error in the prediction of the (a) MinDist and (b) MinMaxDist schedule for various values of θ.

θ. Notice that there exists an optimal number of buckets which, depending on the θ considered, ranges between 10 and 20. In the rest of the section we have chosen the value 10, because it is a good compromise between the memory consumption and the accuracy of the model.



(a) (b)

Figure 3.5: (a) Probability $P(\text{MinDist}(q, \mathcal{D}_i) = 0 | d(q, \mathcal{D}_i) \leq \theta)$ and (b) relative error in the prediction of MinDist as a function of θ.

If the schedule considered is MinDist, the relative error is approximately constant for a given value of θ and the dependency is shown in Figure 3.5(b). At a first glance, this is a "strange" behavior, which however, can be explained by looking at Figure 3.5(a), i.e., to the probability $P(\text{MinDist}(q, \mathcal{D}_i) = 0 | d(q, \mathcal{D}_i) \leq \theta)$. There is a strong correlation

between the event that a random region in the data set contains an object o^* s.t. $d(q, o^*) \leq \theta$ and the event that, for that specific region, $\text{MinDist}(q, \mathcal{D}_i) = 0$. Due to the ordering criterion, if the correlation is high enough, only regions whose MinDist is 0 are explored, being the resulting prediction independent from the number of buckets of the histogram. As a final remark, note that the "quality" of the prediction degrades when θ increases: this is due to the decrease of the first bucket's precision, which brings the relative error back to "normal" levels (if compared to the values shown in Figure 3.4).

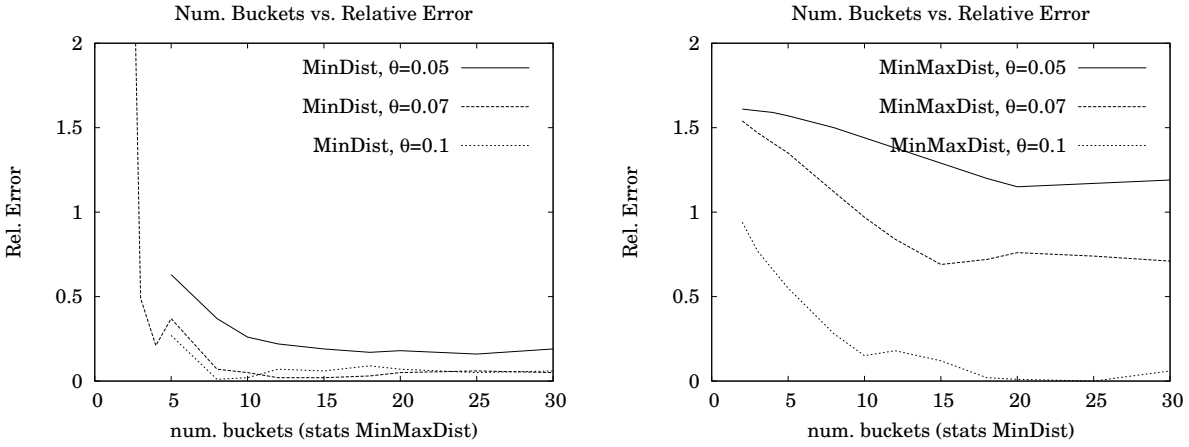


Figure 3.6: Relative error for the prediction of (a) the schedule MinDist based on the statistics $G_i(\theta|\text{MinMaxDist})$ and (b) the schedule MinMaxDist based on the statistics $G_i(\theta|\text{MinDist})$.

In Figures 3.6(a) and 3.6(b) we show, for indicators MinDist and MinMaxDist respectively, the relative error obtained if a "wrong" set of statistics were used. The predictions are still good, which confirms the robustness of the model and this can be verified by comparing the relative error of Figures 3.4 and 3.6(b).

In general, if we refer to histograms that are accurate enough (i.e., which have a sufficient number of buckets), the prediction is more accurate if the ordering criterion and the statistics are homogeneous: this is true if there exists a positive correlation between the ordering of the region and the probability of finding an element at a distance lower than θ, which is what actually happens in practice. On the other hand, however, if we are looking for a more efficient solution in terms of memory consumption at the price of a little degradation in the model performance, it is interesting to consider using statistics on MinDist only, because, due to the correlation described by Figure 3.5.(a), it performs better with fewer number of buckets (see Figure 3.7).

Now consider a schedule that orders the data region according to a criterion that is

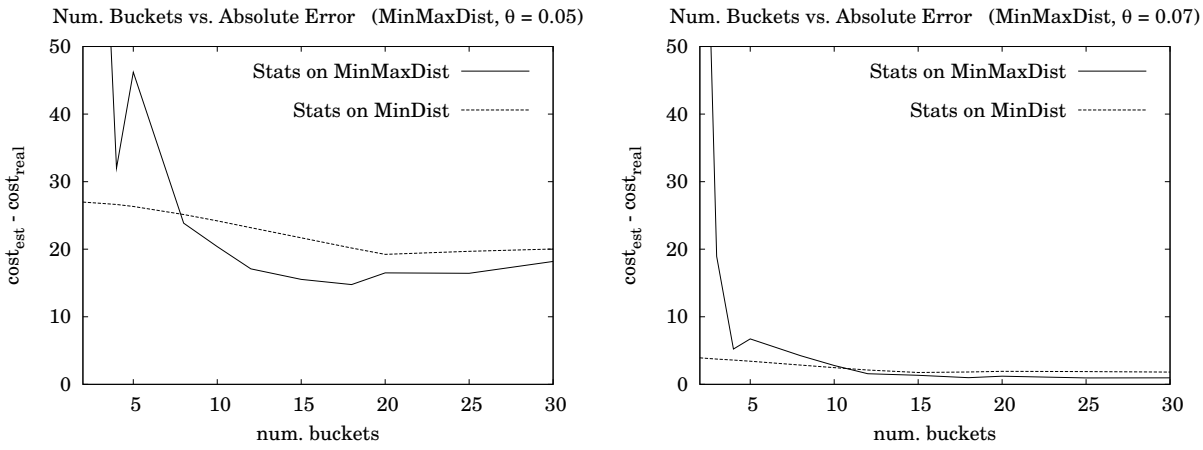


Figure 3.7: Absolute error in the prediction of MinMaxDist with statistics on (a) MinMaxDist and on (b) MinDist.

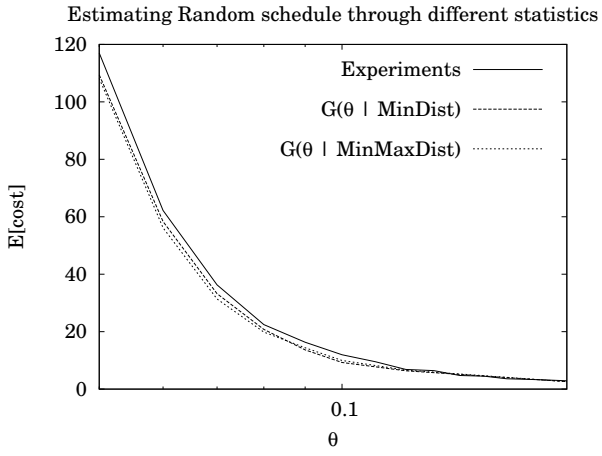


Figure 3.8: Random schedule and expected cost based on $G_i(\theta|\text{MinDist})$ and $G_i(\theta|\text{MinMaxDist})$.

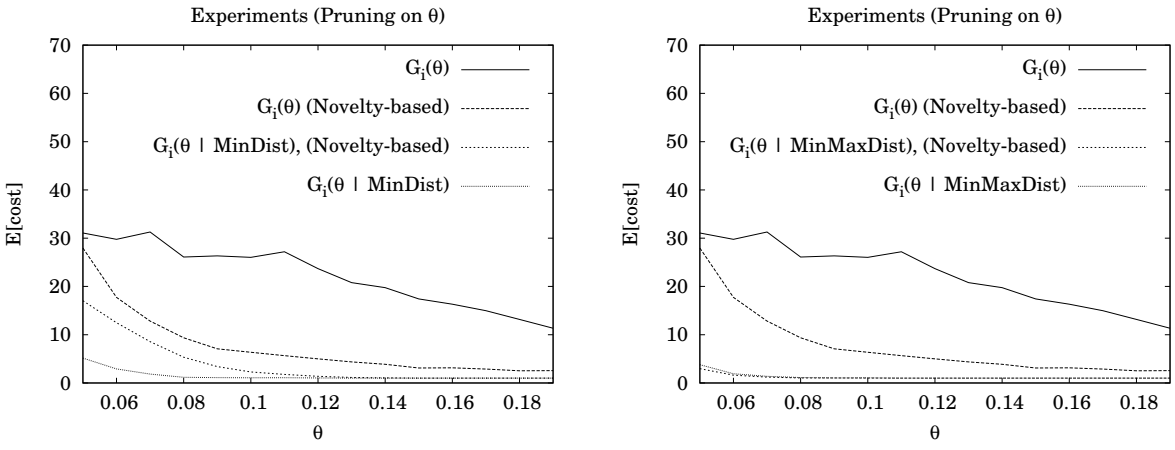


Figure 3.9: Average cost of a query for the optimal-on-the-average schedules using (*a*) MinDist and (*b*) MinMaxDist statistics.

independent from the indicator on which the statistics are built. We expect that the model $\mathcal{CM}_{wp}$ predicts approximately the same cost with any set of statistics, given that they are accurate enough. In Figure 3.8 we show the cost of a random schedule and the expected cost, computed by using statistics on MinDist and on MinMaxDist.

3.6.2 Optimal Schedules

The cost model presented in the previous sections is a powerful tool to estimate the expected cost of a workload of queries, however it can also be used to derive an optimal-on-the-average schedule, as explained in section 3.4. In particular, it suggests that the optimal ordering criterion is the one which chooses, at each step i, a region $\mathcal{D}_i$ such that the probability $\Pr\{\exists o \in \mathcal{D}_i\}$ is maximized.

Figure 3.2 shows us that a query-independent simple distance distribution $G_i(\theta)$ cannot estimate the expected cost of a schedule. The same distance distribution, as we would have expected, cannot be used as the ordering criterion as well: it does not distinguish among different queries, therefore it generates a static ordering of the regions of the space which, adopted by any query executed in the data set, leads to poor performances.

In Figure 3.9(*a*) and 3.9(*b*), we have shown the average cost of a query when the ordering criterion is based on the probability of finding the nearest neighbor in the chosen region. Each schedule corresponds to a different implementation of $\Pr\{d_i(q) \leq \theta\}$ presented in Section 3.4. In Figure 3.9(*a*) the query-dependent schedules are conditioned to the values of MinDist and in Figure 3.9(*b*) they are conditioned to the values of Min-

MaxDist. Also notice that the query independent schedules are reported in both figures for ease of completeness, though they are identical.

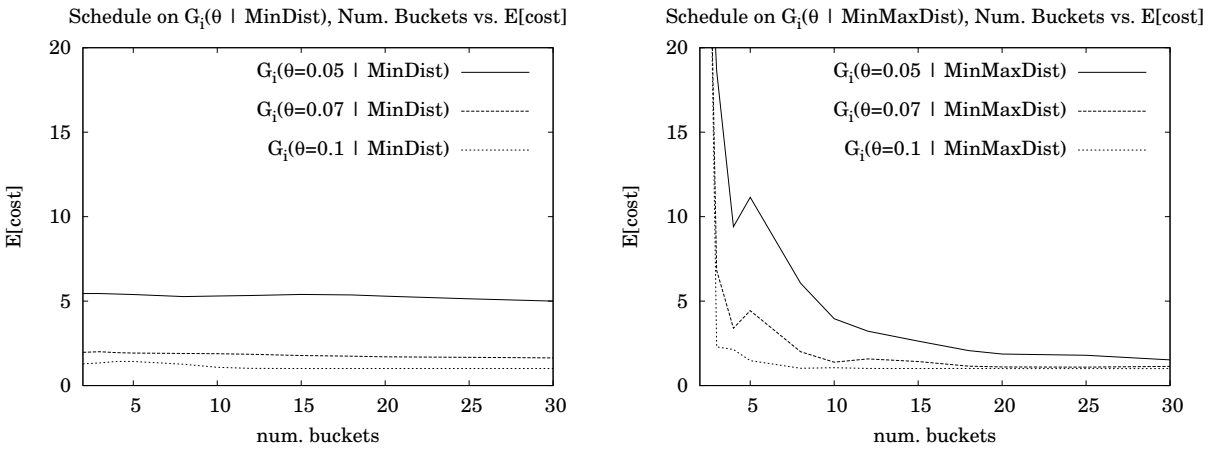


Figure 3.10: Average cost of a query for different values of θ using a schedule based on (a) $G_i\left(\theta|\text{MinDist}\right)$ and (b) $G_i\left(\theta|\text{MinMaxDist}\right)$.

The more accurate is the estimate of the $\Pr\left\{d_i\left(q\right) \leq \theta\right\}$ probability, the lower is the average cost of the query. The novelty-based query-dependent schedule is too complex to implement and, in the case of the MinMaxDist indicator doe not have any relevant improvement on the simple query-dependent schedule. Therefore, in Figure 3.10 we only consider a simple query-dependent schedule and compare the average cost of a query and how it depends on the number of buckets in the histograms of $G_i\left(\theta|\text{MinDist}\right)$ and $G_i\left(\theta|\text{MinMaxDist}\right)$ (Figures 3.10(a) and 3.10(b), respectively).

3.6.3 Entropy

In Figure 3.9 we observe different performances between the schedules based on MinDist and those based on MinMaxDist. This effect is even more visible if we allow a large number of buckets in the histogram of $G_i\left(\theta|\text{MinMaxDist}\right)$, making the MinMaxDist indicator heavily outperforming MinDist.

The ultimate reason for these differences is to be searched in the *information* that each indicator brings as a representative of the region. We have experimentally seen that the task of estimating the probability $\Pr\left\{d_i\left(\mathbf{q}\right) \leq \theta\right\}$ for each region $\mathcal{D}_i$ through the distribution $G_i\left(\theta\right)$ leads to poor results: from an information-theoretic point of view, conditioning that distribution to the event that *a query has seen a certain indicator* means reducing the uncertainty of the system, thus increasing its information. In Figure 3.11.(a)

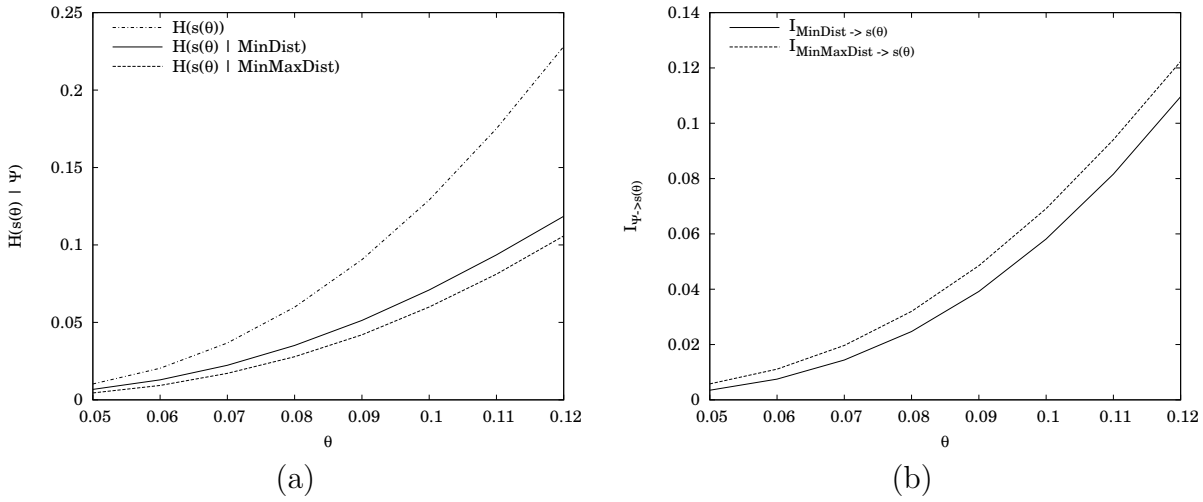


Figure 3.11: (a) Entropy and (b) mutual information for different indicators.

we show the entropy associated to various indicators and in Figure 3.11.(b) the additional information that they introduce into the system.

It is interesting to compare Figures 3.9 and 3.11: the indicator MinMaxDist, which has a better performance, is also the one to which it is associated the lower entropy. As we have already introduced in Section 3.5.2, we exploit the knowledge of the conditional entropy of these indicators in order to choose among the available indicators the one which has a higher correlation with the event that "the region contains the nearest neighbor with respect to the query". Notice that this information, in general, depends on the considered data set.

3.6.4 Scheduling on Indicators Values

There are cases in which it is not possible to rely on a set of statistics for each data region, for example because it is not possible to rebuild the whole data structure in order to collect such statistics, or because the data cannot be directly accessed. In these situations, we can exploit our results on the relation between indicator values and their ability to predict the content of data regions.

In Figure 3.12 we show the average cost of a query for different indicators. Notice that, as we would have expected, the MinMaxDist outperforms the MinDist indicator and the MaxDistindicator is slightly better than the MinMaxDist.

As a final remark, notice that the concept of optimality for a schedule, now, is more vague, because we should also consider the entropy associated to an indicator.

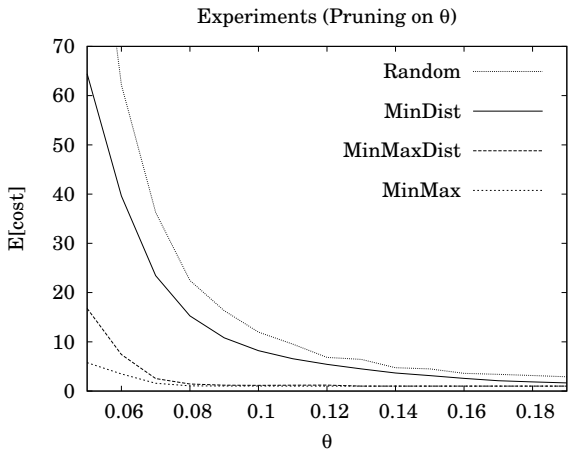


Figure 3.12: Average cost of the query for various indicators.

3.7 Hierarchical networks

As data sets grow exponentially in size, the number of data regions in which they are partitioned grows, eventually becoming a *data set of regions* which is smaller in size, but still too big to be managed efficiently by the system. The split rule presented in Section 2.3.1, then, can be reiterated, in order to create certain number of *levels*, each of them associated to a different division of the data set.

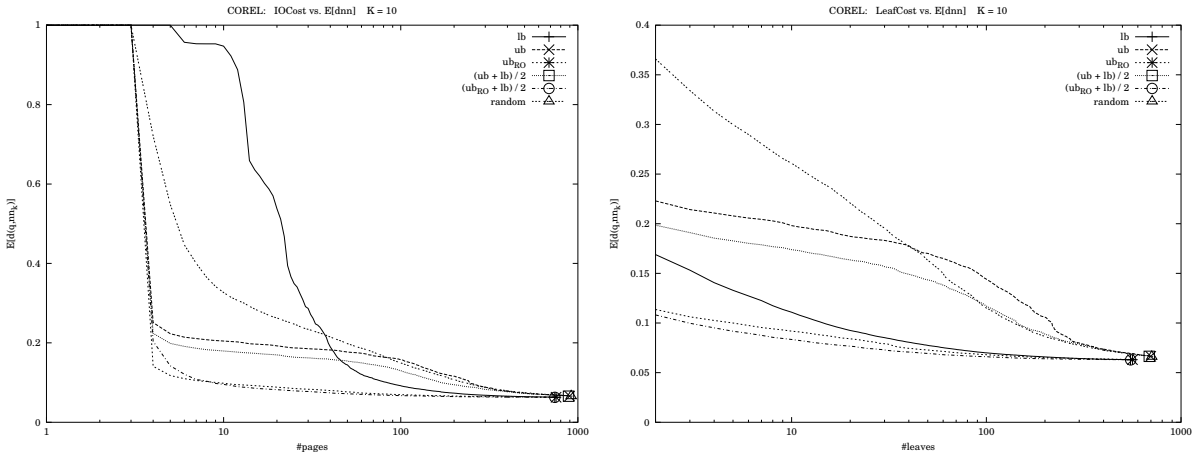
A hierarchical network consists of a set of regions organized on two or more levels with the characteristic that the regions residing at the lower levels of the structure are fully contained in the regions at the higher level. In particular, each region $\mathcal{D}_i$ at level l has all its child regions $\mathcal{D}_1, \mathcal{D}_2, \ldots$ at level $l+1$ and $\mathcal{D}_i \supseteq \bigcup_j \mathcal{D}_j$. Query processing typically begins at the root region and descends the hierarchy, exploring the most promising sub-hierarchies first.

3.7.1 Experimental Evaluation

In this section, we show that if a hierarchy has more than two levels the problem of deriving analytically an optimal schedule becomes increasingly difficult, because, at each step, only a partial knowledge on the network structure and on the probabilistic information associated to the regions is available. We performed an extensive evaluation with the M-tree index structure [CPZ97] on the three considered data sets and validated experimentally the generality of our hypotheses. The overall conclusion is that most of the conclusions drawn for 2-levels hierarchies are still valid for general hierarchies. However, for some indicators, performance highly degrades for taller hierarchies, since they mislead

to choose to explore the wrong parts of the hierarchy at upper levels.

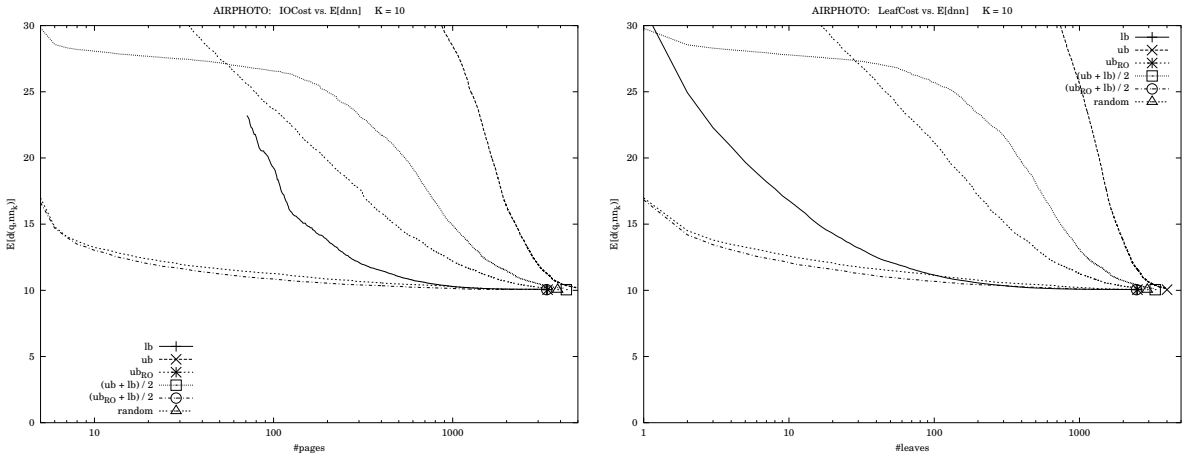
In this section we show the experimental results of a cost-optimal control policy for a variety of indicators. As an example, in Figures 3.13, 3.14, and 3.15 we plot the distance of the 10-th NN to the query as a function of the number of accessed regions and of accessed leaf regions, respectively (results for other values of k were similar and have been omitted here for brevity). Notice that lb refers to the MinDist indicator, ub to the MinMaxDist and ub_{RO} to the MaxDist. *random* represents a random schedule and $\frac{lb+ub}{2}$ and $\frac{lb+ub_{RO}}{2}$ are two indicators whose meaning is evident.



(a) (b)

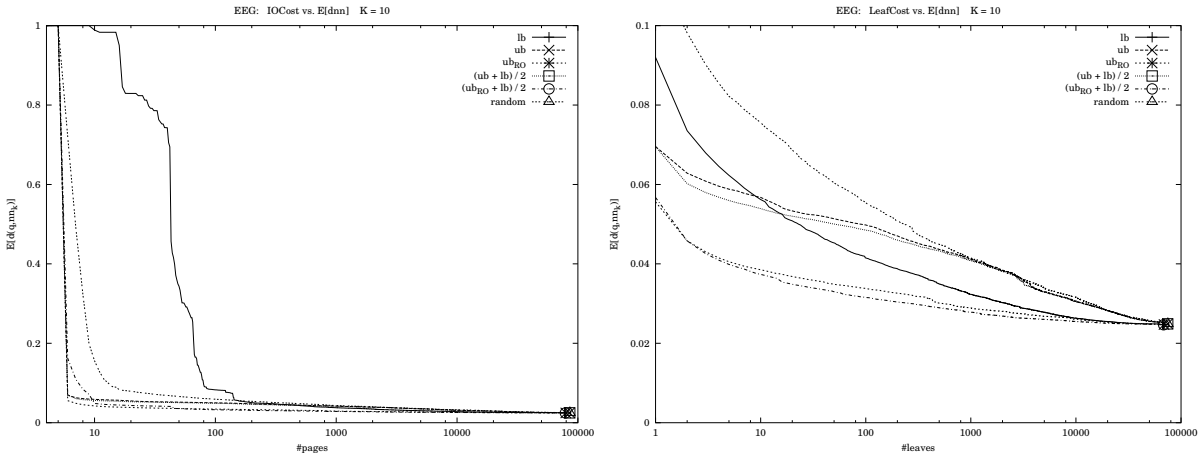
Figure 3.13: Result quality ($k = 10$) for different indicators on the `Corel` data set as a function of the number of fetched regions (a) and accessed leaves (b).

As for the $k > 1$ case, we performed an extensive evaluation with the M-tree index structure [CPZ97] on the three considered data sets and validated experimentally the generality of our hypotheses. The



(a) (b)

Figure 3.14: Result quality ($k = 10$) for different indicators on the `Airphoto` data set as a function of the number of fetched regions (a) and accessed leaves (b).



(a) (b)

Figure 3.15: Result quality ($k = 10$) for different indicators on the `EEG` data set as a function of the number of fetched regions (a) and accessed leaves (b).

Chapter 4

Partially Connected Networks

In Chapter 2 we have introduced the idea of partitioning a data set into different regions in order to (*i*) optimize the efficiency of query processing or (*ii*) model, from a logical point of view, a situation already existing in practice, in the form of a data set already partitioned in physically different regions, such as a distributed system. In this chapter we deal with those network organizations in which each region is directly linked to a finite number of other regions and does not have any knowledge about the others, such that a query that is processed at region, say, $\mathcal{D}_i$ has no means to guess the global topology of the network other than $\mathcal{D}_i$'s direct neighbors. Despite the evident gap that intercurs between an ad-hoc data structure (such as well-tuned hierarchy) and a flat network in terms of efficiency of the query processing, there is a more notable difference that concerns the *quality* of the results that are found by a query in the two scenarios. In general, it is not guaranteed that all regions can be reached by any other region in the network by simply navigating the links among two neighbors. Consider, for example, a situation in which there exists a small cluster of regions which have *ingoing* links but not *outgoing* ones. If a query starts from any one region belonging to that group, there is no way it can find its way out, therefore either it finds the optimal solution in one of those regions or the global solution cannot be found at all.

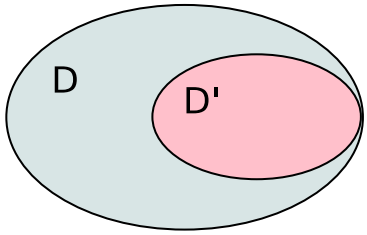


Figure 4.1: The reachability problem in flat topologies

In Figure 4.1, we show the *reachability problem* from an abstract point of view: for

each region $\mathcal{D}_i$, there exists a subset $\mathcal{D}' \subseteq \mathcal{D}$ of the original data set $\mathcal{D}$, such that the solution for a query q can only be found in $\mathcal{D}'$. Therefore, the definition of *exact solution* for a query q strongly depends on the region where the query is executed the first time.

Maintaining a flat topology does not require any central coordination: however, if we compare a flat topology to a hierarchical system which is built on the same identical data, the latter is more efficient and does not incur in the reachability problem. In terms of efficiency and efficacy, then, there is no apparent need to prefer a flat topology to a hierarchical one when dealing with a centralized scenario.

In the following we will show how to organize data regions in a flat topology and how to query them. In Section 4.1 we investigate the problem of building and maintaining a *Metric Overlay Network* (MON), i.e., a particular type of peer-to-peer (P2P) network which is embedded in a metric space, in a scenario typical of information retrieval [BYRN99]. In such a scenario, a critical issue is constituted by the overhead generated by the network maintenance algorithms and by the data they exchange. A variety of query processing techniques are presented in Section 4.4.2, where we show experimental results based on the same data sets that had been used in Section 3.6. .

4.1 Introduction to P2P networks

Peer-to-peer networks potentially offer major advantages for large-scale decentralized search of information such as file sharing or Web search [SW05]. We envision an architecture where every peer has a powerful local search engine, with its own crawler, indexer, and query processor. Such a peer can compile its own content from thematically focused crawls and other sources, and make this content available in a P2P overlay network. Search requests issued by a peer can first be executed locally, on the peer's locally indexed content, with a strong potential for personalization and other advanced IR techniques. In cases when the recall of the local search result is unsatisfactory, the query may be forwarded to a small set of other peers that are expected to provide thematically relevant, high-quality and previously unseen results. Deciding on this set of target peers is the *query routing* problem in a P2P search network (aka. collection selection).

A practically viable query routing strategy needs to consider the similarity of peers in terms of their thematic profiles, the overlap of their contents, the potential relevance of a target peer for the given query, and also the costs of network communication. Many proposals have been made in the literature, for example: globally available term statistics about the peers' contents [CLC95, GGMT99, MYL02, BNST05], epidemic routing using gossiping strategies [KNOT06], routing indices with peer summaries from local neigh-

borhoods [CGM04, LTQ+05], statistical synopses such as Bloom filters or hash sketches maintained in a directory based on distributed hash tables (DHT) [BMT+05, MBTW06, PRL+06], randomized expander graphs with low-diameter guarantees [MS05, MS06] and randomized rendezvous [PMW05], clustering of thematically related peers [DNV06a, DNV06b, LC06], superpeer-based hierarchical networks [LC03, LC05b], cost/benefit optimization based on coarse-grained global knowledge [NF04, NF06], and many more.

Typically, query routing decisions would be made at query run-time, when the query is issued at some peer. But many of the above methods involve directory lookups, statistical computations, and multi-hop messages; so it is desirable to precompute some basic elements of query routing decisions and amortize this information over many queries. A technique for doing this is to encode a similarity-based precomputed binary relation among peers into a so-called *semantic overlay network* [CGM04, ACM05]. Each peer P becomes directly connected to a small number of peers that are likely to be good routing targets for many of P's queries. At query run-time, the query router would consider only the semantic overlay network neighbors of the query initiator and select a subset of these based on a more detailed analysis of similarity, overlap, networking costs, etc.

From a given peer's viewpoint, one particularly intriguing choice of network neighbors are those peers with the lowest *Kullback-Leibler (KL) divergence* [CT91] regarding the term-frequency distributions in the peers' locally indexed contents. This information-theoretic measure captures the general thematic similarity of two peers in a natural manner and is well-founded in the recent work on statistical language models for IR [LC05a]. In a language model (LM), queries and documents are viewed as samples generated from an underlying probability distribution over terms. When we assume that the contents of a peer represents this distribution and assume that queries are generated according to a multinomial model, the best peer for a query is the one that has the highest likelihood of generating the query; this is mathematically equivalent, in terms of peer ranking, to the lowest KL divergence between the query LM and the peer LM. Finally, if we do not have a specific query at hand at the time we want to precompute the similarity of two peers, we consider the contents of the query-initiating peer as a substitute for the query itself and replace the query LM by that peer's contents LM.

LM's have become state-of-the-art practice for many IR tasks, and they have already been proposed also for P2P IR by [LC03]. In conventional IR settings, with one LM for each document and one LM for each query, parameter estimation and smoothing are very critical parts of LM-based approaches; details of smoothing techniques are often decisive for search result quality [ZL04]. In contrast, a P2P setting provides a much more convenient and robust environment for LM's, as the peers have rich and naturally

available sources for parameter estimation and smoothing. Instead of a document LM we can use the entire contents of a peer as the basis of the LM or as a background corpus for smoothing, and instead of a query LM we can leverage the user-behavior history (query logs, click streams, bookmarks) for robust parameter estimation. In this sense, LM's and P2P networks are a "marriage made in heaven".

Unfortunately, however, an LM-based approach to P2P IR also entails major computational costs, and it is unclear how to make this approach practically viable in a large-scale environment.[1] More specifically, we face the following efficiency problems:

1. Computing the exact KL divergence between two peers' term-frequency distributions incurs non-negligible overhead as it ranges over high-dimensional feature spaces. It would be desirable to disregard a large part of "insignificant" features, but it is not obvious how to efficiently approximate the KL divergence this way and control the approximation error.

2. The computations involve shipping term-frequency vectors over the network. With high-dimensional feature spaces, these messages have non-negligible size and may incur significant consumption of network bandwidth.

3. Most severely, as the KL divergence is not a metric (i.e., the triangle inequality does not hold), there is no obvious way of transitively inferring, from knowing the distances for some pairs of peers, transitive distances so as to eliminate peers that are "too far away" from a given peer. Thus, we would need to compute the KL divergence for *all pairs of peers*, and this quadratic cost is out of the question for a scalable approach. Being able to prune the search space of all peer pairs is fundamentally important and poses a great challenge.

4.1.1 Our Solution

We address the above problems by combining insights and methods from different areas and integrating them into a novel kind of network which we call *Metric Overlay Network*, because its ultimate objective is to embed both peers and data objects in a metric space, in order to facilitate network maintenance and query routing. First, we utilize recent results from information theory [ES03], which show that the square root of the *Jensen-Shannon (JS) divergence* is a metric. The JS divergence is a symmetrized and smoothed relative of the KL divergence. We adopt the JS divergence instead of the KL divergence as a metric

[1] This is probably the reason why prior work has considered LM-based approaches only for superpeer-based P2P systems, but these approaches do not scale up to completely decentralized networks with millions of equally standing peers and no hierarchical structure at all.

distance measure because it allows us to effectively prune the search space of peer pairs. Second, we build on existing methods for metric similarity search in centralized settings and adapt them for our P2P setting. Peers meet randomly and exchange information about their nearest neighbors in the metric space. Peers remember the distances to interesting peers, and with the JS-divergence-based metric, they can effectively use the triangle inequality for early elimination of uninteresting peers. This technique is key for scalability. Third, we compress the term-frequency vectors and speed up the computation of two peers' JS divergence by using appropriately designed compact synopses based on Bloom filters [Blo70]. Our technique comes with guarantees about the approximation error.

1. Fast comparisons of per-peer LM's by synopsis-based approximations with error bounds,
2. Low communication cost by LM compression and search-space pruning using the JS-based metric,
3. Judicious message routing for MON construction and maintenance.

4.2 Network Architecture

Our system consists of a physical layer (an example is the TCP/IP infrastructure) connecting a variable number of autonomous peers, which cooperate in order to build and maintain a virtual network of regions that are directly linked only if their local data collections are "semantically similar". Ideally, this similarity should be (reciprocal to) a *metric distance*, computed by a function $d : \mathbb{X} \times \mathbb{X} \rightarrow \mathbb{R}_0^+$ which satisfies the triangular inequality and where $\mathbb{X}$ is a space containing all objects and peers in the system. Each peer P_i is responsible for a region of the data set $\mathcal{D}_i$ and is represented by the ball region $\mathcal{B}_{c_i}(r_i)$, centered in c_i and with radius r_i. Notice that the problem of choosing c_i such that r_i is minimized is hard and out of the scope of this dissertation.

In general, we are interesting in viewing the P2P network as a *metric space* and construct the MON from the metric distances between peers. The advantage of such a model would be that we can exploit the triangle inequality to prune the search space of all peer-pairs when we construct the edges of the MON. Once we know that for peers x, y, z the distances are upper-bounded by $d(x, y) \leq a$ and $d(y, z) \leq b$, we can infer that $d(x, z) \leq a + b$. With sufficiently small values a and b we could then consider peer z as a candidate to be linked to x in the MON. Conversely, if we do not have any indication of this kind and rather suspect that z is semantically far away from x based on other

observed distances, we could prune z from our search for good neighbors of peer x even without computing $d(x, z)$.

With the above consideration, an intriguing strategy is to run a process of random (but possibly biased) meetings of peer-pairs. When two peers "meet", we compute or approximate their semantic distance and keep this pair as a candidate for a MON edge if we estimate that the distance is small enough. We iterate these meetings and use the triangle inequality and other techniques for distance estimation. In the end, the peer-pairs with the lowest (estimated) distances form the edges of the MON. As peers may leave the P2P network anytime, new peers join, and the document contents of a peer evolves over time, we actually run peer meetings continuously in the background in order to maintain the MON in the presence of this dynamics. We present more details of this MON construction later in Section 4.3.

In a MON peers search for their nearest neighbors with respect to the function d. When this procedure is given a sufficiently large time, it eventually converges to a stable situation in which each peer has found its exact nearest neighbors (this is true only in a very slow-changing network). This kind of links is usually referred to as *short-distance links*, because only the k_s closest regions are kept as neighbors, while the others are simply left behind. At the beginning of this Chapter we have introduced the reachability problem, as a serious limitation to the accuracy of any query processing technique in a P2P network. To overcome to such difficulty, each peer is provided with k_l *long-distance links*, so called because they store a set of neighbors chosen at random among those that the peer has previously met during its lifetime. The reason for this second type of links is two-fold: they avoid the network to become disconnected if a highly clustered group of regions start establishing links one with each other and can be used to jump from one region to the other of the space, thus reducing the impact of reachability problem (notice, however, that it cannot be completely avoided).

4.2.1 Metric Distances and Language Model

In our scenario we consider each data object to be described by a set of *keywords*, such as a text document which is characterized by its terms, an image, which is often associated to a set of tags, or even an Mp3 file. We further associate a *weight* to each keyword so as to represent the importance of the keyword as a description for the given object. A query is also a set of keywords, for which the weights are explicitly assigned by the user or they are given by the system (e.g., through relevance feedback techniques [WFSP00]). The *geometrical descriptor* of an object o_i, then, consists of the set of tuples $\{\langle k_1, v_{1,i}\rangle, \ldots, \langle k_n, v_{n,i}\rangle\}$, where each k_j is a keyword and $v_{j,i}$ is the weight of k_j in o_i. Analogously, a *peer descriptor*

is obtained by the descriptors of the objects $o_1, \ldots, o_m$ stored at this peer as follows. For each keyword $k_j \in o_1 \cup o_2 \cup \ldots \cup o_m$, the average weight v_j is given by $v_j = \frac{\sum_{i=1}^{m} v_{j,i}}{m}$ and the resulting descriptor is the set $\{\langle k_1, v_1\rangle, \langle k_2, v_2\rangle, \ldots\}$. Notice that $v_{j,i} = 0$ if keyword k_j is not present in the descriptor o_i.

Such a descriptor is usually referred in Information Retrieval as the *Language Model* of the object o_i [BYRN99]. In general, a peer P_Θ is geometrically characterized by a Language Model Θ and a radius r_Θ, such that all objects o_i stored at P_Θ have a distance (that will be defined soon in this section) from Θ lower than r_Θ. In other words, $\mathcal{B}_\Theta(r_\Theta)$ is the ball region associated to P_Θ.

From now on, we will refer to a peer through its LM and will use gGreekletters, such as Θ and Ψ, to denote the LM's associated to peers P_Θ and P_Ψ.

A typical "distance" which measures the dissimilarity between any two objects o_Θ and o_Ψ is the Kullback-Leibler divergence [CT91] between their respective Language Models Θ and Ψ:

$$D_{KL}(\Theta \parallel \Psi) = \sum_{i=1}^{|\mathcal{T}|} \theta_i log \frac{\theta_i}{\psi_i} \tag{4.1}$$

where θ_i and ψ_i are the weights associated to term t_i in the two Language Models. Unfortunately, D_{KL} is not symmetric and does not satisfy the triangular inequality. A symmetrized version is the Jensen-Shannon divergence [CT91],

$$D_{JS}(\Theta, \Psi) = \frac{1}{2} \cdot [D_{KL}(\Theta \parallel \Phi) + D_{KL}(\Psi \parallel \Phi)] \tag{4.2}$$

where $\Phi = \frac{1}{2}(\Theta + \Psi)$. Adopting an information-theoretic approach, if we have a random variable A whose observations can be drawn with equal probability from any of two distributions Θ and Ψ, then the minimum codelength needed to represent A can be found computing the "inefficiency distance" of both distributions from Φ and by averaging over the result. Recent advances in the field of information theory [ES03] have shown that the measure $\sqrt{D_{JS}}$ is a metric. Thus, the metric distance function $d : \Theta \times \Psi \rightarrow \mathbb{R}_0^+$ between any two Language Models Θ and Ψ is defined as:

$$d(\Theta, \Psi) = \sqrt{\sum_{t_i \in \mathcal{T}_\Theta \cup \mathcal{T}_\Psi} d_{JS}(\theta_i, \psi_i)} \tag{4.3}$$

and

$$d_{JS}(\theta_i, \psi_i) = \frac{1}{2}\left[\theta_i \cdot \log \frac{2 \cdot \theta_i}{\theta_i + \psi_i} + \psi_i \cdot \log \frac{2 \cdot \psi_i}{\theta_i + \psi_i}\right]. \tag{4.4}$$

In Table 4.2.1 we summarize the notation that will be used in the rest of the section. In particular, Θ and Ψ denote the Language Models of peers P_Θ and P_Ψ; $\widetilde{\Theta}$ and $\widetilde{\Psi}$ are their approximate versions (introduced in Appendix A) and r_Θ and r_Ψ denote the search radii for the two peers (used by the meeting algorithms in Section 4.3).

Table 4.1: Summary of relevant notation

Symbol	Description
$\mathcal{T}$	Global term space
$\mathcal{T}_\Theta \subset \mathcal{T}$	Term space of peer P_Θ
Θ, Ψ	LM's associated with peers P_Θ, P_Ψ
$\widetilde{\Theta}$, $\widetilde{\Psi}$	Approximate LM's of peers P_Θ, P_Ψ
θ_i, ψ_i	Weight of the i-th term in P_Θ, P_Ψ
$\widetilde{\theta}_i$, $\widetilde{\psi}_i$	Approx. Weight of the i-th term
$\theta_H^{(i)}$	Weight associated with the l-th BF
r_Θ, r_Ψ	Search radii for peers P_Θ, P_Ψ
$d = d(\Theta, \Psi)$	Distance between P_Θ and P_Ψ
$\widetilde{d} = d(\widetilde{\Theta}, \widetilde{\Psi})$	Approx. distance between P_Θ and P_Ψ

4.3 Building and Maintaining a MON

When two peers meet they first exchange their LM's and then compute their similarity by applying Equation 4.3: unfortunately, this operation leads to high communication costs because it requires a huge set of term weights to be sent across the network.

In general, the problem of network overhead becomes critical whenever the peer descriptors that are exchanged during a meeting are too big. Our solution requires the peer descriptors and the distance function to follow these rules:

1. A peer descriptor $\mathcal{D}_i$ is split in I *segments* $\mathcal{D}_i^{(1)}, \ldots, \mathcal{D}_i^{(I)}$, which are eventually compressed to reduce;

2. Given two peers P_1 and P_2, their descriptors $\mathcal{D}_1$ and $\mathcal{D}_2$, and any two segments $\mathcal{D}_1^{(i)} \in \mathcal{D}_1$ and $\mathcal{D}_2^{(i)} \in \mathcal{D}_2$ such that $\mathcal{D}_1^{(i)}$ and $\mathcal{D}_2^{(i)}$ represent the same partition in their respective descriptors, then the distance $d\left(\mathcal{D}_1^{(i)}, \mathcal{D}_2^{(i)}\right)$ computed between the two segments is a lower-bound of the exact distance $d_{\mathcal{D}_1}(\mathcal{D}_2)$;

3. Given two pairs of segments $\mathcal{D}_1^{(i)}, \mathcal{D}_1^{(j)} \in \mathcal{D}_1$ and $\mathcal{D}_2^{(i)}, \mathcal{D}_2^{(j)} \in \mathcal{D}_2$, such that $\mathcal{D}_1^{(i)} \subseteq \mathcal{D}_1^{(j)}$ and $\mathcal{D}_2^{(i)} \subseteq \mathcal{D}_2^{(j)}$, then $d\left(\mathcal{D}_1^{(i)}, \mathcal{D}_2^{(i)}\right) \leq d\left(\mathcal{D}_1^{(j)}, \mathcal{D}_2^{(j)}\right)$, i.e., the previous property is incremental.

A vector space provided with the Euclidean distance follows the previous rules:

Example 4.1 (Euclidean distance)**.** *Let $\mathcal{D}_1$ and $\mathcal{D}_2$ be two vectors whose coordinates are $\langle v_{1,1}, \ldots, v_{n,1}\rangle$ and $\langle v_{1,2}, \ldots, v_{n,2}\rangle$, respectively, and $\mathcal{D}_1^{(i)}$ and $\mathcal{D}_2^{(i)}$ the projection of the two vectors in a subspace such that $\mathcal{D}_1^{(i)} = \langle v_{1,1}, \ldots, v_{i,1}\rangle$ and $\mathcal{D}_2^{(i)} = \langle v_{1,2}, \ldots, v_{i,2}\rangle$. The*

Euclidean distance does guarantee the previous properties:

$$d\left(\mathcal{D}_1, \mathcal{D}_2\right) = \sqrt{\sum_{j=1}^{n} (v_{j,1} - v_{j,2})^2} \geq \sqrt{\sum_{j=1}^{i} (v_{j,1} - v_{j,2})^2} = d\left(\mathcal{D}_1^{(i)}, \mathcal{D}_2^{(i)}\right)$$

If the previous rules are satisfied, then a peer descriptor can be sent to the remote peer one segment at a time and the remote peer can decide to early stop the meeting if it realizes that the distance between them is growing too much. In Appendix A we show how a LM can be partitioned in I independent segments and how it can be represented through a set of Bloom-filters [Blo70]. We further present a technique which exploits this representation to efficiently compute the distance between the peers, allowing only a small and controllable error on the final value.

4.3.1 Algorithms for MON Construction and Maintenance

In this section we present the algorithms needed to build the semantic overlay as outlined in Section 4.2. Each peer aims to find its k *nearest neighbors* in the network, in terms of the introduced metric distance. The nearest neighbor problem in a metric space has been widely studied for centralized environments [HS03, ZADB05], where the space regions relevant for the query are distinguished from the non relevant ones by exploiting information of an index data structure built on the data collection; since in our scenario we cannot rely on such an index structure, each peer needs to perform a series of random meetings with other peers in order to find its neighbors and build its own view of the network.

Whenever a meeting (see Algorithm 3) occurs, the two peers exchange information in order to compute a lower-bound on their similarity distance [2], so that they can decide whether they should become MON neighbors or not. The *search radius* of a peer is defined as the distance from the farthest peer in the current list of nearest neighbors, and is used to prune the search space of those regions whose distance (or lower-bound) is too high. To increase the probability of success of future meetings, after a peer P_Θ has completed a meeting, it computes a lower-bound between the met peer P_Ψ and all peers currently in its neighbors list and "suggests" these values to P_Ψ (see Algorithm 4). Algorithm 5 shows the strategy by which peers are chosen to initiate a meeting with. Note that, in order to minimize the chance that two peers meet too frequently, each peer maintains a list of peers already met, along with their distances or lower-bounds.

[2] In this section the lower-bound is intended to be an approximation on the final distance, to not be confused with the MinDist indicator, which expresses the minimum distance between a query and a region.

The random choice of meeting partners can be facilitated by the underlying network, using, for example, a distributed hash table or epidemic message spreading. The random choice may be biased, based on different criteria such as network distance (e.g., measured in round-trip latency). In our actual algorithm, we use a priority queue which stores peers (and their lower-bounds) suggested by others: the highest priority is given to the peer with the lowest lower-bound, in order to shrink the initiating peer's search radius as quickly as possible [HS03].

For additional flexibility and robustness to network dynamics (i.e., peer failures, churn, and evolving contents of peers), each peer periodically initiates new meetings, even if it has reached its "optimal state".

The Meeting Algorithm

Algorithm 3 $meeting(P_\Theta, P_\Psi)$

Require: P_Θ, P_Ψ, r_Θ and r_Ψ

1: $i \leftarrow 0$ and $lb_\Theta \leftarrow 0$
2: **while** $(lb_\Theta < r_\Theta \wedge i < I)$ **do**
3: P_Θ: $receive(P_\Psi, BF_\Psi^{(i)})$ and $lb_\Theta.update(\cdot)$
4: $i \leftarrow i + 1$
5: **if** $(i = I)$ **then** {All BF's were received}
6: P_Θ: $\widetilde{d} \leftarrow lb_\Theta$
7: **if** $(\widetilde{d} < r_\Theta)$ $neighbors(P_\Theta).Insert([P_\Psi, \widetilde{d}])$
8: P_Θ: $send(P_\Psi, \widetilde{d})$
9: **if** $(\widetilde{d} < r_\Psi)$ **then**
10: P_Ψ: receive P_Θ's Bloom-filters
11: P_Ψ: $neighbors(P_\Psi).Insert([P_\Theta, \widetilde{d}])$
12: **else** {P_Ψ is pruned on lower-bound}
13: P_Θ: $send(P_\Psi, lb_\Theta)$
14: **if** $(lb_\Theta \geq r_\Psi)$ **then** Stop
15: $i \leftarrow 0$ and $lb_\Psi \leftarrow 0$
16: **while** $(lb_\Psi < r_\Psi \wedge i < I)$ **do**
17: P_Ψ: $receive(P_\Theta, BF_\Theta^{(i)})$ and $lb_\Psi.update(\cdot)$
18: $i \leftarrow i + 1$
19: **if** $(lb_\Psi \geq r_\Psi)$ **then** Stop
20: **else** P_Ψ: $\widetilde{d} \leftarrow lb_\Psi$
21: P_Ψ: $neighbors(P_\Psi).Insert([P_\Theta, \widetilde{d}])$

Algorithm 3 shows the procedure by which two peers P_Θ and P_Ψ exchange information about their LM's, compute a lower-bound and, if necessary, the distance $\widetilde{d}$. The search

radii of P_Θ and P_Ψ are denoted as r_Θ and r_Ψ, respectively, and lb_Θ and lb_Ψ are the lower-bounds computed at each peer. As explained in Section A.2, the lower-bound in lines 1-4 (and 15-18) is computed by using a partial representation of the two Language Models, i.e., only the subset of the I BF's that is currently available. Thus, as a new BF arrives from the remote peer, the value of the lower-bound increases (line 3 for P_Θ and 17 for P_Ψ), and when no BF's are left, eventually turns into the distance $\widetilde{d}$ (i.e., the second condition at lines 2 and 16 fails). This technique minimizes the number of BF's sent when an unsuccessful meeting occurs. Unfortunately, if P_Θ is not interested in the meeting anymore, P_Ψ needs to restart the computation (line 15), since, in general, the i-th BF in different peers contains different terms. Note that if P_Θ has already computed the distance $\widetilde{d}$, P_Ψ needs P_Θ's BF's only to make it one of its MON neighbors (lines 9-11).

Algorithm 4 *gossip*(P_Θ, P_Ψ)

Require: lb_Θ, $\widetilde{d}_1, \ldots, \widetilde{d}_k$, PQ_{met}

1: V_Θ: vector containing up to k elements
2: **for** $i = 1, \ldots, k$ **do**
3: $\quad P_i$: i-th neighbor of P_Θ
4: $\quad V_\Theta[i] = \langle P_i, |lb_\Theta - \widetilde{d}_i| \rangle$
5: P_Θ: $send(P_\Psi, V_\Theta)$
6: $V_\Psi \leftarrow receive(P_\Psi, V_\Psi)$
7: **for all** $(\langle P, lb \rangle \in V_\Psi)$ **do**
8: $\quad$ **if**$(P \notin PQ_{met})$ **then** $PQ.Enque(P)$ with priority lb

Algorithm 4 shows the procedure by which any two peers P_Θ and P_Ψ exchange information about their knowledge of the network topology. We present the algorithm from P_Θ's viewpoint denoting with lb_Θ the lower-bound it computed on $\widetilde{d}(\Theta, \Psi)$ and with $\widetilde{d}_1, \ldots, \widetilde{d}_k$ the distances between P_Θ and its k nearest neighbors $P_1, \ldots, P_k$. Another input to the algorithm is the list PQ_{met}, containing those peers that P_Θ has met during previous meetings. This list, filled at the end of Algorithm 3, was not shown to simplify the notation.

By applying the triangular inequality to P_Θ, P_i and P_Ψ, it is possible to compute a lower-bound on the distance $\widetilde{d}(P_\Psi, P_i)$ without the need of any additional computation (line 4). P_Θ, then, builds the vector V_Θ with the pairs $\langle P_i, lb \rangle$ and sends it to P_Ψ (line 5). When receiving P_Ψ's vector, P_Θ extracts all P_Ψ's neighbors and inserts them in its candidate list PQ, with priority given by the lower-bound computed by P_Ψ (lines 6-8).

Algorithm 5 *choosePeer*()

Require: PQ, α

1: $\Pr_{rm} \leftarrow \Pr\{random\ meeting\ occurs\}$
2: **if** $(PQ \neq \oslash$ **and** $\Pr_{rm} > \alpha)$ **then** $P \leftarrow PQ.Deque()$
3: **else** $P \leftarrow chooseRandomPeer()$
4: $meeting(P_\Theta, P)$
5: $gossip(P_\Theta, P)$
6: **if** $r_\Theta > \widetilde{d}_k$ **then** $r_\Theta \leftarrow \widetilde{d}_k$
7: **if** $(PQ.First().lb > r_\Theta)$ **then** $PQ.Clear()$

Algorithm for Network Maintenance

Algorithm 5 shows the procedure adopted by peer P_Θ to choose the next peer to meet. When a peer joins the network, its search radius is set to the maximum distance, so that the first k meetings are always successful. Note that, in general, the peer to meet is chosen as the top one in the priority queue PQ, i.e., the one with the smallest lower-bound; however, it might happen that PQ is empty and a peer is chosen at random.

To add flexibility to the meeting strategy, and to increase the chances that even peers far from each other will eventually meet, with a certain probability α a random meeting will occur even with a non-empty priority queue (lines 1-2). When a peer P has been chosen, the *meeting*() and *gossip*() procedures are invoked (lines 4-5) and, if P has become a neighbor of P_Θ, the search radius is updated (line 6). Note that if r_Θ becomes smaller than any lower-bound in PQ (line 7), the priority queue may be safely emptied because it does not contain any peer whose distance is lower than r_Θ.

4.3.2 Cost Analysis

In this section, we derive a cost model for Algorithm 3, which describes a meeting between two peers P_Θ and P_Ψ. In the first phase of the algorithm, peer P_Θ asks P_Ψ for one BF at a time and, at each step i of the algorithm, checks whether the condition $lb^{(i)} < r_\Theta$ is satisfied, before continuing the meeting. The second phase is analogous to the first one, with P_Ψ asking P_Θ for its BF's. To characterize the cost of the meeting, we introduce the probability distribution $G_P^{(i)}(r)$, which denotes the probability that, upon the reception of i BF's, peer P is still interested in the meeting:

$$G_P^{(i)}(r) = \Pr\left\{lb^{(i)} \leq r,\ lb^{(i)} = lb(BF^{(1)}, \ldots, BF^{(i)})\right\}. \tag{4.5}$$

The cost of the i-th step of the first phase of the algorithm, then, is characterized by the cost $\mathcal{C}_{BF}$ of a Bloom-filter (proportional to its size m), multiplied by the probability that

the BF is needed by P_Θ. The cost $\mathcal{C}_\Theta$ for the first phase, then, is given by

$$\mathcal{C}_\Theta = \mathcal{C}_{BF} \cdot \left[1 + \sum_{i=1}^{I-1} G_P^{(i)}(r_\Theta)\right]. \tag{4.6}$$

Note that the sum ranges between 0 and $I-1$, with $G_P^{(0)}(r) = 1$, assuming that no prior information is given about peers that have not been met.

The cost $\mathcal{C}_\Psi$, which characterizes the second phase of Algorithm 3, can be expressed as:

$$\mathcal{C}_\Psi = G_P^{(I-1)}(r_\Theta) \cdot \mathcal{C}_{\Psi,1} + \left[1 - G_P^{(I-1)}(r_\Theta)\right] \cdot \mathcal{C}_{\Psi,2} \tag{4.7}$$

where P_Θ, with probability $G_P^{(I-1)}(r_\Theta)$, has computed the distance $\widetilde{d}$ (i.e., it has asked for I BF's) and, with probability $1-G_P^{(I-1)}(r_\Theta)$, has computed a lower-bound using $i^* < I$ BF's.

$$\begin{cases} \mathcal{C}_{\Psi,1} = I \cdot \mathcal{C}_{BF} \cdot G_P^{(I)}(r_\Theta) \\ \mathcal{C}_{\Psi,2} = \mathcal{C}_{BF} \cdot \left[1 + \sum_{i=1}^{I-1} G_P^{(i)}(r_\Psi)\right] \cdot G_P^{(i^*)}(r_\Psi) \end{cases} \tag{4.8}$$

In $\mathcal{C}_{\Psi,1}$'s expression, P_Ψ will not ask for any BF, unless it wants P_Θ to become its neighbor (a lower-bound computed with I BF's is equal to the distance $\widetilde{d}$). On the other hand, to compute $\mathcal{C}_{\Psi,2}$, we take into consideration the probability that P_Ψ stops the meeting by means of P_Θ's lower-bound (we approximate the probability $\Pr\{lb^{(i^*)} \leq r_\Psi\}$ with $G_P^{(i^*)}(r_\Psi)$); if this is not the case, and P_Ψ has to compute its own lower-bound, the same considerations valid for $\mathcal{C}_\Theta$ apply.

The overall cost is proportional to the number I of BF's that are sent across the network, which should be kept as low as possible (note that this can be done at the price of a higher absolute error ε_{abs}); on the other hand, by increasing I, the average number n of elements inserted in each BF will automatically decrease, thus reducing the cost $\mathcal{C}_{BF}$ as well. This suggests that there exists an optimal I which minimizes transmission costs: computing this optimal value, however, is not easy because the data distribution among peers, which is represented by the $G_P^{(i)}(r)$ functions, should also be taken into account, since they play a key role in the cost model. If we consider, for example, $\mathcal{C}_\Theta$ and $\mathcal{C}_{\Psi,2}$ in Equations 4.6 and 4.8, we observe that whenever we have to deal with a set of $G_P^{(i)}(r)$ functions that are fast decreasing on i, a higher number I of BF's could still be acceptable because, on average, only a subset of the whole LM will be asked by a peer during a meeting.

4.4 Query Processing in a MON

The lack of information about the distribution of the data among peers strongly reduces the efficiency of P2P systems. This problem is alleviated by adopting a MON where

peers become neighbors if they share similar content. In this section we investigate the performances of similarity queries in a MON. We briefly review the most common query answering techniques in unstructured P2P networks (Section 4.4.1). In Section 4.4.2 we present three different routing strategies specifically designed to take advantage of the MON structure as it is opposed to a standard P2P network. The main difference among these techniques is the information that they are allowed to exploit when taking routing decisions. Our experimental evaluation, in Section 4.5, is conducted assuming a "stable" network, i.e., disregarding the dynamics of the networks.

4.4.1 Query Processing in P2P networks

Query answering in unstructured networks is usually a hard task, because the absence of correlation between the data stored at different peers and the network topology prevents the peers from having any knowledge about the network topology that might be used during query routing. In the Gnutella (protocol 0.4) network [Gnu], the first and most widespread used P2P network, a *breath-first* strategy is adopted. When a peer receives a query it simply broadcasts it to all its neighbors. To avoid flooding the whole network, a TTL (time-to-live) is used, i.e., the query expires after having being forwarded TTL times. Apart from the obvious network overhead that is generated, there are two other problem: choosing the correct TTL is not easy and the data objects that are too far away from the querying peer cannot be retrieved.

An alternative routing strategy is the Random Walk [LCC+02], which consists in forwarding any incoming query to a randomly chosen neighbor and relies on the assumption that, since the most popular data objects are also those that have a higher replication rate, than the randomness of the routing strategy still allows to achieve good performances. In [CGM02] each peer stores, for each one of its neighbors, a *routing index* [CGM02], which summarizes the *content which can be found if the query were forwarded to that neighbor.* Routing indices are, then, exploited to make the queries follow the most promising paths.

A different philosophy is that of Semantic Overlay Networks [CGM04], unstructured networks in which peers with related content tend to be directly linked or, at least, they are at a short hop-distance from each other. The idea is to form a topically focused cluster, such that the task of finding a "good" answer to a query becomes easy as soon as the clusters that best match the query are found. The fireworks query model [KNS02] is based on a system in which peers are clustered based on a similarity function $sim(P_i, P_j)$, where P_i and P_j are peers. When, at query time, peer P_1 generates the query q, a random walk is initiated in the network, using only the long-distance links of the encountered peers; any peer P that is met during the walk is matched against the query and if the

condition $sim(q, P) < \epsilon$ is verified (for some value ϵ), then the query has found the target cluster. The query, then, goes through a limited broadcast (only short-distance links are used) which aims at reaching all regions around the target peer that might be able to answer it.

For a more comprehensive survey on query answering techniques in P2P networks we refer to [BCLP05].

4.4.2 Algorithms for Query Routing

We have presented the MON building and maintenance algorithms in Section 4.3. In this section we investigate the problem of query routing in a MON by assuming that each peer has found its k_s short-distance neighbors and has chosen k_l other peers to be its long-distance ones. In particular, we assume that the network has reached a "stable" state, i.e., for each peer the short-distance neighbors it has found are its *true* k_s nearest neighbors in the network.

Given a query at time t, the network can be partitioned in two sets $\mathcal{L}_{\mathcal{NV}}(t)$ and $\mathcal{L}_{\mathcal{V}}(t)$ of peers which represent:

Non-visited peers $\mathcal{L}_{\mathcal{NV}}(t)$: A list of peers which, at time t, have not been visited yet. We can distinguish among peers *relevant* with respect to the query and peers *not relevant*. The internal composition of $\mathcal{L}_{\mathcal{NV}}(t)$ depends on the current result $\widetilde{\mathcal{R}}$. If a k-NN search is performed, whenever a new object is found, the search radius may be reduced, thus making it possible to prune unexplored regions. When no more relevant peers are left, the search should be stopped because no better results can be found by the current query.

Visited peers $\mathcal{L}_{\mathcal{V}}(t)$: This are the peers which, at time t, have been already visited by the query, i.e., a sort of *search history*. The search history is used to avoid contacting the same peer twice.

Notice that, since we do not consider a dynamic network, the two sets depends on the time t only indirectly (then we can omit its indication) through the number of *steps* of the algorithm, because when a peer is extracted from $\mathcal{L}_{\mathcal{NV}}$ it is inserted in $\mathcal{L}_{\mathcal{V}}$.

In the ideal case, the union of the two lists $\mathcal{L}_{\mathcal{NV}} \cap \mathcal{L}_{\mathcal{V}}$ coincides with the whole network,and at step 0 all peers are in $\mathcal{L}_{\mathcal{NV}}$. A query q, then, chooses, at each step, the best peer to be visited according to the query criteria (see Chapter 3 for details). The real scenario, however, is quite different: q only knows the peers that it has already visited *plus* their neighbors. At step 0 $\mathcal{L}_{\mathcal{NV}}$ contains the querying peer and its neighborhood and $\mathcal{L}_{\mathcal{V}}$ is empty, however both of them may become very long as soon as q is forwarded to

new peers and it might become a problem to append them to the query, because their size might grow too much. the

In the following we present three routing strategies that differ in the way they manage and store the lists $\mathcal{L}_{\mathcal{NV}}$ and $\mathcal{L}_{\mathcal{V}}$. In Section 4.5, then, we will evaluate them experimentally.

Given either $\mathcal{L}_{\mathcal{NV}}$ or $\mathcal{L}_{\mathcal{V}}$ lists, they can be maintained as lists of peers attached to the query, or distributed lists stored at the peers where the query is routed. Notice that:

- If $\mathcal{L}_{\mathcal{NV}}$ is attached to the query, for each peer $P_i \in \mathcal{L}_{\mathcal{NV}}$, also the indicator value $\Psi\left(q, \mathcal{D}_{\mathcal{D}_i}\right)$ needs to be remembered;
- A solution based on distributed lists means that each query has a state at each peer it traverses, with the need of implementing a synchronization technique.

In the following we present the three routing strategies:

Biased random walk: Neither $\mathcal{L}_{\mathcal{NV}}$ nor $\mathcal{L}_{\mathcal{V}}$ is maintained. Peers always route incoming queries toward the most promising neighbor with respect to the quality criterion, without considering the case of already visited regions.

Best-first: Both $\mathcal{L}_{\mathcal{NV}}$ and $\mathcal{L}_{\mathcal{V}}$ are maintained within the query which, at each step, is always routed toward the best peer among those that have been encountered so far. If the quality criterion is accurate enough, this strategy should guarantee high quality results, because the most promising peer is always contacted first. However, major problems arise if we consider the overhead of transmitting such a query in the network because the two lists might become big and space consuming.

Depth-first: The list of visited peers $\mathcal{L}_{\mathcal{V}}$ is stored by the query, but $\mathcal{L}_{\mathcal{NV}}$ is distributed. A peer which receives an incoming query routes it to the most promising neighbor and the chosen neighbor does the same, and so on; the query descends the network until it finds a peer whose neighbors have been already visited or they are not relevant with respect to the quality criterion. The query, then, tracks back to the last visited peer and so on until it finds a yet non-visited neighbor. The network assumes a hierarchical structure which is visited in a depth-first fashion and whose root is the peer which originated the query.

Finally, in Section 4.5 we present three routing strategies that differ in the way they manage the lists $\mathcal{L}_{\mathcal{NV}}$ and $\mathcal{L}_{\mathcal{V}}$. In particular, we show that the more a query is allowed to know about the lists, the more accurate are the retrieved results.

4.5 Experimental Evaluation

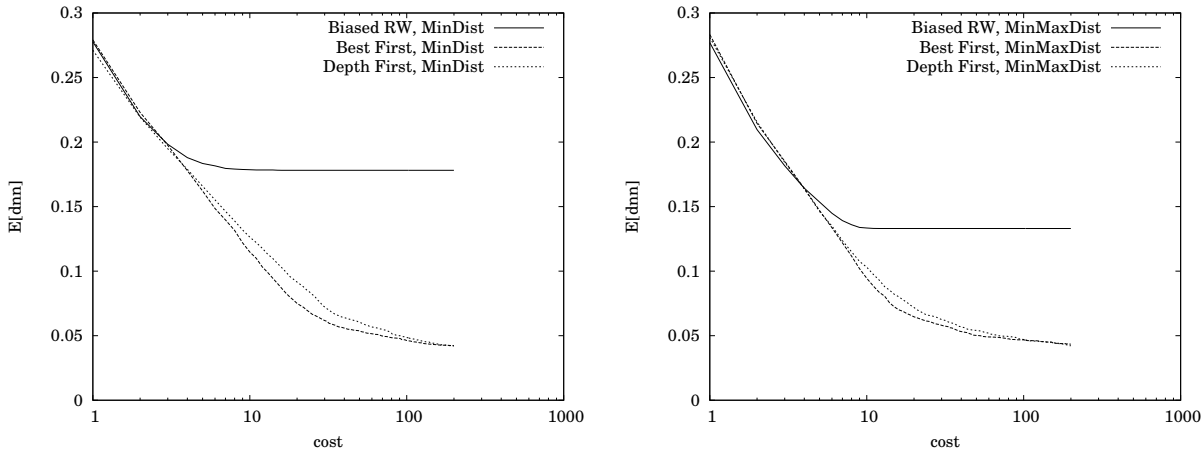
We have tested our system using the `Corel` data set that was also used for the completely connected network in Section 3.6. Our experimental setup consisted in the following points:

Data partitioning The `Corel` data set (67457 objects) has been partitioned into 266 data regions containing, approximately, 253 objects.

Queries assignment The query distribution follows the data distribution. We run 681 queries which are assigned to peers randomly.

Network maintenance The network is built according to the techniques presented in Section 4.3. Each peer is represented by a centroid (i.e., the center of the region) and is linked (short-distance links) to its k nearest neighbors in the network.

Long-distance links Each peer further have k *long-distance* links which connect it to k random peers in the network. We utilize long-distance links only when a short-distance link is not available, either because they have been all explored or because their lower-bound is too high.



(a) (b)

Figure 4.2: Average cost of the query with different routing strategies using theMinDist (a) and MinMaxDist indicators.

In Figure 4.2(a) and 4.2(a) we show the performances of an on-line query that is executed according to a cost-based control policy. The regions of the network are ordered with respect to the values of the indicator (MinDist in Figure 4.2(a) and MinMaxDist in Figure 4.2(b)). We can make the following observations:

- If the routing is based on a biased random walk, the performances are very poor, because the lack of "memory" on the visited regions avoid the query to take unexplored paths that might eventually lead to better results;
- The performances of the best-first and the depth-first strategies have approximately the same performances (the best-first performs slightly better);
- As we would have expected from the formal analysis of the previous chapter, the MinMaxDist indicator is still more accurate that the MinDist.

In general, the good performances that can be achieved by the similarity search paradigm in a P2P network justifies its adoption is such a scenario. Moreover, the existence of powerful tools of analysis in the case of completely connected networks leaves open to possible improvements.

Finally, we would like to point to Appendix B, where we show a real setting in which the property of a network to be embedded in a metric space allows to achieve anonymity and censorship resistance in a P2P network.

Chapter 5

Conclusions

In this thesis we presented a framework for processing approximate similarity queries in general metric spaces.

The main conclusions we can draw from our work are the following:

- The problem of approximate similarity search in complex data bases cannot be considered fully solved, which contrasts with the problem of exact search, for which an optimal solution does exists [BBKK97, HS03].

- We have developed a model that predicts the average cost of a query with high precision. This model is based on very general assumptions and only exploits probabilistic information about the data regions.

- We have derived an optimal-on-the-average schedule and we have shown how it can be conveniently implemented, given the statistical description of a data region.

- We have studied the problem of approximate similarity search to the case of P2P networks. In this scenario, we have proposed the MON, a novel data structure for processing these queries and we have applied it to two completely different scenarios.

- We have applied the notion of MON to the field of Information Retrieval in P2P networks, where the similarity between peers is computed by a metric distance which takes advantage of the KL-divergence and a Language Model is represented through a set of Bloom filters.

- We have applied the notion of MON to a problem of security in P2P networks. We have presented a Clouds, an anonymous and censorship-resistant search infrastructure where anonymity is achieved by clouds, namely peer clusters hiding the actual identity of communication participants and censorship resistance is guaranteed by

the query and the answers to follow different paths in the network. By exploiting the metric properties of the MON we were able to give probabilistic guarantees on the level of censorship resistance achieved by Clouds.

5.1 Future Directions

Throughout the thesis we have pointed out several interesting issues for future research. These include:

- Our cost model only applies to completely connected networks and its analytical complexity prevents us to extend it to any other kind of network. We plan to find an alternative way to estimate the cost of a query in hierarchical and partially connected networks. This might involve the introduction of suitable approximations or non-obvious simplifying assumption.

- We plan to extensively test the MON in the near future, in particular in terms of systems performances. We are particularly interested in query processing techniques which can give some kind of quality guarantees on the final result. This may involve combining our approach with distributed indexes for metric spaces (e.g., work along the lines of [FGZ05, BNFZ06]).

- Referring to its application to the field of information retrieval [LW06], we believe that a MON has a broader applicability. In many modern information systems, not only distributions in data collections, but also user preferences, recommendations, profiles, and other elements of user behavior can be described in terms of LM-style distributions with information-theoretic comparisons being natural measures of similarity. Thus, embedding measures like KL divergence or JS divergence in a metric search space is an appealing direction for much broader classes of social networks.

- Finally, Clouds is currently under development and we plan to evaluate, apart from its efficiency and effectiveness, the anonymity and censorship resistance achieved.

Appendix A

Compressing a Language Model

In this chapter we address the problem of how to compress a Language Model by using a set of I Bloom-filters and how this representation can be exploited to efficiently compute the distance between the peers, allowing only a small and controllable error on the final value. At the end of the chapter, we will give practical guidelines to the design of the system, in terms of correctly choosing the number and the size of the Bloom-filters used to represent the Language Model.

A.1 A Solution based on Bloom-Filters

Bloom filters have received a lot of attention since their introduction in [Blo70], because of their ability of giving probabilistic error guarantees when representing a set of elements drawn from any given domain. A Bloom-filter is a vector of m bits initially set to 0, which is used to represent a set of n elements and to subsequently test their membership to the set. Each element is drawn from a finite universe U and is hashed by h hash functions which independently set one of the m bits. To test if an element is in the Bloom-filter, the h hash functions are applied to it: if each bit addressed by the hash functions is 1, then the searched element "should" be present, being the false positive probability given by

$$fpp \approx \left[1 - e^{-\frac{hn}{m}}\right]^h. \tag{A.1}$$

To represent a LM in a compact manner, first note that when comparing two LM's by the square root of the JS divergence, the most important contributors to a large distance are the terms for which the two LM's have *widely differing weights*. So these terms should be captured more precisely than the terms for which the two LM's show low differences in the term weights. Our approach conceptually compresses an LM in two steps. First, we construct a histogram of possible term weights with a small number

of equi-width histogram cells. Conceptually, each histogram cell is associated with the subset of terms whose weights fall into the cell's boundaries. As a second step, we then compress these subsets by mapping them onto a Bloom-filter (BF). Thus, we arrive at one BF per histogram cell of each LM.

A histogram for P_Θ which describes the distribution of terms in Θ is built over the domain of weights by partitioning it in I disjoint intervals of equal-size, such that the i-th interval corresponds to the range $\left[\theta_H^{(i)} - \frac{\delta}{2}, \theta_H^{(i)} + \frac{\delta}{2}\right]$. Each interval is represented by a Bloom-filter which contains all terms whose weight belongs to it: the approximate Language Model $\widetilde{\Theta}$ is represented by the set $\{BF(\theta_H^{(i)})\}_{i=1,\dots,I}$, where $BF(\theta_H^{(i)})$ is the i-th Bloom-filter. To extract from a given set of BF's the weights which characterize the corresponding $\widetilde{\Theta}$, the following rules apply:

1. For each term $t_i \in \widetilde{\Theta}$, its weight $\widetilde{\theta}_i$ is $\theta_H^{(i)}$ if $t \in BF_\Theta(\theta_H^{(i)})$.

2. When a term belongs to two or more BF's at the same time, we suspect a false positive. This case should happen with very low probability, but it may happen because of the lossy compression characteristics of Bloom-filters. We could then make a heuristic choice such as choosing the average value of the term weights, or we may directly ask the original peer for the correct weight. In any case, the metric properties of the data space are preserved if, for each peer, only one of the these policies is consistently applied.

Finally, we can define the approximate distance $\widetilde{d}$ between two peers P_Θ and P_Ψ as

$$\tilde{d}(\Theta, \Psi) = d(\widetilde{\Theta}, \widetilde{\Psi}). \tag{A.2}$$

by observing that, if there exists only one possible approximation for each LM, then $\widetilde{d}$ will still be a metric distance over the space of all peers.

In the following, we study the behavior of Equation 4.3, given the approximation introduced by the BF-based histograms. Note that representing a weight distribution with a histogram is a well-known technique in the database community and has already been extensively studied, however the problem we are facing now is quite different because we do not need to estimate the performance of this technique *per se*, rather we want to characterize the approximation introduced in the computation of the distance function.

We start our analysis by introducing $\varepsilon_{\Theta,max}$, defined as the maximum distance between a LM and its approximation, i.e., as $\max\{d(\Theta, \widetilde{\Theta})\}$ for any Θ. By applying the triangular inequality to any two peers P_Θ and P_Ψ whose exact distance is d, we can write that:

$$\max\{0, \widetilde{d} - 2 \cdot \varepsilon_{\Theta,max}\} \leq d \leq \widetilde{d} + 2 \cdot \varepsilon_{\Theta,max} \tag{A.3}$$

where $\widetilde{d}$ is the the approximate distance $\widetilde{d}(P_\Theta, P_\Psi)$. From Equation A.3, then, we compute an upper-bound on the absolute error ε_{abs}, defined as the the difference between the approximate and exact distance between any pair of peers:

$$\varepsilon_{abs} = |d(\Theta, \Psi) - d(\widetilde{\Theta}, \widetilde{\Psi})| \leq 2 \cdot \varepsilon_{\Theta,max}. \tag{A.4}$$

Before fully characterizing $\varepsilon_{\Theta,max}$, we consider the distance $d(\Theta, Lmax)$ as it is computed by Equation 4.3 and move the square-root inside the sum; then, we find that:

$$d(\Theta, \widetilde{\Theta}) \leq \sum_{t_i \in \mathcal{T}} \sqrt{d_{JS}(\theta_i, \widetilde{\theta_i})}. \tag{A.5}$$

Note that each term in Equation A.5 is a single-term LM: for any pair of weights θ_i and $\widetilde{\theta}_i$, then, if the difference $|\theta_i - \widetilde{\theta}_i|$ is maximized, their JS-divergence is maximized as well. Since each approximate weight $\widetilde{\theta_i}$ is drawn from the set $\{\theta_H^{(1)}, \ldots, \theta_H^{(I)}\}$, we conclude that the natural choice for θ_i is among $\theta_H^{(i)} - \frac{\delta}{2}$ and $\theta_H^{(i)} + \frac{\delta}{2}$. If we define ε_{lat} as the maximum distance between any θ_i and its approximation $\widetilde{\theta_i}$, i.e.,

$$\varepsilon_{lat} = \max_d \left\{ d = \sqrt{d_{JS}\left(\theta_H^{(i)}, \theta_H^{(i)} \pm \frac{\delta}{2}\right)},\ \forall i \right\}, \tag{A.6}$$

we obtain that $\varepsilon_{\Theta,max} = N \cdot \varepsilon_{lat}$, where $N = |\mathcal{T}|$ is the number of terms in Θ. We can further substitute this result in Equation A.4 and eventually find the expression of the absolute error

$$\varepsilon_{abs} \leq 2 \cdot N \cdot \varepsilon_{lat} \tag{A.7}$$

which, as we would have expected, is proportional to the approximation introduced by the BF-based histogram.

A.2 Efficient Technique for Distance Computation

A distance function based on the JS-divergence is computationally very expensive, because it consists of a sum of several terms whose mathematical expression is not trivial.

Consider two Language Models $\widetilde{\Theta}$ and $\widetilde{\Psi}$, represented by a set of I Bloom-filters as described in Appendix A.1, and suppose you want to compute the distance $d\left(\widetilde{\Theta}, \widetilde{\Psi}\right)$ between them. The computation can be highly optimized if we have a data structure M, that we represent as a matrix of dimension $[I \times I]$, which stores, at location (i, j), the result of the computation of the JS-divergence (Equation 4.2) between any two weights drawn from the set $\{\theta_H^{(1)}, \ldots, \theta_H^{(I)}\}$. The algorithm works as follows:

1. For each term $t_i \in \mathcal{T}$ search the two sets of Bloom-filters in parallel to obtain the two approximate weights $\widetilde{\theta}_i$ and $\widetilde{\psi}_i$.

2. When there is indication of a false positive (i.e., a term appearing in two BF's, see above), we heuristically choose a value or we directly ask the peer for the exact value.

3. In repeated computations (i.e., in a series of peer-pair meetings), we use the pre-computed value at location $M[\widetilde{\theta}_i, \widetilde{\psi}_i]$ or, if not present, we compute it explicitly and insert it into M.

4. We maintain an incremental sum over all terms and return the square-root of the final value.

This technique has the major advantage that M is common to all meetings and that its size is proportional to $O(I^2)$ which, considering the small number of BF's needed to represent a Language Model, leads to a negligible space consumption. The approximate distance is computed very efficiently and has the desirable property to be incremental (this will be fully exploited in Section 4.3.1 to allow peers to perform an early pruning during their meetings). Suppose, for example, that P_Θ has received from P_Ψ only $I' < I$ BF's: the lower-bound is then obtained by applying Equation 4.2 only to the terms contained in the available BF's. By separately applying the square-root to the result, the computation does not need to start again when a new BF is received.

A.3 Choosing Bloom-filter Sizes

A key element for our architecture is to choose the correct number I and the size m of the Bloom-filters used to represent a peer. From Equation A.7, we observe that each term contributes to the absolute error with a value proportional to ε_{lat}: given Equation A.6, we need to compute the single-term JS-divergence between the generic θ_H and the lower- and upper-limits of its interval (we should consider the farthest one). If we denote with a and b the two weights such that $a < b$ holds, i.e. we indicate that $(a, b) = \left(\theta_H - \frac{\delta}{2}, \theta_H\right)$ or $(a, b) = \left(\theta_H, \theta_H + \frac{\delta}{2}\right)$, we characterize their JS-divergence in terms of Equation 4.4 as:

$$d_{JS}(a, b) = a \cdot \log \frac{2a}{a+b} + b \cdot \log \frac{2b}{a+b} \leq (b - a) \log \frac{b}{a} \tag{A.8}$$

The proof of this inequality is straightforward, by observing that the first term is always negative; the difference $b - a$ equals to $\frac{\delta}{2}$ in both cases and the logarithm is always positive

and increases with the two values getting closer to 0. The expression of ε_{lat} is, thus, the following:

$$\varepsilon_{lat} \leq \sqrt{\frac{\delta}{2} \cdot \log \frac{\theta_H^{(1)}}{\theta_H^{(1)} - \frac{\delta}{2}}}, \quad \theta_H^{(1)} \leq \theta_H^{(i)} \ \ \forall i \tag{A.9}$$

with $\theta_H^{(1)}$ being the smallest weight used to build a BF with. This result has a major impact on the tuning of the system. In particular, it suggests that variable-width histograms would be preferable as different weight ranges contribute differently to the overall approximation. We make the following observations that may lead to further optimizations: the first BF might contain only terms with a weight close to 0. Moreover, it produces the largest error. BF's representing higher weights should have wider histogram intervals, since they cause smaller absolute errors.

Consequently, we should use variable-width histogram boundaries for the BF approximation, with carefully tuned cell widths. Here we can apply results from selectivity estimation and data synopses in database systems [GMP02, KW03]. The analysis of the LM approximation error can be extended along these lines, but this is beyond the scope of the current paper.

The last step of our tuning procedure consists in choosing the size of the Bloom-filters: consider, for a moment, the false positive probability given by Equation A.1. If we use an optimal number of hash functions, given by $h = m/n \cdot \ln 2$ [BM03], Equation A.1 can be approximated by $fpp = \left(\frac{1}{2}\right)^{\frac{m}{n}}$, which gives us the possibility to tune the size of our BF's in order to achieve the desired value of fpp.

Unfortunately, this is not always the case: in particular, when the number n of elements inserted in the BF grows too much, the compression technique based on BF's might become not convenient anymore. Suppose, for example, that n gets close to $|\mathcal{T}|$ (which is the number of elements in the universe): to have a false positive probability $fpp \approx 1/2$ (which is indeed a very high value), we need at least $m \approx |\mathcal{T}|$ bits in the BF. On the other hand, if we use a bit vector of size $|\mathcal{T}|$, we can represent the whole universe without any error, as long as we assume to have a function which univocally maps the i-th object on the i-th bit of the vector.

Appendix B

Anonymity and censorship resistance in MON's

In this chapter we present *Clouds*, an anonymous and censorship resistant search infrastructure built on top of a MON. Anonymity is achieved by clustering semantically close peers so as to hide the actual identity of the communication participants. Censorship resistance is guaranteed by a cryptographic protocol securing the anonymous communication between the querying peer and the resource provider. Finally, theoretical results formalize the degree of anonymity and censorship resistance provided by the architecture.

B.1 Cloud-based Anonymity

The communication protocols traditionally adopted in P2P networks do not guarantee anonymity to any of the participants because they need to exchange their physical addresses in order to be identified in the network and the whole communication is made publicly known.

Our solution constitutes in hiding the identity of both the querying and the answering peer behind a group of other peers, which we call *cloud*. Anonymity in Clouds is achieved by implying that all peers in a cloud share the same probability of being involved in a communication that has this cloud as a start- or end-point. One of the first systems to employ this idea in a client/server model was Crowds [RR98], where the group of peers is referred to as *crowd*. Contrary to our approach, crowds rely on a trusted server to store the list of participating peers and the crowd topology is assumed to be known in advance.

A naive solution to achieve anonymous communication between two peers P and P' consists in removing any reference to their addresses from the messages they exchange and allow them to communicate through message broadcasts. This way, the identities of

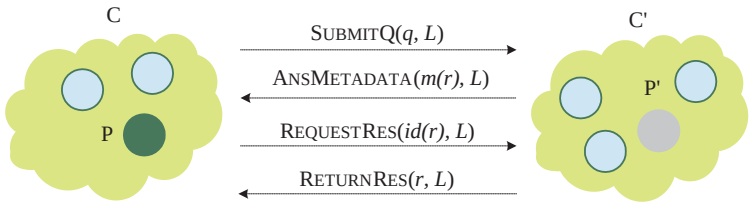


Figure B.1: A cloud-based P2P protocol, with P_i belonging to cloud $\mathcal{C}_i$.

the two peers are kept secret, at the price of a considerable increase in message traffic. A more sophisticated technique consists in defining small groups of semantically close peers, namely clouds. A cloud can be represented as a finite graph whose nodes are peers and edges are direct links among such peers. Each cloud is thus associated to a cloud-identifier $\mathcal{C}_{id}$, chosen by the cloud generating peer P. This identifier represents a *point in the space surrounding* P.

The communication between P and P' is described by the protocol of Figure B.1, where peers P and P' are hidden behind clouds $\mathcal{C}$ and $\mathcal{C}'$ and their physical addresses are replaced, in each of the exchanged messages, by their respective cloud identifiers. During the first step of the protocol the identifiers of all clouds that are traversed by the SUBMITQ message are collected into a list L, called *footprint list*. L initially contains only cloud C and whenever the query reaches a peer P_i such that $P_i \in \mathcal{C}_i$, $\mathcal{C}_i$ is added to L if and only if L does not already contain $\mathcal{C}_i$. This way, the footprint list always describes the path from P to P' in terms of clouds traversed. The only information a peer is allowed to know about the network consists of the physical addresses of its direct neighbors and the clouds they participate in. Thus, after the first message, the footprint list is utilized to efficiently reach P from P' and vice-versa. This routing technique differs significantly from the techniques currently used to guarantee anonymity, such as Freenet routing [CMH+02] or Onion routing [GRS99]. While in these approaches the path connecting the communicating peers is univocally determined, in our case intra-cloud broadcast introduces flexibility in the selection of paths. This way, the communication between the querying peer and the resource provider is not interrupted in the case of peer failure. Notice that intra-cloud broadcast can be further optimized by using techniques like gossiping protocols [EGKM04] or spanning trees [Dim87].

Finally, notice that the resulting network is an *unstructured network of clouds* mapped onto possibly overlapping sections of the original overlay network. Since cloud identifiers are points in the space, traditional search techniques as presented in Section 4.4.2 can be applied, i.e., given a query q at cloud $\mathcal{C}$, we look for cloud $\mathcal{C}'$ which may contain resources matching q. This property is also exploited in Section B.2 to achieve censorship resistance

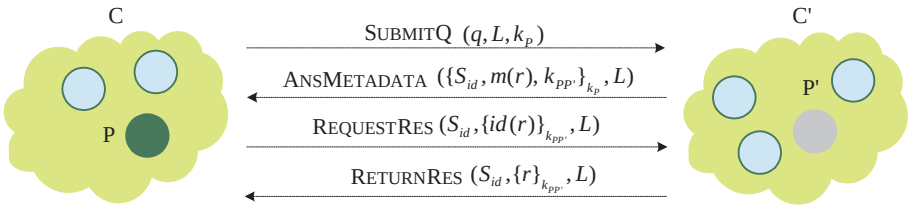


Figure B.2: A cloud-based cryptographic P2P protocol

against man-in-the-middle attacks.

Intuitively, using clouds as a way of communicating is sufficient to allow a peer to perform anonymous and efficient similarity search in a MON (for a more detailed discussion see [BLMT07]). In the following section we show how Clouds can be improved to achieve censorship resistance.

B.2 Cloud-based Censorship Resistance

While anonymity is a well-understood property that has been widely studied and formalized (see for example [SS99, Shm04, MVdV04]), only a few works have been proposed on the formal definition of censorship-resistance [PRW05, DA04]. In general, we can distinguish between two kinds of censorship, i.e., at storage and communication level. The former focuses on preventing hosts from storing undesired data, while the latter on filtering the communication according to message content. Traditional P2P approaches to censorship-resistance, such as Freenet [CMH+02], typically prevent censorship at storage level through data replication: information is spread all over the network, usually without the involved peers knowing what they are storing, and storage providers are kept anonymous. However, even if censorship at storage level is prevented, censorship at communication level may still be possible. For instance, a malicious host might inspect the content of incoming messages and block them, whenever they convey or refer to undesired data. In this section we present a cryptographic protocol for achieving censorship-resistance at communication level in a P2P environment.

B.2.1 Censorship Resistance: A First Attempt

Censorship may happen when nodes in the network are willing to prevent peers from sharing undesired data. A malicious node may censor a communication between two peers by inspecting the content of incoming messages and by blocking (*i*) undesired queries in SUBMITQ messages; (*ii*) undesired resource identifiers in ANSMETADATA and

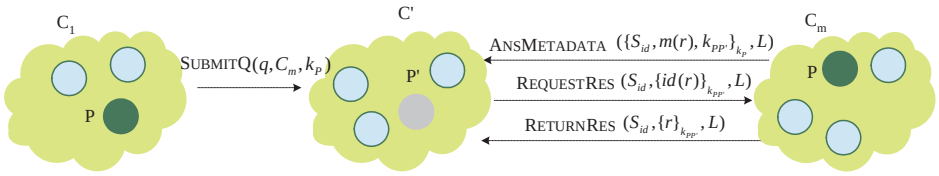


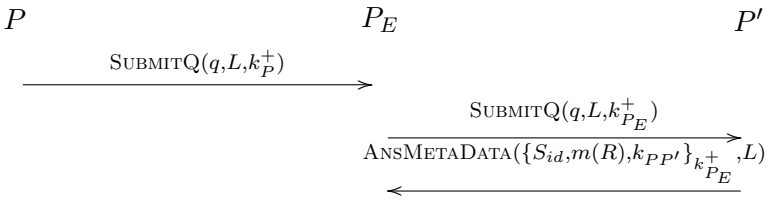
Figure B.3: A cloud-based cryptographic P2P protocol

REQUESTRES messages; and undesired resources in RETURNRES messages. In order to prevent censorship, we thus need to protect the information sent on the network.

The query is the only message that cannot be encrypted since it is supposed to be read by every peer. However, this is not a relevant problem, at least until the number of censoring nodes is limited, since queries are typically forwarded through multiple paths and blocking all of them requires a pervasive control of the network. In contrast, the rest of the messages follows a certain path and stopping them may suffice to censor the communication. For tackling this problem, we let P generate a key-pair, which is composed of a public key k_P^+ and a private key k_P^-. The public key is sent in SUBMITQ along with the query. In ANSMETADATA, P' encrypts resource metadata with k_P^+: this way only P can read the message, thus preventing censorship due to metadata inspection. The ciphertext contains also a fresh session key $k_{PP'}$ (i.e., a symmetric key only used in one session and then discarded). Notice that only P can read $k_{PP'}$, since it is encrypted with P's public key. Session key $k_{PP'}$ is used in REQUESTRES and RETURNRES for protecting message content, thus avoiding censorship due to identifier and resource inspection: the use of symmetric cryptography in place of public key cryptography allows for reducing the complexity of encryption and decryption operations.

B.2.2 Censorship Resistance: A Second Attempt

The previously proposed cryptographic protocol is conceptually appealing but provides only a limited degree of censorship-resistance, since a malicious peer residing in the path between P and P' can impersonate P and perform the following attack:



The attacker P_E replaces P's public key with its own key in SUBMITQ and, upon the reception of ANSMETADATA, gets the knowledge of the session key $k_{PP'}$ and, even worse,

can stop the communication between P and P' after inspecting the resource identifier $m(R)$. This attack is well-known in cryptography and it arises since P and P' are attempting to build up a secret communication without sharing any previous secret. Notice that the attacker has to reside in the path used by P and P' for exchanging the first message. On the one hand, the query is typically replicated and follows several paths, thus the attacker should control a large fraction of the network to perform such an attack: a single safe path between P and P' (i.e., it is not populated by a malicious peer) suffices to prevent the man-in-the-middle attack. On the other hand, the resource provider does not know which queries have been filtered by malicious peers and is thus forced to reply to all of them. This leads to replicated answers and inefficient communication.

We tackle this problem by guaranteeing that SubmitQ and AnsMetaData follow different paths and by letting the rest of the communication take place on the path followed by AnsMetaData. The refined version of the protocol is depicted in Figure B.3. Notice that in SubmitQ, P specifies a cloud different from $\mathcal{C}_1$ (i.e., the cloud which the message originates from): we assume that a peer belongs to several clouds and, in the specific example, let us suppose that P belongs to cloud $\mathcal{C}_m$ too. In order to perform the previously shown attack, the censoring peer should either control all the paths followed by SubmitQ or lie both in one of the paths followed by SubmitQ and in the path followed by AnsMetaData (see Figure B.3). In the latter case the attacker can use the destination cloud for binding SubmitQ and AnsMetaData messages, thus censoring the communication if the query is undesired. The former case requires a pervasive control of the network, while the probability that the latter occurs depends on the distance between $\mathcal{C}_1$ and $\mathcal{C}_m$ and the position of the attacker, as formally characterized in the next section. Finally, we remark that after the second message exchange, the cryptographic scheme is effective and the secrecy of the rest of the messages is preserved even if such messages follow the same path.

Finally, notice that Clouds does not rely on a public-key infrastructure, where every principal is associated to a key-pair and a trusted third party is in charge of public key distribution. In our approach, the querying peer generates a key-pair and distributes the public component (message SubmitQ) in an anonymous way. Furthermore, peers may decide to use a previously generated key-pair even in later protocol sessions.

B.2.3 A Probabilistic Model for Censorship Resistance

In this section we present an effective method for estimating the probability that a malicious peer P_E lies in the paths of both SubmitQ and AnsMetaData messages. As discussed in the previous section, whenever this happens the attacker can effectively mon-

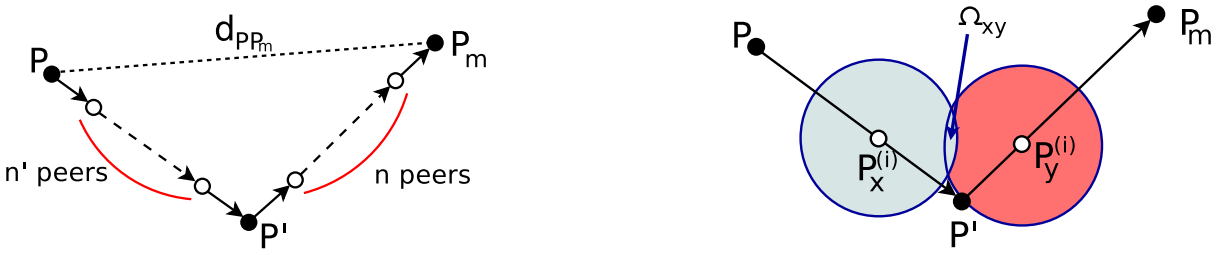


Figure B.4: (*a*) Graphical representation of the ideal paths followed by the first two messages and (*b*) overlap between two ball regions $\mathcal{B}_x$ and $\mathcal{B}_y$ belonging to the two paths.

itor and censor the communication. Devising an effective technique that reduces this probability is crucial for enhancing the censorship resistance of our architecture.

Assume that $P \in \mathcal{C}_1$ sends a SubmitQ message to $P' \in \mathcal{C}'$ and asks for the answer AnsMetaData to be sent to position P_m such that the distance $d(P, P_m)$ is d_{PP_m}. The *ideal paths* of the two messages are shown in Figure B.4(a) and they consist of two sets of peers aligned in straight lines. In general, however, peers do not occupy such ideal positions: in the following, $P_i^{(i)}$ denotes the position that a peer P_i would occupy if the routing were ideal and $P_i^{(r)}$ denotes its real position in the vector space (i.e., $P_i^{(r)}$ *is* P_i).

As it is discussed in Section B.2.2 a man-in-the-middle attack can take place if a malicious peer is in the paths of both SubmitQ and AnsMetaData messages, and thus responsible for two ideal positions $P_x^{(i)}$ and $P_y^{(i)}$ which belong to the path from P to P' and from P' to P_m, respectively. Without loss of generality, we assume that $P_x^{(r)}$ and $P_y^{(r)}$ lie within a finite distance r_i and r_y from $P_x^{(i)}$ and $P_y^{(i)}$, as it is shown in Figure B.4(b).

Intuitively, then, the probability that a man-in-the-middle attack takes place is proportional to the area defined by the intersection of the ball regions $\mathcal{B}_x$ and $\mathcal{B}_y$ and centered in $P_x^{(i)}$ and $P_y^{(i)}$ and it also depends on the number of peers in the network that are actually contained in that region. In the rest of the section we formalize this intuition and propose a method to compute the probability of a man-in-the-middle attack, given two peers P and P' and the position of a malicious peer P_E with respect to P.

In general, the real position of a peer P_i is unknown and can only be estimated in terms of its distance from $P_i^{(i)}$. Let $G_{P_i^{(i)}}(r)$ be the probability that $P_i^{(r)}$ is at a distance not greater than r from $P_i^{(i)}$, i.e., it is within a *ball region* $\mathcal{B}_i$ of radius r centered in $P_i^{(i)}$:

$$G_{P_i^{(i)}}(r) = \Pr\{d(P_i^{(r)}, P_i^{(i)}) \leq r\} \tag{B.1}$$

The distribution $G_{P_i^{(i)}}(r)$ in Equation B.1 depends on the specific position $P_i^{(i)}$: since it is not practical to compute and store it for each possible $P_i^{(i)}$, in our analysis we approximate

$G_{P_i^{(i)}}(r) \approx G(r)$, where given a random ideal position of a peer, $G(r)$ is the probability distribution of the real position.

As anticipated, the ball regions $\mathcal{B}_x$ and $\mathcal{B}_y$ in Figure B.4.*b* delimit the area in which there is a non-zero probability of finding $P_x^{(r)}$ and $P_y^{(r)}$. As formalized by Theorem B.1, two paths, then, intersect if

- $P_x^{(r)} \in \mathcal{B}_y$ and $P_y^{(r)} \in \mathcal{B}_x$ (assuming that they belong to $\mathcal{B}_x$ and $\mathcal{B}_y$, respectively), and
- $P_x^{(r)} = P_y^{(r)} = P_E$, where P_E is the peer that is responsible for them.

Theorem B.1 (Probability of path intersection)**.** *For any pair of random paths, the probability that there exists a peer P which is responsible for point $P_x^{(i)}$ belonging to the first path and point $P_y^{(i)}$ belonging to the second path is given by*

$$\Pr\{\exists P_E \mid P_x^{(r)} = P_y^{(r)} = P_E\} = \Omega(\mathcal{B}_x, \mathcal{B}_y) = \frac{\Pr\{P_x^{(r)} \in \mathcal{B}_y \wedge P_y^{(r)} \in \mathcal{B}_x\}}{|\mathcal{B}_x \cap \mathcal{B}_y|} \tag{B.2}$$

where the probability at the numerator is conditioned to the events $P_x^{(r)} \in \mathcal{B}_x$ and $P_y^{(r)} \in \mathcal{B}_y$, while $|\mathcal{B}_x \cap \mathcal{B}_y|$ is the number of peers in the intersection of the two regions.

Proof. By representing the numerator of Equation B.2 as the product of two independent events, we have that for any peer P_E which lies in the intersection $\mathcal{B}_x \cap \mathcal{B}_y$, the probability that $P_E = P_x^{(r)}$ is equal to $\frac{\Pr\{P_x^{(r)} \in \mathcal{B}_y | P_x^{(r)} \in \mathcal{B}_x\}}{|\mathcal{B}_x \cap \mathcal{B}_y|}$. The probability that P_E corresponds to both $P_x^{(r)}$ and $P_y^{(r)}$, than, is equal to $\Pr\{P_x^{(r)} \in \mathcal{B}_y\} \cdot \Pr\{P_y^{(r)} \in \mathcal{B}_x\}$. This expression is further multiplied by $|\mathcal{B}_x \cap \mathcal{B}_y|$ to obtain the probability that *at least one* of the peers in the intersection satisfies the constraints. □ □

Now if we consider a malicious peer P_E whose ideal position $P_E^{(i)}$ lies in the path of message SUBMITQ, we can assume independence among $\{P_E = P_i\}_{i=1\ldots n}$, where $P_E = P_i$ represents the event that P_E is also the i-th peer in the path from P' to P_m. A man-in-the-middle attack, then, takes place if P_E is one of the n intermediate peers which route message ANSMETADATA to P_m, as it is defined by the following corollary:

Corollary B.1 (Probability of Censorship)**.** *Let us assume a communication session between two peers P and P', with return address P_m and malicious peer P_E in the path from P to P'. Let $P_E^{(i)}$ be the ideal position of P_E and with $P_1^{(i)}, \ldots, P_n^{(i)}$ the ideal positions of the n peers from P' to P_m.*

Then, the probability $\Gamma(P_E^{(i)}, P_1^{(i)}, \ldots, P_n^{(i)})$ that P_E will censor the communication through a man-in-the-middle attack is given by

$$\Gamma(P_E^{(i)}, P_1^{(i)}, \ldots, P_n^{(i)}) = \sum_{i=1}^{n} \Omega(\mathcal{B}_E, \mathcal{B}_i) \tag{B.3}$$

Proof. This is a direct consequence of Theorem B.1 and the assumption of independence among the events $P_E = P_i$. □ □

In the following, we consider the problem of computing, from a practical point of view, the probability defined by Corollary B.1. In particular we rely on the results presented by Amato et al. in [ARSZ03], where they introduce the concept of *proximity* as a measure of the number of elements that are contained in the intersection of two ball regions when a metric space is defined over the data set.

Let the *distance probability distribution* $F(r)$ be the probability that for any pair of random peers P_1 and P_2 in the network their distance in the space is not greater than r.

$$F(r) = \Pr\{d(P_1, P_2) \leq r\} \tag{B.4}$$

Equation B.4 was first introduced in [CPZ98] to estimate the number of resources from a data set that lie within a certain distance r from the query [CP00b]; in [ARSZ03] it was further exploited to compute the proximity $\mathcal{X}(\mathcal{B}_x, \mathcal{B}_y)$ of any two ball regions, defined as the probability that a (random) resource belongs to both $\mathcal{B}_x$ and $\mathcal{B}_y$.

$$\mathcal{X}(\mathcal{B}_x, \mathcal{B}_y) \approx \int_0^{b_x(d_{xy}, r_x, r_y)} \int_{b_y^1(x, d_{xy}, r_x, r_y)}^{b_y^2(x, d_{xy}, r_x, r_y)} f(x) \cdot f(y) dy dx \tag{B.5}$$

The full details of how Equation B.5 is derived (along with practical hints on its implementation) can be found in [ARSZ03], here we only give the intuition. The joint density $f_{X,Y}(x, y)$ is integrated over the volume of the intersection $\mathcal{B}_x \cap \mathcal{B}_y$ and X and Y represent the events that a resource belongs to $\mathcal{B}_x$ or $\mathcal{B}_y$, respectively. Since X and Y are independent, $f_{X,Y}(x, y) = f_X(x) \cdot f_Y(y)$ holds. Finally notice that the integration bounds $b_x(\cdot)$, $b_y^1(\cdot)$ and $b_y^2(\cdot)$ depend on the specific implementation of the integral.

By relying on these results, Equation B.5 can be used to estimate the value $\Omega(\mathcal{B}_x, \mathcal{B}_y)$ defined in Theorem B.1. Let the proximity $\mathcal{X}$ be a template which can be indifferently applied to both $G_{P_x^{(i)}}(r)$ and $F(r)$. Equation B.2, then, is rewritten as

$$\Omega(\mathcal{B}_x, \mathcal{B}_y) = \frac{\mathcal{X}[G_{P_x^{(i)}}] \cdot \mathcal{X}[G_{P_y^{(i)}}]}{\mathcal{X}[F]} \tag{B.6}$$

Notice that Equation B.6 relies only on two sets of global statistics, namely $F(r)$ and $G(r)$, which can be easily maintained by each peer. Furthermore, the computation resolves to a geometrical problem because only the relative distances between $P_E^{(i)}$ and the ideal positions $P_1^{(i)}, \ldots, P_n^{(i)}$ need to be determined.

Bibliography

[ACM05] Karl Aberer and Philippe Cudré-Mauroux. Semantic overlay networks. In *VLDB Tutorial*, page 1367, Aug. 2005.

[AGP99] Swarup Acharya, Phillip B. Gibbons, and Viswanath Poosala. Aqua: A fast decision support systems using approximate query answers. In *VLDB*, pages 754–757, 1999.

[ARSZ03] Giuseppe Amato, Fausto Rabitti, Pasquale Savino, and Pavel Zezula. Region proximity in metric spaces and its use for approximate similarity search. *ACM Trans. Inf. Syst.*, 21(2):192–227, 2003.

[BBKK97] Stefan Berchtold, Christian Böhm, Daniel A. Keim, and Hans-Peter Kriegel. A cost model for nearest neighbor search in high-dimensional data space. In *Proceedings of the 16th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS'97)*, pages 78–86, Tucson, AZ, May 1997. ACM Press.

[BCLP05] Ilaria Bartolini, Paolo Ciaccia, Alessandro Linari, and Marco Patella. Critical analysis of query processing techniques for heterogeneous environments. Technical Report D3.R2, WISDOM (Web Intelligent Search based on DOMain ontologies) - Italian MIUR Project, 2005. Available at URL `http://dbgroup.unimo.it/wisdom/` (valid as of March 15, 2007).

[BLMT07] Michael Backes, Alessandro Linari, Matteo Maffei, and Christos Tryfonopoulos. Anonymity and censorship resistance in semantic overlay networks. In *Submitted to PET*, 2007.

[Blo70] Burton H. Bloom. Space/time trade-offs in hash coding with allowable errors. *Commun. ACM*, 13(7):422–426, July 1970.

[BM03] Andrei Broder and Michael Mitzenmacher. Network applications of bloom filters: a survey. *Internet Mathematics*, 1(4):485–509, 2003.

[BMT+05] Matthias Bender, Sebastian Michel, Peter Triantafillou, Gerhard Weikum, and Christian Zimmer. Improving collection selection with overlap awareness in P2P search engines. In *SIGIR*, pages 67–74, Aug. 2005.

[BN04] Benjamin Bustos and Gonzalo Navarro. Probabilistic proximity searching algorithms based on compact partitions. *J. Discrete Algorithms*, 2(1):115–134, 2004.

[BNFZ06] Michal Batko, David Novk, Fabrizio Falchi, and Pavel Zezula. On scalability of the similarity search in the world of peers. In *INFOSCALE*, May 2006.

[BNST05] Wolf-Tilo Balke, Wolfgang Nejdl, Wolf Siberski, and Uwe Thaden. DL meets P2P - distributed document retrieval based on classification and content. In *ECDL*, pages 379–390, Sep. 2005.

[Bri95] Sergey Brin. Near neighbor search in large metric spaces. In *Proceedings of the 21st International Conference on Very Large Data Bases (VLDB'95)*, pages 574–584, Zurich, Switzerland, September 1995. Morgan Kaufmann.

[BYRN99] Ricardo A. Baeza-Yates and Berthier A. Ribeiro-Neto. *Modern Information Retrieval*. ACM Press / Addison-Wesley, 1999.

[CCV02] Antonio Corral, Joaquín Cañadas, and Michael Vassilakopoulos. Approximate algorithms for distance-based queries in high-dimensional data spaces using r-trees. In *ADBIS*, pages 163–176, 2002.

[CGM02] Arturo Crespo and Hector Garcia-Molina. Routing indices for peer-to-peer systems. In *Proceedings of the 22nd International Conference on Distributed Computing Systems (ICDCS'02)*, pages 23–32, Vienna, Austria, jul 2002. IEEE Computer Society. Online publication.

[CGM04] Arturo Crespo and Hector Garcia-Molina. Semantic overlay networks for P2P systems. In *AP2PC*, pages 1–13, July 2004.

[CLC95] James P. Callan, Zhihong Lu, and W. Bruce Croft. Searching distributed collections with inference networks. In *SIGIR*, pages 21–28, July 1995.

[CMH+02] Ian Clarke, Scott G. Miller, Theodore W. Hong, Oskar Sandberg, and Brandon Wiley. Protecting free expression online with freenet. *IEEE Internet Computing*, 6(1):40–49, 2002.

[CNBYM01] Edgar Chávez, Gonzalo Navarro, Ricardo Baeza-Yates, and José Luis Marroquín. Proximity searching in metric spaces. *ACM Computing Surveys*, 33(3):273–321, September 2001.

[CP98] Paolo Ciaccia and Marco Patella. Bulk loading the M-tree. In *Proceedings of the 9th Australasian Database Conference (ADC'98)*, pages 15–26, Perth, Australia, February 1998. Springer.

[CP00a] Paolo Ciaccia and Marco Patella. PAC nearest neighbor queries: Approximate and controlled search in high-dimensional and metric spaces. In *Proceedings of the 16th International Conference on Data Engineering (ICDE 2000)*, pages 244–255, San Diego, CA, March 2000. IEEE Computer Society.

[CP00b] Paolo Ciaccia and Marco Patella. Pac nearest neighbor queries: Approximate and controlled search in high-dimensional and metric spaces. In *Proceedings of the 16th International Conference on Data Engineering, ICDE*, pages 244–255. IEEE Computer Society, 2000.

[CP02] Paolo Ciaccia and Marco Patella. Similarity queries. Technical Report CSITE-02-02, CSITE–CNR, 2002.

[CPZ97] 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, Athens, Greece, August 1997. Morgan Kaufmann.

[CPZ98] Paolo Ciaccia, Marco Patella, and Pavel Zezula. A cost model for similarity queries in metric spaces. In *Proceedings of the Seventeenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, PODS*, pages 59–68. ACM Press, 1998.

[CT91] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory.* John Wiley & sons, 1991.

[DA04] George Danezis and Ross Anderson. The economics of censorship resistance. In *Proceedings of Workshop on Economics and Information Security (WEIS04)*, May 2004.

[Dim87] Dimitri P. Bertsekas and Robert G. Gallager. *Data Networks.* Prentice Hall, 1987.

[DNV06a] Christos Doulkeridis, Kjetil Noervaag, and Michalis Vazirgiannis. Scalable semantic overlay generation for P2P-based digital libraries. In *ECDL*, Sep. 2006.

[DNV06b] Christos Doulkeridis, Kjetil Noervaag, and Michalis Vazirgiannis. The SOWES approach to P2P web search using semantic overlays. In *WWW*, pages 1027–1028, May 2006.

[EGKM04] Patrick Th. Eugster, Rachid Guerraoui, Anne-Marie Kermarrec, and Laurent Massoulié. Epidemic information dissemination in distributed systems. *IEEE Computer*, 37(5):60–67, 2004.

[ES03] Dominik M. Endres and Johannes E. Schindelin. A new metric for probability distributions. *IEEE Trans. Inf. Theory*, 49(7):1858–1860, July 2003.

[FGZ05] Fabrizio Falchi, Claudio Gennaro, and Pavel Zezula. A content-addressable network for similarity search in metric spaces. In *DBISP2P*, pages 126–137, Aug. 2005.

[GGMT99] Luis Gravano, Hector Garcia-Molina, and Anthony Tomasic. GlOSS: Text-source discovery over the internet. *ACM Trans. Database Syst.*, 24(2):229–264, June 1999.

[GIS03] Luis Gravano, Panagiotis G. Ipeirotis, and Mehran Sahami. QProber: A system for automatic classification of hidden-web databases. 21(1):1–41, January 2003.

[GMP02] Phillip B. Gibbons, Yossi Matias, and Viswanath Poosala. Fast incremental maintenance of approximate histograms. *ACM Trans. Database Syst.*, 27(3):261–298, Sep. 2002.

[Gnu] The Gnutella project. `http://www.gnutella.com` (valid as in Fef. 2007).

[GR00] Jonathan Goldstein and Raghu Ramakrishnan. Contrast plots and p-sphere trees: Space vs. time in nearest neighbour searches. In *VLDB*, pages 429–440, 2000.

[GRS99] David Goldschlag, Michael Reed, and Paul Syverson. Onion routing. *Commun. ACM*, 42(2):39–41, 1999.

[Gut84] Antonin Guttman. R-trees: A dynamic index structure for spatial searching. In *Proceedings of the 1984 ACM SIGMOD International Conference on Management of Data*, pages 47–57, Boston, MA, June 1984. ACM Press.

[Hel97] Joseph M. Hellerstein. Online processing redux. *IEEE Data Eng. Bull.*, 20(3):20–29, 1997.

[HS03] Gísli R. Hjaltason and Hanan Samet. Index-driven similarity search in metric spaces. *ACM Transactions on Database Systems*, 28(4):517–580, December 2003.

[KNOT06] Panos Kalnis, Wee Siong Ng, Beng Chin Ooi, and Kian-Lee Tan. Answering similarity queries in peer-to-peer networks. *Inf. Syst.*, 31(1):57–72, Mar. 2006.

[KNS02] Irwin King, Cheuk Hang Ng, and Ka Cheung Sia. Distributed content-based visual information retrieval system on peer-to-peer networks. *ACM Transactions on Information Systems*, 22(3):477–501, 2002.

[KW03] Arnd Christian König and Gerhard Weikum. Automatic tuning of data synopses. *Inf. Syst.*, 28(1-2):85–109, Mar. 2003.

[LC03] Jie Lu and James P. Callan. Content-based retrieval in hybrid peer-to-peer networks. In *CIKM*, pages 199–206, Nov. 2003.

[LC05a] Xiaoyong Liu and W. Bruce Croft. Statistical language modeling for information retrieval. *Annual Review of Information Science and Technology*, 39:3–31, 2005.

[LC05b] Jie Lu and Jamie Callan. Federated search of text-based digital libraries in hierarchical peer-to-peer networks. In *ECIR*, pages 52–66, Mar. 2005.

[LC06] Jie Lu and Jamie Callan. User modeling for full-text federated search in peer-to-peer networks. In *SIGIR*, aug 2006.

[LCC+02] Qin Lv, Pei Cao, Edith Cohen, Kai Li, and Scott Shenker. Search and replication in unstructured peer-to-peer networks. In *ICS*, pages 84–95, 2002.

[LTQ+05] Alexander Löser, Christoph Tempich, Bastian Quilitz, Wolf-Tilo Balke, Steffen Staab, and Wolfgang Nejdl. Searching dynamic communities with personal indexes. In *ISWC*, pages 491–505, Nov. 2005.

[LW06] Alessandro Linari and Gerhard Weikum. Efficient peer-to-peer semantic overlay networks based on statistical language models. In *P2PIR*, Nov. 2006.

[MBTW06] Sebastian Michel, Matthias Bender, Peter Triantafillou, and Gerhard Weikum. Iqn routing: Integrating quality and novelty in P2P querying and ranking. In *EDBT*, pages 149–166, Mar. 2006.

[MS05] Peter Mahlmann and Christian Schindelhauer. Peer-to-peer networks based on random transformations of connected regular undirected graphs. In *SPAA*, pages 155–164, July 2005.

[MS06] Peter Mahlmann and Christian Schindelhauer. Distributed random digraph transformations for peer-to-peer networks. In *SPAA*, Aug. 2006.

[MVdV04] S. Mauw, J. Verschuren, and E.P. de Vink. A formalization of anonymity and onion routing. In *Proceedings of ESORICS 2004: 9th European Symposium On Research in Computer Security*, volume 3193, pages 109–124. LNCS, 2004.

[MYL02] Weiyi Meng, Clement T. Yu, and King-Lup Liu. Building efficient and effective metasearch engines. *ACM Comput. Surv.*, 34(1):48–89, Mar. 2002.

[NF04] Henrik Nottelmann and Norbert Fuhr. Combining CORI and the decision-theoretic approach for advanced resource selection. In *ECIR*, pages 138–153, Apr. 2004.

[NF06] Henrik Nottelmann and Norbert Fuhr. Comparing different architectures for query routing in peer-to-peer networks. In *ECIR*, pages 253–264, Apr. 2006.

[PMW05] Josiane Xavier Parreira, Sebastian Michel, and Gerhard Weikum. p2pDating: Real life inspired semantic overlay networks for web search. In *SIGIR workshop on Heterogeneous and Distributed Information Retrieval*, aug 2005.

[Poo97] Viswanath Poosala. *Histogram-Based Estimation Techniques in Database Systems.* PhD thesis, 1997.

[PRL+06] Ivana Podnar, Martin Rajman, Toan Luu, Fabius Klemm, and Karl Aberer. Beyond term indexing: A P2P framework for web information retrieval. *Informatica*, 30(2):153–161, June 2006.

[PRW05] Ginger Perng, Michael K. Reiter, and Chenxi Wang. Censorship resistance revisited. In *Proceedings of Information Hiding Workshop (IH 2005)*, June 2005.

[RKV95] Nick Roussopoulos, Stephen Kelley, and Frédéric Vincent. Nearest neighbor queries. In *Proceedings of the 1995 ACM SIGMOD International Conference on Management of Data*, pages 71–79, San Jose, CA, May 1995. ACM Press.

[RR98] Michael K. Reiter and Aviel D. Rubin. Crowds: anonymity for web transactions. *ACM Transactions on Information and System Security*, 1(1):66–92, 1998.

[Shm04] Vitaly Shmatikov. Probabilistic model checking of an anonymity system. *Journal of Computer Security*, 12(3-4):355–377, 2004.

[SS99] Paul F. Syverson and Stuart G. Stubblebine. Group principals and the formalization of anonymity. In *FM '99: Proceedings of the Wold Congress on Formal Methods in the Development of Computing Systems-Volume I.* Springer-Verlag, 1999.

[SW05] Ralf Steinmetz and Klaus Wehrle. *Peer-to-Peer Systems and Applications.* Springer, 2005.

[WFSP00] Leejay Wu, Christos Faloutsos, Katia P. Sycara, and Terry R. Payne. Falcon: Feedback adaptive loop for content-based retrieval. In *VLDB*, pages 297–306, 2000.

[Yia93] Peter N. Yianilos. Data structures and algorithms for nearest neighbor search in general metric spaces. In *Proceedings of the 4th ACM-SIAM Symposium on Discrete Algorithms (SODA'93)*, pages 311–321, Austin, TX, January 1993. ACM Press.

[ZADB05] Pavel Zezula, Giuseppe Amato, Vlastislav Dohnal, and Michal Batko. *Similarity Search: The Metric Space Approach (Advances in Database Systems).* Springer-Verlag New York, Inc., 2005.

[ZL04] ChengXiang Zhai and John D. Lafferty. A study of smoothing methods for language models applied to information retrieval. *ACM Trans. Inf. Syst.*, 22(2):179–214, Apr. 2004.